Form Methods Syst Des (2010) 36: 97–113
DOI 10.1007/s10703-009-0076-y

On simulation-based probabilistic model checking of mixed-analog circuits

Edmund Clarke · Alexandre Donzé · Axel Legay

Published online: 8 July 2009

Abstract In this paper, we consider verifying properties of mixed-signal circuits, i.e., circuits for which there is an interaction between analog (continuous) and digital (discrete) values. We use a simulation-based approach that consists of evaluating the property on a representative subset of behaviors and answering the question of whether the circuit satisfies the property with a probability greater than or equal to some threshold. We propose a logic adapted to the specification of properties of mixed-signal circuits in the temporal domain as well as in the frequency domain. We also demonstrate the applicability of the method on different models of $\Delta-\Sigma$ modulators for which previous formal verification attempts were too conservative and required excessive computation time.

Keywords Probabilistic model-checking · Simulation-based techniques · Mixed-signals circuits verification · Delta-sigma modulators

This research was sponsored by the GSRC (University of California) under contract no. SA423679952, National Science Foundation under contracts no. CCF0429120, no. CNS0411152, and no. CCF0541245, Semiconductor Research Corporation under contract no. 2005TJ1366, Air Force (University of Vanderbilt) under contract no. 18727S3, International Collaboration for Advanced Security Technology of the National Science Council, Taiwan, under contract no. 1010717, and a grant from the Belgian American Educational Foundation.
Two preliminary versions of this paper appear in the Proceedings of the 4th Haifa Verification Conference and in the Proceedings of the 2nd Workshop on Formal Verification of Analog Circuits, respectively.

E. Clarke
Carnegie Mellon University, Computer Science Department, Pittsburgh, USA

A. Donzé (✉)
VERIMAG Laboratory, 2, Avenue de Vignates, 38610 Gières, France
e-mail: alexandre.donze@imag.fr

A. Legay
INRIA Rennes, Computer Science Department, Rennes, France

1 Introduction

Given a property ϕ, the *Probabilistic Model Checking Problem* consists of checking whether a stochastic system satisfies ϕ with a probability greater than or equal to a certain threshold θ. This problem is generally solved with a *numerical approach* that consists of computing the *exact* probability for the system to satisfy ϕ and by comparing the result to θ. The way the probability is computed depends on the nature of the system as well as on the property that is considered. Successful results (see e.g. [2, 5, 6]) and tools (see e.g. [4, 11]) exist for various classes of systems, including (continuous time) Markov Chains and Markov Decision Processes. The numerical approaches computes the probability of the property by considering all executions of the system. This is one of the drawbacks of these approaches as it may not scale for systems of large size. Another way to solve the probabilistic Model Checking problem is to use a simulation-based approach. The key idea is to infer whether or not the system satisfies the property by observing some of its executions. Of course, in contrast to a numerical approach, a simulation-based solution does not guarantee a correct result. However, it is possible to bound the probability of making an error. Simulation-based methods are known to be far less memory and time-intensive than numerical ones, and are sometimes the only option [25].

In this paper, we consider applying the simulation-based procedure proposed by Younes [26, 27, 29, 30] to verify properties of *mixed-signal circuits*, i.e., circuits for which there is an interaction between analog (continuous) and digital (discrete) values. Our first contribution is to propose a version of stochastic discrete-time event systems that fits into the framework introduced by Younes with the additional advantage that it explicitly handles analog and digital signals. We also introduce *probabilistic signal linear temporal logic*, a logic adapted to the specification of properties for mixed-signal circuits in the *temporal* domain and in the *frequency* domain.

Our second contribution is the analysis of a class of Δ–Σ modulators. A Δ–Σ modulator is an efficient *Analog-to-Digital Converter circuit*, i.e., a device that converts analog signals into digital signals. A common critical issue in this domain is the analysis of the *stability* of the internal state variables of the circuit. The concern is that the values that are stored by these variables can grow out of control until they reach a maximum value, at which point we say that the circuit *saturates*. Saturation is commonly assumed to compromise the quality of the analog-to-digital conversion. In [7] and [10], reachability techniques developed in the area of hybrid systems are used to analyze the stability of a third-order modulator. Their idea is to use such techniques to guarantee that for *every* input signal in a given range, the states of the system remain stable. While this reachability-based approach is sound, it has important drawbacks such as (1) signals with long duration cannot be practically analyzed, and (2) properties that are commonly specified in the frequency domain rather than in the time domain cannot be checked. Our results show that a simulation-based approach makes it possible to handle properties and signals that are beyond the scope of the reachability-based approach. In our experiments, we analyze discrete-time signals with 24000 sampling points in seconds, while the approach in [7] takes hours to analyze signals with up to 31 sampling points. We are also able to provide insight into a question left open in [7] by observing that saturation does not always imply an improper signal conversion. This can be done by comparing the Fourier transform of each of the input analog signals with the Fourier transform of its corresponding digital signal. Such a property can easily be expressed in our logic and Model Checked with our simulation-based approach. We are unaware of other formal verification techniques that can solve this problem.

Structure of the paper. In Sect. 2, we recap some basic knowledge about signal theory. Section 3 introduces Younes' approach of verification using hypothesis testing. Our model

and logic are introduced in Sect. 4. Issues related to Δ–Σ modulators and their verification are discussed in Sect. 5. Section 6 presents the experiments we conducted. Finally, Sect. 7 concludes the paper.

2 Signal definitions

We use $\mathbb{N}$, $\mathbb{R}$, and $\mathbb{C}$ to denote the sets of natural, real, and complex numbers, respectively. Let the *time set* $\mathcal{T}$ be a finite set of non-negative real numbers $\{t_0, t_1, \ldots, t_{N-1}\}$, where $N \in \mathbb{N}$. To simplify the presentation, we assume that $t_{i+1} - t_i = \delta t$, where $\delta t \in \mathbb{R}_{>0}$. A *digital set* is a set with 2^b elements, i.e., an element of the set can be encoded by b bits. A *frequency set* is a subset of $\mathbb{R}$. An *analog signal* is a mapping $\xi : \mathcal{T} \to \mathbb{R}$. A *digital signal* is a mapping $\xi : \mathcal{T} \to \mathcal{D}$, where $\mathcal{D}$ is a digital set. A *frequency-domain* signal is a mapping $\hat{\xi} : \mathcal{F} \to \mathbb{C}$, where $\mathcal{F}$ is a frequency set. The value at time $t \in \mathcal{T}$ of a signal ξ is denoted by $\xi[t]$. Let $t, t' \in \mathcal{T}$, the *restriction* of a signal ξ to $[t, t']$, denoted by $\xi_{|_{[t,t']}}$, is a signal such that:

$$\xi_{|_{[t,t']}}[\tau] = \begin{cases} \xi[\tau] & \text{if } \tau \in [t, t'], \\ 0 & \text{else.} \end{cases}$$

The restriction of a frequency-domain signal to an interval of frequencies is defined similarly.

The *Fourier transform* (see [20]) is a functional F that maps a time-domain signal $\xi : \mathcal{T} \to \mathbb{R}$ to a *frequency-domain* signal $\hat{\xi} = F(\xi)$. The *inverse Fourier transform* is used to "reconstruct" ξ from $\hat{\xi}$, i.e., $\xi = F^{-1}(\hat{\xi})$. Formally, for all ν in $\mathcal{F}$ and for all t in $\mathcal{T}$ we have

$$F(\xi)[\nu] = \int_{\mathcal{T}} \xi[t] e^{-\mathrm{i}2\pi\nu t} dt \quad \text{and} \quad F^{-1}(\hat{\xi})[t] = \xi[t] = \int_{\mathcal{F}} \hat{\xi}[\nu] e^{\mathrm{i}2\pi\nu t} d\nu.$$

An efficient algorithm known as the *Fast Fourier Transform algorithm* (see, e.g., [9]) is used to compute a discrete approximation of the Fourier transform.

