Centre Fédéré en Vérification

Technical Report number 2008.101

Statistical Model Checking of Mixed-Analog Circuits

Edmund Clarke, Alexandre DonzÃĺ, Axel Legay

This work was partially supported by a FRFC grant: 2.4530.02

http://www.ulb.ac.be/di/ssd/cfv

Statistical Model Checking of Mixed-Analog Circuits*

Version of June 20, 2008

Edmund Clarke, Alexandre Donzé, Axel Legay

School of Computer Science
Carnegie Mellon University, Pittsburgh, PA 15213
{emc|adonze|alegay@cs.cmu.edu}

Abstract. Model Checking properties for systems involving continuous state variables is known to be a difficult problem. This holds, in particular, for mixed-signal circuits, i.e., circuits for which there is an interaction between analog (continuous) and digital (discrete) quantities. In this paper, we consider the problem of *Statistical* Model Checking of mixed-signal circuits. Instead of verifying a property exhaustively with respect to the behaviors of the model, we evaluate it on a representative subset of behaviors, generated by simulation, and answer the question of whether the circuit satisfies the property with a probability greater than or equal to some value. The answer is correct up to a certain probability of error, which can be pre-specified. The method automatically determines the maximal number of simulations needed to achieve the desired accuracy, thus providing a convenient way to control the trade-off between precision and computational cost, even for complex systems. We provide a logic adapted to the specification of properties for mixed-signal circuits, in the temporal domain as well as in the frequency domain, which is highly relevant in this specific context. Finally, we demonstrate the applicability of the method on a model of a $\Delta - \Sigma$ modulator for which previous formal verification attempts were too conservative and required excessive computation time.

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. It is generally solved with a *numerical approach* that consists in 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. [BHHK03,CY95,CG04,BRV04]) and tools (see e.g. [KNP04,CB06])

* This research was sponsored by the Gigascale Systems Research Center (GSRC), the Semiconductor Research Corporation (SRC), the Office of Naval Research (ONR), the Naval Research Laboratory (NRL), the Army Research Office (ARO), and the General Motors Lab at CMU awards to E.M.C., and a B.A.E.F grant.

exist for various classes of systems, including (continuous time) Markov Chains and Markov Decision Processes. The drawback of these approaches is that they compute the probability considering all the executions of the system, which may not scale up for systems of large size. Another way to solve the probabilistic Model Checking problem is to use a *statistical approach* based on hypothesis testing and simulation (e.g., [You05,You06,YS02,YS06] or [SVA04,SVA05]). The key idea is to deduce whether or not the system satisfies the property by observing some of its executions. Of course, in contrast to a numerical approach, a test-based solution does not guarantee a correct result. However, it is possible to bound the probability of making an error. Statistical approaches are known to be far less memory and time intensive than numerical ones, and are sometimes the last resort [YKNP06]. In this paper, we focus on the statistical procedure proposed by Younes [You05,You06,YS02,YS06] that consists in using Wald's *sequential probability ratio test* [Wal45] to verify bounded-time properties of discrete-time event systems. Younes' approach has at least three advantages: (1) it automatically determines the number of simulations needed to achieve the desired accuracy, thus providing a convenient way to control the trade-off between precision and computational cost, even for complex systems, (2) it tries to minimize this number, and (3) it proposes a general framework that includes many probabilistic models such as (continuous-time) Markov Chains, and some models of stochastic hybrid systems.

In this work, we consider applying statistical techniques to verify properties of *mixed-signal circuits*, i.e., circuits for which there is an interaction between analog (continuous) and digital (discrete) quantities. Our first contribution is to propose a version of stochastic discrete-time event systems that fits into the framework of [You05,You06,YS02,YS06] with the additional advantage that it explicitly handles analog and digital signal. We also introduce *probabilistic signal linear temporal logic*, a logic adapted to the specification of properties for mixed-signal circuits in the *temporal domain* as well as in the *frequency domain*, which is highly relevant in this specific context. Our second contribution is the analysis of a $\Delta - \Sigma$ modulator. A $\Delta - \Sigma$ modulator is an efficient *Analog-to-Digital Converter circuit*. It takes an analog value $u \in \mathbb{R}$ as input and encodes it into a digital value $v \in \mathcal{D}$. The verification of this circuit is made difficult by its hybrid (analog-digital) nature. A critical issue in this analysis is the stability of the modulator, i.e., the possibility that its internal state variables keep increasing until reaching a maximum value, i.e., a *saturation state*. Saturation is commonly assumed to compromise the quality of the analog-to-digital conversion.