Remark 1 There are many operations that are easier to perform in the frequency domain than in the time domain, e.g., convolution and differentiation. For these operations, it is convenient to compute Fourier transforms, perform the desired operation in the frequency domain and use the inverse Fourier transform to get the desired result. The Fourier transform is also useful when dealing with signals that are easier to analyze in the frequency domain than in the time domain (e.g., sound as shown in Fig. 1).

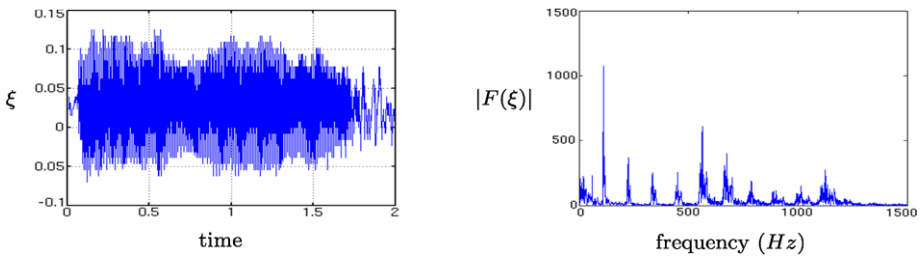

Fig. 1 Example of a Fourier transform. The signal was obtained by recording a human voice. Its Fourier transform lies in the interval [0 Hz, 1500 Hz] (its value is 0 outside this interval)

3 Solving the probabilistic model checking problem with a simulation-based approach

3.1 The probabilistic model checking problem

We use $\Pr(E)$ to denote the probability of event E. We consider a stochastic system $\mathcal{S}$ which has a unique initial state and a linear property ϕ. The executions of $\mathcal{S}$ are finite, observable, and can be generated "on demand". We also assume that one can decide in finite time whether an execution of $\mathcal{S}$ satisfies ϕ.

The *Probabilistic Model Checking Problem* consists of deciding whether $\mathcal{S}$ will satisfy ϕ with a probability greater than or equal to a given threshold θ. The latter is denoted by $\mathcal{S} \models \Pr_{\geq\theta}(\phi)$. The probability for $\mathcal{S}$ to satisfy ϕ is denoted $\Pr(\mathcal{S} \models \phi)$, or $\Pr(\phi)$ when $\mathcal{S}$ is clear from the context.

The Probabilistic Model Checking Problem is well-defined if and only if one can assign a probability to the set of executions of $\mathcal{S}$ that satisfy ϕ. One way to solve the problem is to use a *numerical approach* that consists of computing the *exact* probability for the system to satisfy ϕ and comparing the result to θ. The way the probability is computed depends on the nature of the system as well as on the property. Successful results (see e.g. [2, 5, 6]) and tools (see e.g. [4, 11]) exist for various classes of systems, including (continuous time) Markov Chains and Markov Decision Processes.

The main drawback of numerical approaches to probabilistic verification is that it has to compute the probability of satisfying the formula with respect to all the executions of the system. In contrast, *simulation-based* methods generate a finite number of executions from the model to determine whether or not it satisfies a given property. Simulation-based methods can estimate the probability that a property holds to an arbitrarily high degree of confidence, although the number of required executions increases with the desired confidence. In comparison to numerical methods, simulation-based methods are usually more efficient and scale to larger systems because they are not exhaustive in nature [25]. Another advantage is that the ability to reason on one execution at a time makes it possible to decide a larger class of temporal properties.

3.2 The statistical model checking approach

Recently, a simulation-based approach to the Probabilistic Model Checking Problem has been been introduced by Younes and Simmons [26, 27, 29, 30]. This approach is based on hypothesis testing. The idea is to check the property ϕ on a sample set of simulations and to decide whether the system satisfies $\Pr_{\geq\theta}(\phi)$ based on the number of executions for which ϕ holds compared to the total number of executions in the sample set. With such an approach, we do not need to consider all the executions of the system. Let $p = \Pr(\phi)$, to determine whether $p \geq \theta$, we can test the hypothesis $H : p \geq \theta$ against $K : p < \theta$. A simulation-based solution does not guarantee a correct result but it is possible to bound the probability of making an error. The *strength* (α, β) of a test is determined by two parameters, α and β, such that the probability of accepting K (respectively, H) when H (respectively, K) holds, called a Type-I error (respectively, a Type-II error) is less or equal to α (respectively, β).

A test has *ideal performance* if the probability of the Type-I error (respectively, Type-II error) is exactly α (respectively, β). However, these requirements make it impossible to ensure a low probability for both types of errors simultaneously (see [27] for details). A solution to this problem is to relax the test by working with an *indifference region* (p_1, p_0) with $p_0 \geq p_1$ ($p_0 - p_1$ is the *size of the region*). In this context, we test the hypothesis $H_0 : p \geq p_0$

against $H_1 : p \leq p_1$ instead of H against K. If the value of p is between p_1 and p_0 (the indifference region), then we say that the probability is sufficiently close to θ so that we are indifferent with respect to which of the two hypotheses K or H is accepted. The thresholds p_0 and p_1 are generally defined in term of the single threshold θ, e.g., $p_1 = \theta - \delta$ and $p_0 = \theta + \delta$. We now need to provide a test procedure that satisfies the requirements above. In the next two subsections, we recall two solutions proposed by Younes in [27, 31].

3.2.1 Single sampling plan

Let B_i be a discrete random variable with a Bernoulli distribution of parameter p. Such a variable can only takes two values 0 and 1 with $\Pr[B_i = 1] = p$ and $\Pr[B_i = 0] = 1 - p$. In our context, each variable B_i is associated with one simulation of the system. The outcome for B_i, denoted b_i, is 1 if the simulation satisfies ϕ and 0 otherwise. To test hypothesis H_0 against hypothesis H_1, we specify a constant c. If $\sum_{i=1}^{n} b_i$ is larger than c, then H_0 is accepted, else H_1 is accepted. The difficult part in this approach is to find values for the pair (n, c), called a *single sampling plan*, such that the two error bounds α and β are respected. In practice, one tries to work with the smallest value of n possible so as to minimize the number of simulations performed. This results in an optimization problem, which generally does not have a closed-form solution except for a few special cases (see [27] for a discussion). In his thesis [27], Younes proposes a binary search based algorithm (Algorithm 2.1, page 21) that, given p_0, p_1, α, β, computes an approximation of the minimal value for c and n.

3.2.2 Sequential probability ratio test

The sample size for a single sampling plan is fixed in advance and independent of the observations that are made. However, taking those observations into account can increase the performance of the test. As an example, if we use a single plan (n, c) and the $m > c$ first simulations satisfy the property, then we could (depending on the error bounds) accept H_0 without observing the $n - m$ other simulations. To overcome this problem, Younes proposed a procedure to test $H_0 : p \geq p_0$ against $H_1 : p \leq p_1$ that is based on the *sequential probability ratio test* proposed by Wald [24]. The approach is briefly described below.

In the sequential probability ratio test, one has to choose two values A and B, with $A > B$. These two values should be chosen to ensure that the strength of the test is respected. Let m be the number of observations that have been made so far. The test is based on the following quotient:

$$\frac{p_{1m}}{p_{0m}} = \prod_{i=1}^{m} \frac{\Pr(B_i = b_i \mid p = p_1)}{\Pr(B_i = b_i \mid p = p_0)} = \frac{p_1^{d_m}(1 - p_1)^{m - d_m}}{p_0^{d_m}(1 - p_0)^{m - d_m}}, \tag{1}$$

where $d_m = \sum_{i=1}^{m} b_i$. The idea behind the test is to accept H_0 if $\frac{p_{1m}}{p_{0m}} \geq A$, and H_1 if $\frac{p_{1m}}{p_{0m}} \leq B$. An algorithm for sequential ratio testing consists of computing $\frac{p_{1m}}{p_{0m}}$ for successive values of m until either H_0 or H_1 is satisfied. This has the advantage of minimizing the number of simulations. In each step i, the algorithm has to check the property on a single execution of the system, which is handled with a new Bernoulli variable B_i whose realization is b_i. In his thesis [27], Younes proposed a logarithmic based algorithm (Algorithm 2.3 page 27) SPRT that given p_0, p_1, α and β implements the sequential ratio testing procedure. Computing ideal values A_{id} and B_{id} for A and B in order to make sure that we are working with a test of strength (α, β) is a laborious procedure (see Sect. 3.4 of [24]). In his seminal paper [24], Wald showed that if one defines $A_{id} \geq A = \frac{(1-\beta)}{\alpha}$ and $B_{id} \leq B = \frac{\beta}{(1-\alpha)}$, then we obtain a new test whose strength is (α', β'), but such that $\alpha' + \beta' \leq \alpha + \beta$, meaning that either $\alpha' \leq \alpha$ or $\beta' \leq \beta$. In practice, we often find that both inequalities hold.

3.2.3 Additional information on Younes' work

We briefly mention other contributions by Younes on Statistical Model Checking.

1. *Nested operators.* Younes' algorithm has been extended to handle formulas where ϕ can also contain probabilistic operators. We will not use this extension in the paper.
2. *Black-box systems.* Sequential testing is not always appropriated. Indeed, there are some systems, called *black-box systems*, that cannot be controlled to generate executions on demand (which means that they are not totally observable) and for which we are only given a set of trajectories generated during actual execution of the system. In such a case, one may prefer to use a non-sequential approach, such as an extension of the single sampling plan introduced above. This has been investigated by Younes in [26] and by Sen et al. in [22, 23].
3. *Distributed implementation.* SPRT can be implemented in a distributed manner. This has been investigated by Younes in [28]. To ensure the independence of the simulations, Younes used the scheme proposed in [13].
4. *Complexity.* Younes showed that the complexity of SPRT depends on the number of simulations needed to reach a decision, the time needed to check whether a given execution satisfies ϕ and the time needed to generate a simulation.

4 Model and logic

4.1 Stochastic signal discrete-time event systems

Our main motivation is to verify properties of mixed-signal circuits. For this purpose, we define *stochastic signal discrete-time event systems*, which extend the classical stochastic discrete-time event systems with information about signals. During an execution, these systems have to remain in the same state between the occurrence of two events. The signals associated with each execution are thus piecewise-constant.

Definition 1 *Stochastic signal discrete-time event system* (SSDES) is a tuple $\mathcal{S} = (\mathcal{T}, S, s_0, \rightarrow, \pi_a, \pi_d, \mathcal{B}, L)$ where

- $\mathcal{T}$ is a finite set of non-negative real numbers $\{t_0, t_1, \ldots, t_{N-1}\}$, with $t_{i+1} - t_i = \delta t > 0$;
- S is the set of states, defined as $S = A_s \times D_s$, where $A_s \subset \mathbb{R}^{n_a}$ and $D_s \subset \mathcal{D}^{n_d}$, n_a and n_d being the number of analog and digital values which define the state of the system;
- $s_0 \in S$ is the initial state;
- The relation $\rightarrow: S \times S$ is the transition relation of the system. We assume that for every state s, the relation $\rightarrow$ induces a probability distribution, i.e.,

$$\forall s \in S, \quad \int_S \Pr(s \rightarrow s' | s' \in dS)\, dS = 1;$$

- $\pi_a : S \times \{1, \ldots, n_a\} \rightarrow A_s$ is a *projection operator* such that for all $s = (s_a^1, \ldots, s_a^{n_a}, s_d^1, \ldots, s_d^{n_d})$ and $1 \leq j \leq n_a$, $\pi_a(s, j) = s_a^j$;
- π_d is defined in a similar manner as π_a;
- $\mathcal{B}$ is a finite set of Boolean propositions;
- L is a mapping from S to $2^{\mathcal{B}}$, which assigns to each state the elements in $\mathcal{B}$ that are true in that state. If $b \in L(s)$, then we say that s satisfies b.

Given $y \geq 0$, we use S^y to denote the set of all sequences of y states. Let $\omega = s_0 \dots s_{k-1}$ be a finite sequence of k states of $\mathcal{S}$. Given $0 \leq i \leq k-1$, we use $\omega(i)$ and ω^i to denote the i-th state of ω and the i-th *suffix* $s_i \dots s_k$, respectively. The i-th *prefix* of ω, denoted ω_i is the sequence $s_0 \dots s_{i-1}$. The length of ω, denoted $|\omega|$, is the number of states in ω. An *execution* of an SSDES $\mathcal{S} = (\mathcal{T}, S, s_0, \rightarrow, \pi_a, \pi_d, L)$ is a sequence of N states $\sigma = s_0 s_1 \dots s_{N-1}$ such that s_0 is the initial state of the system and for each $i \in 0 \dots N-1$, $s_i \in S$ and $s_i \rightarrow s_{i+1}$. Our model is assumed to have the Markovian property, i.e., the probability of a transition from a state s_i to a state s_{i+1} in the execution depends on s_i and nothing else. An SSDES is thus an infinite-state Markov chain equipped with information and operations on analog and digital signals.

Relation between executions and signals: To each execution σ of $\mathcal{S}$, we associate n_a analog signals denoted by $\xi_a^1(\sigma), \dots, \xi_a^{n_a}(\sigma)$ and n_d digital signals denoted by $\xi_d^1(\sigma), \dots, \xi_d^{n_a}(\sigma)$. At each time instant t_k, each of these signals takes the analog or digital value which is the corresponding component of the state $\sigma(k)$. For example, if $\sigma = s_0 s_1 \dots s_{N-1}$ then we have

$$\xi_a^1(\sigma)[t_0] = \pi_a(s_0, 1), \quad \xi_a^1(\sigma)[t_1] = \pi_a(s_1, 1), \quad \dots, \quad \xi_a^1(\sigma)[t_{N-1}] = \pi_a(s_{N-1}, 1).$$

4.2 Probabilistic signal linear temporal logic

We introduce the *probabilistic signal linear temporal logic* (SLTL) to reason about the set of executions of an SSDES $\mathcal{S} = (\mathcal{T}, S, S_0, \rightarrow, \pi_a, \pi_d, \mathcal{B}, L)$. We first introduce the definition of a *y-sequence predicate*.

Definition 2 Given $y \in [1, N-1]$, a y-sequence predicate for $\mathcal{S}$ is a predicate on a sequence of y states, i.e., on S^y.

In the rest of the paper, we use $\mathcal{P}_y$ to denote a set of y-sequence predicates, with $1 \leq y \leq N-1$. Observe that a predicate defined on the entire execution must belong to $\mathcal{P}_N$, such a predicate is referred to as an *execution predicate*.