In [DDM04] and [GKR04], techniques developed for general hybrid systems were used to analyze the stability of a third-order modulator. The idea is to apply exhaustive reachability 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 strictly precise, it has important drawbacks such as (1) signals with large ranges cannot be practically analyzed and (2) there are interesting

properties that cannot be expressed as reachability properties. As an example, It is unclear whether saturation immediately induces an improper signal conversion or if the system can bear short saturation periods without a major decrease in quality of the result. The latter cannot be checked with a reachability-based approach. Here, we propose to analyze a stochastic version of a $\Delta - \Sigma$ modulator. This allows us to handle more complex temporal properties and signals with larger ranges. As an example, in our experiments, we have been able to analyze discrete signals with more than 24000 sampling points in seconds, while the approach in [DDM04] was limited to 31 points in hours. We were also able to answer the open question in [DDM04] and showed that it is unlikely for short saturation periods to imply an improper signal conversion. These results are correct up to a certain probability, which can be chosen arbitrarily high without a significant increase of the computation time.

The rest of the paper is organized as follows. Section 2 briefly recalls the essentials of statistical probabilistic Model Checking. Section 3 presents the model and the logic used. Section 4 recalls some facts about Fourier analysis and introduces frequency-domain predicates that fits into the framework of Section 3. Section 5 describes $\Delta - \Sigma$ modulators, and Section 6 describes the experiments that we performed on a third-order $\Delta - \Sigma$ modulator. We conclude with some directions for future work.

2 Statistical Probabilistic Model Checking

The following section introduces the concept of *Statistical Probabilistic Model Checking*. We assume the reader to be familiar with basic concepts in probability theory.

2.1 The Problem

We use $Pr(E)$ to denote the probability of event E. We consider a system $\mathcal{S}$ whose executions are *observable* and a property ϕ. We assume that one can decide whether an execution of $\mathcal{S}$, denoted by σ, satisfies ϕ. The *Probabilistic Model Checking Problem* consists in deciding whether the executions of $\mathcal{S}$ satisfy ϕ with a probability greater than or equal to a given threshold θ. The latter is denoted by $\mathcal{S} \models Pr_{\geq\theta}(\phi)$. This statement only makes sense if one can define a probability space on the executions of the system as well as on the set of executions that do satisfy ϕ. In the rest of this section, we assume that such a probability space can always be defined, and we present a statistical procedure to solve this problem.

2.2 Statistical Approach [You05,You06]

A statistical approach to the Probabilistic Model Checking Problem 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 explore the set of all the reachable states of the system. Formally, let B_i be a discrete random variable with a Bernoulli distribution. Such a variable can only take 2 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 determine whether $\mathcal{S}$ satisfies ϕ with a probability $p \geq \theta$, we can test the hypothesis $H : p \geq \theta$ against $K : p < \theta$. A test-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 [You05] for details). A solution to this problem is to relax the test by working with an *indifference 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 both the values of p and θ are between p_1 and p_0, 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.

2.3 Algorithmic Schemes

Fixed size samples We now need to provide a test procedure that satisfies the requirements above. We recall two solutions proposed by Younes in [You05,YS06].The first solution is to decide acceptance with *fixed-size samples*. The idea is as follows. A *sample* of size n consists of n observations, $b_1, \ldots, b_n$ of the Bernoulli variables $B_1, \ldots, B_n$ that represent our experiment. To test hypothesis H_0 against hypothesis H_1, we specify a constant c. If $\sum_{i=1}^{n} b_i$ is greater 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. Clearly, this number has to be greater if α and β are smaller but also if the size of the indifference region is smaller. This results in an optimization problem, which generally does not have a closed-form solution except for a few special cases (see [You05] for a discussion). In his thesis [You05], Younes proposed a binary search based algorithm (Algorithm 2.1 page 21) SINGLE SAMPLING-PLAN that, given p_0, p_1, α, β, computes an approximation of the minimal value for n.

Sequential sampling test The sample size for a single sampling plan is fixed in advance and independent of the observations that have been 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. It thus seems to be better to test H_0 against H_1 after each experiment. This is the purpose of a *sequential acceptance sampling test* procedure.

We recall the *sequential probability ratio test* proposed by Wald [Wal45]. We first 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 $\frac{p_{1m}}{p_{0m}} \leq A$, and H_1 if $\frac{p_{0m}}{p_{1m}} \leq B$. An algorithm for sequential ratio testing consists in computing $\frac{p_{1m}}{p_{0m}}$ for successive values of m until either H_0 or H_1 is satisfied. 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 [You05], Younes proposed a logarithmic based algorithm (Algorithm 2.3 page 27) SPRT that given p_0, p_1, α, β implements the sequential ratio testing procedure. In practice, choosing A and B such that the strength (α, β) is respected is non-trivial. In his seminal paper [Wal45], Wald showed that if one defines $A = \frac{(1-\beta)}{\alpha}$ and $B = \frac{\beta}{(1-\alpha)}$, then we obtain a new test whose strength is (α', β'), but such that $\alpha' + \beta' \leq \alpha + \beta$ (meaning that at least either $\alpha' \leq \alpha$ or $\beta' \leq \beta$).