Example 1 Consider an execution predicate p that decides whether the mean value of the first analog signal associated with an execution σ of an SSDES is greater than 0. Such predicate can be defined as

$$p(\sigma) = \mathbf{T} \quad \text{iff} \quad \frac{1}{N} \sum_{k=0}^{N-1} \xi_a^1(\sigma)[t_k] \geq 0.$$

This example shows that the definition of sequence predicate makes it easy to define properties on entire executions that are not so simple to define with temporal operators. In Sect. 6, we consider several other examples of sequence and execution predicates.

We now introduce the syntax and the semantics for a version of the linear temporal logic (LTL) of Pnueli [17] whose atoms are either Boolean propositions or y-sequence predicates. The syntax of LTL is given by the following grammar:

$$\phi ::= \mathbf{T} \,|\mathbf{F}\, |b \in \mathcal{B} \,|\, \phi_1 \vee \phi_2 \,|\, \phi_1 \wedge \phi_2 \,|\neg\phi \,|\, \bigcirc \phi \,|\, \phi_1 \mathcal{U} \phi_2 \,|\, \phi_1 \widetilde{\mathcal{U}} \phi_2 \,|$$
$$p_y \in \mathcal{P}_y \text{ and } y \in [1, N-1].$$

We now present the semantics of LTL, which here is defined with respect to finite sequences of states of $\mathcal{S}$. Since we define the semantic of LTL properties on finite sequences, we can only specify *bounded LTL properties*.[1]. As in [27], we thus stay in the class of safety properties. The fact that a finite sequence of states ω of $\mathcal{S}$ satisfies the LTL property ϕ is denoted by $\omega \models \phi$. We have the following:

- $\omega \models \mathbf{T}$ and $\omega \not\models \mathbf{F}$;
- $\omega \models b$ with $b \in \mathcal{B}$ if and only if $b \in L(\omega(0))$;
- $\omega \models \phi_1 \vee \phi_2$ if and only if $\omega \models \phi_1$ or $\omega \models \phi_2$;
- $\omega \models \phi_1 \wedge \phi_2$ if and only if $\omega \models \phi_1$ and $\omega \models \phi_2$;
- $\omega \models \neg\phi$ if and only if $\omega \not\models \phi$;
- $\omega \models \bigcirc\phi$ if and only if $|\omega| > 1$ and $\omega^1 \models \phi$;
- $\omega \models \phi_1 \mathcal{U} \phi_2$ if and only if there exists $0 \leq i \leq |\omega| - 1$ such that $\omega^i \models \phi_2$, and for each $0 \leq j < i$, $\omega^j \models \phi_1$;
- $\omega \models \phi_1 \widetilde{\mathcal{U}} \phi_2$ if and only if for each $0 \leq i \leq |\omega| - 1$ such that $\omega^i \not\models \phi_2$ there exists $0 \leq j < i$ such that $\omega^j \models \phi_1$;
- $\omega \models p_y$ if and only if $|\omega| \geq y$ and p_y is true for ω_y.

We denote $\mathbf{T}\mathcal{U}\psi$ by $\Diamond\psi$, denote $\mathbf{F}\widetilde{\mathcal{U}}\psi$ by $\Box\psi$, and for $k \geq 0$, denote $\overbrace{\bigcirc \ldots \bigcirc}^{k \text{ times}}\phi$ by $\bigcirc^k\phi$. Observe that, using this semantic, it is easy to decide whether a finite execution satisfies an LTL formula.

We can now define probabilistic signal linear temporal logic.

Definition 3 (SLTL formula) An *SLTL formula* is a formula of the form $\psi = \Pr_{\geq\theta}(\phi)$, where ϕ is an LTL formula and $\theta \in [0, 1]$.

We say that $\mathcal{S}$ satisfies ψ, denoted by $\mathcal{S} \models \psi$ if and only if the probability for an execution of $\mathcal{S}$ to satisfy ϕ is greater than or equal to θ. The problem is well-defined since we can prove that one can always assign a unique probability measure to the set of executions that satisfy an LTL formula with sequence predicates. Before presenting this result, we first introduce the definition of *witness expansion* for a Markov chain.

Definition 4 Consider the Markov chain $\mathcal{S} = (S, s_0, \rightarrow, L)$; its witness expansion is the Markov chain $\mathcal{S}' = (S', s'_0, \rightarrow', L')$, where

- Each state in S' is a prefix of one of the executions of $\mathcal{S}$;
- $s'_0 = s_0$;
- The transition relation $\rightarrow'$ is defined as follows: (s_x, s_y) belongs to $\rightarrow'$ if and only if $s_x = s_0 \ldots s$, $s_y = s_0 \ldots s s'$, and $(s, s') \in \rightarrow$. The probability distribution from s_x is the probability distribution from s;
- Given a state $s = s_0 s_1 \ldots s_i$ in S', $L'(s) = L(s_i)$.

We now prove our result.

Theorem 1 *Let $\mathcal{S}$ be an SSDES and ϕ be an LTL property with sequence predicates. One can always associate a unique probability measure with the set of executions of $\mathcal{S}$ that satisfy ϕ.*

[1] A bounded LTL property (see [3] for details) is an LTL property with a bound on the scope of its temporal operators $\mathcal{U}$ and $\widetilde{\mathcal{U}}$. To simplify the presentation, we implicitly introduce this bound directly in the semantics.

Proof An SSDES is an infinite-state Markov chain. Each execution of a SSDES can be viewed as infinite execution by considering its last state to be an absorbing state, i.e., a state in which the system stays forever. In [27], it is shown that one can assign a unique probability measure to sets of infinite executions of such a Markov chain using a *probability space* and the classical notion of *basic cylinder*. In [27], it is also shown that this probability distribution is sufficient to assign a probability to the set of executions that satisfy an LTL formula without sequence predicates. We are now left with the case where ϕ can refer to y-sequence predicates. In this case, we first derive from $\mathcal{S}$ its corresponding witness expansion $\mathcal{S}'$. It is easy to see that there is a one-to-one correspondence between the executions of $\mathcal{S}$ and those of $\mathcal{S}'$. We then introduce a new Boolean variable p_i for each y-sequence predicate P_i. Given a state $s_{\mathcal{S}'}$ of $\mathcal{S}'$, we have $p_i \in L(s_{\mathcal{S}'})$ if and only if (1) $s_{\mathcal{S}'}$ is the prefix of an execution of $\mathcal{S}$ (2) $|S_{\mathcal{S}'}| \geq y$ and (3) the sequence formed by the y last states of $S_{\mathcal{S}'}$ satisfies P_i. Let ϕ' be the formula ϕ where each y-execution predicate has been replaced by the LTL formula $\bigcirc^{y-1} p$, where p is the Boolean variable associated with the predicate. The formula ϕ' is an LTL formula. Observe that an execution of $\mathcal{S}$ satisfies ϕ if and only if its corresponding execution in $\mathcal{S}'$ satisfies ϕ'. Since ϕ' is an LTL formula, one can always assign a probability to the set of executions of $\mathcal{S}'$ that satisfy it. By construction, we know that this probability is also the one assigned to the set of executions of $\mathcal{S}$ that satisfy ϕ. □

4.3 Solving the probabilistic model checking for SSDES and SLTL

As we already observed, an SSDES is a Markov chain. If we assume that the executions of the SSDES can be generated on demand and that one can always decide whether an execution satisfies an LTL formula, then one can use Younes' or Peyronnet's techniques to decide whether the SSDES satisfies the property up to some confidence. We use this approach in the paper.

5 A class of mixed-signal circuits: Δ–Σ modulators

This section is a brief introduction to the principles of Δ–Σ *modulation* and the related design issues. The reader can consult [15, 21] for more details on this topic in Signal Processing.

5.1 Analog to digital conversion via Δ–Σ modulation

A Δ–Σ modulator is an *Analog-to-Digital Converter circuit*, i.e., a circuit that takes an analog value $u \in \mathbb{R}$ as input and encodes it into a digital value $v \in \mathcal{D}$. Since digital signal processing is more widely used than analog signal processing, such converters are found in many electrical devices, which motivates their study. The challenge with Analog-to-Digital conversion is to represent the uncountable set of analog values using a finite set of digital values $\mathcal{D}$. The direct approach, which is called *quantization*, consists in mapping u to the digital value v that minimizes the *quantization error* defined as $\delta = u - v$, i.e., it chooses $v = \text{argmin}_{v \in \mathcal{D}} |\delta|$. Obviously, one way to decrease the remaining quantization error is to increase the number of bits used to encode $\mathcal{D}$ and thus the number of possible digital values. Another approach, which is implemented by Δ–Σ modulation, is to measure and compensate for the accumulation of quantization errors during time. As an example, consider the following simple instance of a discrete time Δ–Σ modulator. Let $u(k)$, $v(k)$, $\delta(k) = u(k) - v(k)$ be the analog input, the digital output, and the quantization error at step k,

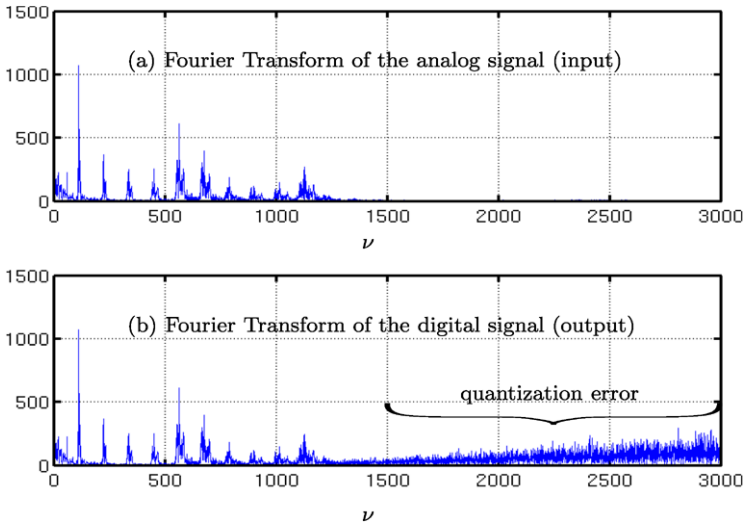

Fig. 2 A sample behavior of the Δ–Σ modulator. The Fourier transform of the output signal (**b**) matches the Fourier transform of the input signal (**a**) on the interval [0, 1500 Hz]. The quantization error is pushed toward frequencies higher than 1500 Hz

respectively. The modulator uses an *integrator* to store the accumulation of errors in a variable $x(k) = \sum_0^k \delta(k)$, so that $x(k+1) = x(k) + \delta(k)$, and determines the next digital output $v(k+1)$ based on the sign of $x(k+1)$, i.e., $\mathcal{D} = \{-1, 1\}$ and $v(k+1) = 1$ if $x(k+1) \geq 0$ and $v(k+1) = -1$ otherwise. A Δ–Σ modulator thus basically consists of a feedback loop controlling the quantization error. To improve the performance, more complex feedback loops can be designed involving more than one integrator. The *order* of a modulator is given by the number of integrators used. Note that Δ–Σ modulators can achieve good performance using a limited number of bits.

5.2 Conversion interpretation in the frequency domain

The benefit of the Δ–Σ modulation approach is clearly apparent in the frequency domain. Indeed, the Fourier transform of the digital signal is the Fourier transform of the analog signal composed with some error due to quantization. The feedback loop in the Δ–Σ modulator is designed to "push" this error towards high frequencies, where it can be isolated and removed, e.g. by using a low-pass filter. The original signal can then be retrieved by using the inverse Fourier transform. The process assumes that the input signal has a limited bandwidth (e.g. in Fig. 2, the Fourier transform of the input signal is zero for frequencies higher than 1500 Hz). Usually, a *low-pass filter* is placed between the analog signal and the Δ–Σ modulator to ensure that the input signal always satisfies this requirement.

Example 2 An illustration of Δ–Σ modulator principle is given in Fig. 2. The plots show the Fourier transform of the digital output of a Δ–Σ modulator for the signal of Fig. 1.

5.3 Verification issues and reachability-based verification

Modulators with more than two integrators are known to exhibit better performance but also introduce a *stability* problem [1]. During an execution, an integrator remembers each input

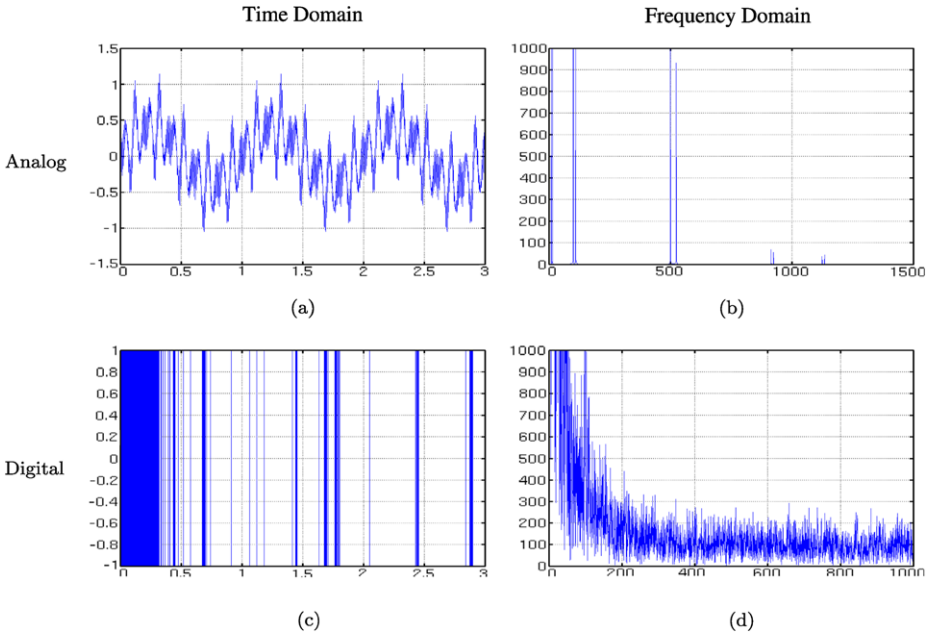

Fig. 3 An example where the Δ–Σ fails. We observe that the Fourier transform of the digital signal (**d**) is clearly different from the Fourier transform of the analog signal (**b**)

and adds it to the sum of all the previously read inputs. Consequently, an important issue is whether the integrators are stable, i.e., whether or not the values stored in the integrators can grow indefinitely. Because integrators have limited capacity, the values of these states would then reach a *saturation level*. Saturation can compromise the quality of the analog-to-digital conversion. In Fig. 3, a behavior for which the conversion failed is presented. The stability analysis of the feedback loop is made difficult by the nonlinearity (in this case, a discontinuity) induced by quantization. This invalidates the direct application of classical linear stability theory which makes the stability analysis of Δ–Σ modulators a challenging problem (see [19]).

State of the art (brief overview). In [7] and [10], the authors use reachability techniques developed in the area of hybrid systems to guarantee that for *every* input signal in a given range, the integrator state will never saturate. However, the technique developed in [7] is not powerful enough to analyze signals with long durations. Moreover, the approach is restricted to saturation and cannot be applied to more complex properties such as those related to the "quality" of the conversion. In the next section, we investigate these issues using an SSDES representation and the techniques introduced in Sect. 3.

6 Experimental results

We have implemented a prototype of a statistical Model Checker in the MATLAB environment. Our procedure takes as input a MATLAB routine that returns an execution of a given SSDES model and a routine that can decide whether an execution satisfy an LTL property ϕ or not. It implements a simple bisection procedure, called BI-SPRT, that makes several iterations of the SPRT algorithm to quickly find values of θ for which the system satisfies

the formula $\Pr_{\geq\theta}(\mathcal{S} \models \phi)$. In this section, we discuss the results we obtained when applying our prototype to different Δ–Σ modulators. We used the `delsig` toolbox [18] to generate models of order 3, 5 and 7. It provides a simulation routine such that given an input signal and a model returns a digital signal.

6.1 On representing Δ–Σ modulators and properties with SSDES

An SSDES is a stochastic system whereas a Δ–Σ modulator is a nondeterministic system due to the input signal. To represent a Δ–Σ modulator with an SSDES, we thus need to resolve this nondeterminism by providing a stochastic generator for the input signals. We first describe the SSDES and then discuss the definition of the stochastic input generator.

6.1.1 SSDES representation

We define an SSDES model $\mathcal{S} = (\mathcal{T}, S, s_0, \rightarrow, \pi_a, \pi_d, L)$ for a n^{th} order Δ–Σ modulator combined with a stochastic input generator as follows:

- *Time.* We set $\mathcal{T} = \{t_0, t_1, \ldots, t_{N-1}\}$ with $t_0 = 0$, $t_{N-1} = 3$ and $\delta t = t_{i+1} - t_i = \frac{1}{8000}$, $N = 24000$;
- *Set of states.* The model contains n integrators such that each contains one real-valued (or analog) variable. A state $s \in S$ can thus be described as a tuple $(u, x_1, x_2, \ldots, x_n, v)$, where
  - $x_1, \ldots, x_n$ are analog variables storing the integrators' states;
  - u is an analog variable storing values for the input signal ξ^u;
  - v is a digital variable storing values for the output signal ξ^v.

  The number of analog signals is thus $n_a = n + 1$ and the number of digital signals $n_d = 1$. We assume that the states of the integrators cannot go beyond certain values that are fixed by the model. When this value is reached, we say that the integrators saturate. In practice, $x_i \in [-1, 1]$ for $i \in \{1, 2, \ldots, n\}$ and $-1, 1$ are the *saturation values*. If we assume that $u \in [-u_{\max}, u_{\max}]$, we get $A_s = [-1, 1]^n \times [-u_{\max}, u_{\max}]$ and $D_s = \{-1, 1\}$. Given an execution $\sigma = s_0 s_1 \ldots s_{N-1}$, we use $u(k) = \pi_a(s_k, 1)$, $x_i(k) = \pi_a(s_k, i + 1)$, and $v(k) = \pi_d(s_k, 1)$. For all $k \in \{0, \ldots, N - 1\}$, we have $\xi^u(\sigma)[t_k] = u(k)$ and $\xi^v(\sigma)[t_k] = v(k)$;
- *Transition relation.* When $u(k)$ is given, the simulation routine computes $x_1(k+1), x_2(k+1), \ldots, x_n(k + 1)$ and $v(k + 1)$. Thus the probability distribution of $s_k \rightarrow s_{k+1}$ for all $(s_k, s_{k+1}) \in S \times S$ is induced by the probability distribution of the input value $u(k + 1)$. The choice of this probability distribution is discussed below.
- *Initial state.* Initially, the values of the integrators' states are 0 and by convention the digital output $v(0)$ is set to 1 and the input value $u(0)$ to 0. Thus the initial state is $s_0 = (0, \ldots, 0, 1)$;
- *Boolean predicates.* We define a Boolean predicate *Satur* which is associated with a state s if and only if one of the analog components of s, i.e., either the input or an integrator state, saturates. Formally, $\mathit{Satur} \in L(s)$ if and only if there exist i in $\{1, \ldots, n_a\}$ such that $\pi_a(s, i) = 1$ or $\pi_a(s, i) = -1$.

6.1.2 Discussion on the stochastic input generator

In [7], the circuit is analysed for all input signals with an amplitude less or equal to some value $u_{\max}$. To get comparable results with our probabilistic approach, we have to define the stochastic input generator so that it samples uniformly the set of all signals with an amplitude bounded by $u_{\max}$. A simple way to do this is as follows: for all k, $u(k)$ is chosen in a set $[-u_{\max}, u_{\max}]$ with a uniform random distribution.

Table 1 Saturation analysis of a 3th order Δ–Σ modulator with $\alpha = 0.001$, $\beta = 0.001$ and $\delta = 0.02$

$u_{\max}$	Nb true	Nb trials	Proba. found	Computational time (s)
0.1	0	342	0	1.54
0.15	321	4847	0.0625	19.23
0.2	14985	29229	0.5	116.34
0.25	881	898	0.96875	3.61
0.3	342	342	1	1.35849

However this class of signals is too broad in practice. As mentioned in Sect. 5.2, the modulators are designed to work better for signals with a given bandwidth of frequencies. More precisely, an important parameter in the design of a Δ–Σ modulator is its *over-sampling ratio* (*OSR*) [18]. This parameter means that the modulator is optimized for signals with a bandwidth which is $\frac{1}{OSR}$ times smaller than the maximal bandwidth permitted by the time-step δt, i.e., $f_m = \frac{1}{2 \cdot \delta t}$ according to the Nyquist theorem [20]. In practice, a low-pass filter is then generally applied on the input signal to eliminate frequencies greater than $f_b = \frac{f_m}{OSR}$. In our experiments, we used a low-pass filter H[2] that we applied to random signals with a maximum amplitude of $u_{\max}$, i.e., for all k, $u(k) = H(w(k))$ where $w(k)$ is chosen in $[-u_{\max}, u_{\max}]$ with a uniform random distribution. The resulting stochastic input generator randomly picks a signal in the set of all signals intended to be used with the model we consider.

6.2 Saturation analysis

We consider the formula $\Pr_{\geq\theta}(\Diamond \mathit{Satur})$, i.e., whether saturation occurs with a probability greater or equal to θ for different values of $u_{\max}$. We first used BI-SPRT to estimate the probability of $\Diamond \mathit{Satur}$ for the third order modulator that is analysed in [7]. In [7], reachability techniques are used to guarantee that for *every* input signal in a given range, the integrator state never saturates. While this approach is clearly sound for proving stability, its computational cost is prohibitive. As an example, in [7], the absence of saturation is proved for a small number of steps (for instance $N = 31$). By providing probabilistic guarantees instead of exhaustiveness, our approach makes it possible to consider much larger horizons ($N = 24000$ in the following experiments). To remain in the same experimental setting as in [7], we do not use a low-pass filter. We set the two error bounds α and β to 0.001 and use an indifference region of size $(p_1, p_0) = (\theta - 0.01, \theta + 0.01)$. The results we obtained are reported in Table 1. The first and fourth columns report the value of $u_{\max}$ and the value of p found, respectively. Column 3 reports the number of simulations performed. The results confirm the fact, proved in [7], that for signals with a maximum amplitude of 0.1, i.e., $u_{\max} = 0.1$, the circuit never saturates whereas if $u_{\max}$ is more than 0.3, the circuit always does.

In Table 2, we show the results we obtained for modulators of orders 3 ($\mathcal{S}_3$), 5 ($\mathcal{S}_5$), and 7 ($\mathcal{S}_7$), respectively. In those experiments, we use the low-pass filter for the input signal. This explain why higher values of $u_{\max}$ can be used without provoking saturation. From these experiments, which cannot be handled with the technique of [7], we can observe that higher order modulators are more likely to saturate.

[2]The description of low-pass filters is outside the scope of this paper. We used a 5[th] order Butterworth filter provided by the Signal Processing Toolbox of Matlab.

Table 2 Estimation of the probability to saturate for $\mathcal{S}_3$, $\mathcal{S}_5$ and $\mathcal{S}_7$, with BI-SPRT algorithm and $\alpha = 0.01$, $\beta = 0.01$ and $\delta = 0.05$

$u_{\max}$	$\Pr(\mathcal{S}_3 \models \Diamond Satur)$	$\Pr(\mathcal{S}_5 \models \Diamond Satur)$	$\Pr(\mathcal{S}_7 \models \Diamond Satur)$
0.25	0	0	0
0.5	0	0	0.0625
0.75	0	0	0.125
1	0	0	0.21875
1.25	0.03125	0.15625	0.4375
1.5	0.21875	0.5	0.75
1.75	0.59375	0.8125	0.9375
2	0.875	1	1
2.25	1	1	1
2.5	1	1	1

6.3 Frequency domain analysis

In addition to improving the computation time, our approach makes it possible to verify more complex properties than those that can be handled with a reachability-based technique. In particular, by defining execution predicates involving the Fourier transform, we can check reliably whether an analog signal was properly converted to a digital one.

The quality of the conversion is usually measured in terms of the signal-to-noise ratio (SNR), i.e., the comparison between the power of the original analog signal (denoted w_s) and the power of the quantization noise (denoted w_n), on the bandwidth of interest $[0,\ f_b]$. Let σ be an execution and $\hat{\xi}^u$ and $\hat{\xi}^v$ be the Fourier transforms of the input analog signal u and the corresponding digital signal v associated with σ. The quantities $w_s(\sigma)$ and $w_n(\sigma)$ are given by:

$$w_s(\sigma) = \left(\sum_{k \le N, \nu_k \le f_b} |\hat{\xi}^u[\nu_k]|^2 \right)^{\frac{1}{2}} \quad \text{and} \quad w_n(\sigma) = \left(\sum_{k \le N, \nu_k \le f_b} |\hat{\xi}^u[\nu_k] - \hat{\xi}^v[\nu_k]|^2 \right)^{\frac{1}{2}},$$

and the signal-to-noise ratio of the execution σ, traditionally given in decibels, is:

$$\mathrm{snr}(\sigma) = 20 \log_{10} \frac{w_s(\sigma)}{w_n(\sigma)}.$$

The higher the signal-to-noise ratio, the better the quality of the conversion. In Table 3, we compute the average value of the SNR over 100 executions for filtered, random input signals generated as above with different values of $u_{\max}$, and for three modulators of different orders.

These results seem to show that the 5$^{\text{th}}$ order modulator performs better than the 3$^{\text{rd}}$ and the 7$^{\text{th}}$ order ones, except for input signals of higher amplitudes for which only the 3$^{\text{rd}}$ order modulator seems to be able to achieve acceptable performance. However, the numbers in this table correspond to a fixed number of simulations (they do not result from an application of our methodology) and thus cannot be used to estimate the actual probabilities that the circuits behave correctly for inputs with given amplitudes. To estimate these probabilities, we can apply our approach using an execution predicate p_{snr} involving the signal-to-noise ratio of an execution. If we assume that a conversion for an execution σ is correct if $\mathrm{snr}(\sigma)$ is more than 30, then the predicate is defined as

$$p_{\mathrm{snr}}(\sigma) = \mathbf{T} \quad \text{iff } \mathrm{snr}(\sigma) \ge 30.$$

Table 3 Comparison of average signal-to-noise ratio for Δ–Σ of different orders computed on 100 executions

$u_{\max}$	SNR, 3rd order Δ–Σ	SNR, 5th order Δ–Σ	SNR, 7th order Δ–Σ
0.5	39.948	46.4423	44.047
1	45.7603	51.9534	49.7312
1.5	49.0874	55.12	52.8598
2	51.4339	57.4009	55.3808
2.5	53.0783	59.0686	50.9091
3	54.5813	57.3029	29.832
3.5	53.7013	45.8932	−2.68854
4	49.6055	21.8596	−9.30988
4.5	41.5892	−3.11012	−9.14168
5	32.2871	−5.03031	−8.1362

Table 4 Probabilities that the Signal-to-noise predicate holds for a 3rd, a 5th and a 7th order Δ–Σ modulator Sys_3, Sys_5 and Sys_7, estimated with the BI-SPRT algorithm with $\alpha = 0.01$, $\beta = 0.01$ and $\delta = 0.05$

$u_{\max}$	$\Pr(\mathcal{S}_3 \models p_{\text{snr}})$	$\Pr(\mathcal{S}_5 \models p_{\text{snr}})$	$\Pr(\mathcal{S}_7 \models p_{\text{snr}})$
2	1	1	1
2.5	1	1	0.9375
3	1	0.96875	0.625
3.5	1	0.78125	0.125
4	1	0.40625	0
4.5	0.84375	0.0625	0
5	0.59375	0	0

This predicate can efficiently differentiate executions for which the conversion was acceptable (as in Fig. 2) from executions for which it is not (as in Fig. 3). It is easy to derive a MATLAB routine that can decide whether or not an execution of a Δ–Σ modulator satisfies p_{snr}. In Table 4, we report the results obtained when we apply the BI-SPRT algorithm to estimate the probability that the executions of our three modulators satisfy the predicate p_{snr}.

6.4 Mixed-domains (time–frequency) analysis

In the previous experiments, we can observe that for some values of $u_{\max}$ (e.g. $u_{\max} = 2$), both the formula $\Diamond Satur$ and the predicate p_{snr} hold with an estimated probability 1. This means that for *all* signals with a maximum amplitude of, e.g., $u_{\max} = 2$, the circuit saturates *and* provides a correct conversion. Thus, there cannot be a causality link between saturation in one state and improper conversion. In [7], it is assumed that the absence of saturation is necessary for p_{snr} to be true. Our experiments show that this may be an overly conservative assumption.

In this section, we are interested in investigating whether saturation during y consecutive states, where $y > 1$ can cause wrong behaviors. For this we defined a y-sequence predicate p_y such that p_y is true for a sequence σ_y of y states if and only if for each $0 \leq i < y$, $\sigma(i) \models Satur$. We evaluated the probabilities $\Pr(\mathcal{S}_5 \models \Diamond p_y)$, $\Pr(\mathcal{S}_5 \models p_{\text{snr}})$ and $\Pr(\mathcal{S} \models (\Diamond p_y \rightarrow \neg p_{\text{snr}}))$ for $y = 10$ and $y = 100$ and different values of $u_{\max}$. The results are reported in Table 5.

Table 5 Mixed-domains analysis for a 5th order Δ–Σ modulator ($\alpha = 0.0001$, $\beta = 0.0001$ and $\delta = 0.05$)

$u_{\max}$	$\Pr(\mathcal{S}_5 \models p_{10})$	$\Pr(\mathcal{S}_5 \models p_{100})$	$\Pr(\mathcal{S}_5 \models p_{\text{snr}})$	$\Pr(\Diamond p_{10} \rightarrow \neg p_{\text{snr}})$	$\Pr(\Diamond p_{100} \rightarrow \neg p_{\text{snr}})$
3	0.15625	0	0.96875	0.84375	1
3.5	0.625	0.09375	0.84375	0.46875	0.96875
4	0.9375	0.53125	0.5	0.5	0.90625
4.5	1	0.90625	0.15625	0.8125	0.90625
5	1	1	0.03125	0.96875	0.96875

We can observe that the probability of $(p_{100} \rightarrow \neg p_{\text{snr}})$ is always very high even when neither p_{100} nor $\neg p_{\text{snr}}$ are trivially satisfied. This shows that there is a correlation between saturation during 100 consecutive states and bad signal conversions.

7 Conclusion

This paper is the first attempt to apply the simulation-based techniques of Younes [26, 27, 29, 30] to verifying non-trivial properties of mixed-signal circuits. In comparison to the formal approach presented in [7], our technique makes it possible to obtain better performance results as well as to handle a larger class of properties. Our results are correct up to a pre-specified probability of error, while those of [7] are exact. Of particular interest is the possibility of specifying properties in the time domain as well as in the frequency domain. We also introduce mixed-domains properties, i.e., properties that apply both for the timed signal and for its Fourier transform.

Our work requires the ability to monitor properties of discrete-time signals, which can easily be done with existing techniques [8, 12]. In a series of recent papers [14, 16], Nickovic et al. proposed techniques for monitoring properties of *dense-time* analog signals. An interesting direction would be to adapt our techniques to work in this latter, more demanding context.

We also intend to consider extensions of SLTL incorporating past temporal operators and a better correlation between execution predicates and temporal operators. We plan to define more complex specifications for frequency domain properties based on the needs of designers of mixed signal circuits. Our ultimate goal is to provide them with a general framework for specifying and verifying properties of mixed-signal circuits.

Acknowledgements We thank H. Younes for answering many questions on his work. We also thank Nir Piterman who provided us with interesting comments and suggestions.

References

1. Aziz PM, Sorensen HV, Van Der Spiegel J (1996) An overview of sigma-delta converters. IEEE Signal Process Mag, pp 61–84, January 1996
2. Baier C, Haverkort BR, Hermanns H, Katoen J-P (2003) Model-checking algorithms for continuous-time Markov chains. IEEE Trans Software Eng 29(6):524–541
3. Biere A, Heljanko K, Junttila TA, Latvala T, Schuppan V (2006) Linear encodings of bounded ltl model checking. Log Methods Comput Sci 2(5)
4. Ciesinski F, Baier C (2006) Liquor: A tool for qualitative and quantitative linear time analysis of reactive systems. In: QEST, IEEE, pp 131–132

5. Ciesinski F, Größer M (2004) On probabilistic computation tree logic. In: Validation of stochastic systems. Lecture notes in computer science, vol 2925. Springer, Berlin, pp 147–188
6. Courcoubetis C, Yannakakis M (1995) The complexity of probabilistic verification. J ACM 42(4):857–907
7. Dang T, Donze A, Maler O (2004) Verification of analog and mixed-signal circuits using hybrid systems techniques. In: Hu AJ, Martin AK (eds) FMCAD'04–Formal methods for computer aided design. Lecture notes in computer science, vol 3312. Springer, Berlin, pp 21–36
8. d'Amorim M, Roşu G (2005) Efficient monitoring of ω-languages. In: CAV. Lecture notes in computer science, vol 3576. Springer, Berlin, pp 364–378
9. Frigo M, Johnson SG (1997) The fastest Fourier transform in the west. Technical report MIT-LCS-TR-728, Massachusetts Institute of Technology, September 1997
10. Gupta S, Krogh BH, Rutenbar RA (2004) Towards formal verification of analog designs. In: ICCAD, pp 210–217
11. Kwiatkowska MZ, Norman G, Parker D (2004) Prism 2.0: A tool for probabilistic model checking. In: QEST, IEEE, pp 322–323
12. Bauer A, Leucker M, Schallhart C (2006) Monitoring of real-time properties. In: FSTTCS. Lecture notes in computer science, vol 4337. Springer, Berlin, pp 260–272
13. Matsumo M, Nishimura T (2000) Dynamic creation of pseudorandom number generators. In: Monte-Carlo and Quasi-Monte Carlo methods 1998. Springer, Berlin, pp 56–69
14. Maler O, Nickovic D, Pnueli A (2008) Checking temporal properties of discrete, timed and continuous behaviors. In: Pillars of computer science, pp 475–505
15. Medeiro F, Pérez-Verdú B, Rodríguez-Vázquez A (2001) Top-down design of high-performance sigma-delta modulators. Kluwer, Dordrecht. Chapter 2
16. Nickovic D, Maler O (2007) Amt: A property-based monitoring tool for analog systems. In: FORMATS, pp 304–319
17. Pnueli A (1977) The temporal logic of programs. In: Proc. 18th annual symposium on foundations of computer science (FOCS), pp 46–57
18. Schreier R (2003) The delta-sigma toolbox version 6.0, January 2003
19. Zakhor A, Hein S (1993) On the stability of sigma delta modulators. IEEE Trans Signal Process, 41, July 1993
20. Smith SW (1997) The scientist and engineer's guide to digital signal processing. California Technical Publishing, San Diego
21. Schreier R, Temes GC (2005) Understanding delta-sigma data converters. Wiley/IEEE Press, Hoboken
22. Sen K, Viswanathan M, Agha G (2004) Statistical model checking of black-box probabilistic systems. In: CAV. Lecture notes in computer science, vol 3114. Springer, Berlin, pp 202–215
23. Sen K, Viswanathan M, Agha G (2005) On statistical model checking of stochastic systems. In: CAV. Lecture notes in computer science, vol 3576, pp 266–280
24. Wald A (1945) Sequential tests of statistical hypotheses. Ann Math Stat 16(2):117–186
25. Younes HLS, Kwiatkowska MZ, Norman G, Parker D (2006) Numerical vs. statistical probabilistic model checking. STTT 8(3):216–228
26. Younes HLS (2005) Probabilistic verification for "black-box" systems. In: CAV. Lecture notes in computer science, vol 3576. Springer, Berlin, pp 253–265
27. Younes HLS (2005) Verification and planning for stochastic processes with asynchronous events. PhD thesis, Carnegie Mellon
28. Younes HLS (2005) Ymer: A statistical model checker. In: CAV. Lecture notes in computer science, vol 3576. Springer, Berlin, pp 429–433
29. Younes HLS (2006) Error control for probabilistic model checking. In: VMCAI. Lecture notes in computer science, vol 3855. Springer, Berlin, pp 142–156
30. Younes HLS, Simmons RG (2002) Probabilistic verification of discrete event systems using acceptance sampling. In: CAV. Lecture notes in computer science, vol 2404. Springer, Berlin, pp 223–235
31. Younes HLS, Simmons RG (2006) Statistical probabilistic model checking with a focus on time-bounded properties. Inf Comput 204(9):1368–1409