Boolean Combination of Properties The SPRT algorithm can be extended to handle Boolean combinations of probabilistic properties. Let $\neg$, $\vee$, and $\wedge$ be the normal *logic operators*, which are read "not", "or", and "and", respectively. Consider the case in which ψ is a Boolean combination of properties of the form $Pr_{\geq\theta}(\phi)$ (note that nested probabilistic operators are not allowed and recall that ϕ is defined as a set of executions). We have the two usual cases:

1. Assume that $\psi' = \neg\psi$, where ψ is a property of the form $Pr_{\geq\theta}(\phi)$, where ϕ is a set of executions. In [You05,YS06], it is proved that if ψ can be verified with a test of strength (α, β), then ψ' can be verified with a test of strength (β, α).
2. Assume that $\psi = \bigvee_{i=1}^{i=n} \psi_i$, where each ψ_i is a property of the form $Pr_{\geq\theta}(\phi)$ (where ϕ is a set of executions), which is verified with a test of strength (α_i, β_i). In [You05,YS06], it is proved that in such a situation, ψ can be verified with a strength of (α, β), where $\alpha = \sum_{i=1}^{n} \alpha_i$ and $\beta = \max_{1 \leq i \leq n} \beta_i$.

Statistical Model Checker The SPRT algorithm can be implemented in order to solve the probabilistic model checking problem for a specific class of systems and a specific class of properties. For this, we need:

- *A simulator* that is able to accurately and efficiently simulate the system and produce observable executions;
- *An execution verifier* that decides whether an execution satisfies a given property.

With these two components, we can implement the realization b_i of the B_i Bernoulli variable used by the SPRT algorithm. The simulator produces the i^{th} trace and the verifier decides whether it satisfies the property ($b_i = \mathbf{T}$) or not ($b_i = \mathbf{F}$).

3 System, Signals and Logics

3.1 Notations and Definitions

Let $\mathcal{T}$ be a finite set of non-negative reals $\{t_0, t_1, \ldots, t_{N-1}\}$, 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 consisting of 2^b elements, which can be encoded in terms of b bits. 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. 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* to $[t, t']$ of a signal ξ, denoted by $\xi_{|_{[t,t']}}$, is a signal such that:

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

Our main motivation is to verify properties of mixed-signal circuits. For this purpose, we propose *stochastic signal discrete-time event systems*, which are an extension of the classical model of stochastic discrete-time event systems with information about signals. During an execution, such systems have to remain in the same state between the occurence of two events. Thus the signals that we associate with an execution are restricted to be piecewise-constant.

Definition 1. *Let $\mathcal{B}$ be a finite set of Boolean propositions. A* stochastic signal discrete-time event system *(SSDES) is a tuple $\mathcal{S} = (\mathcal{T}, S, S_0, \to, \pi_a, \pi_d, L)$ where*

- *$\mathcal{T}$ is a finite set of non-negative reals $\{t_0, t_1, \ldots, t_{N-1}\}$, with $t_{i+1} - t_i = \delta t$;*
- *S is the set of states, defined as $S = A_s \times D_s$, where A_s is a subset of $\mathbb{R}^{n_a}$ and $D_s \subset \mathcal{D}^{n_d}$. We have that n_a and n_d are the number of analog and digital signals associated with $\mathcal{S}$, respectively. These signals will be denoted by $\xi_a^1, \ldots, \xi_a^{n_a}$ and $\xi_d^1, \ldots, \xi_d^{n_d}$, respectively;*
- *S_0 is the set of initial states;*

– *The relation* $\rightarrow$: $S \times S$ *is the transition relation of the system. We assume a probability distribution on* $\rightarrow$*, i.e.,*

$$\forall s \in S, \sum_{s' \in S} Pr(s \rightarrow s') = 1;$$

– 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;*
– $\pi_a : S \times \{1, \ldots, n_a\} \rightarrow A_s$ *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 similarly to* π_a*.*

Let $\omega = s_1 \ldots s_k$ be a finite sequence of states of $\mathcal{S}$. We use $\omega(i)$ and ω^i to denote the i-th state of ω and the sequence $s_i \ldots s_k$, respectively. The length ω, which is 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 $\sigma = s_0 s_1 \ldots s_{N-1}$, with $s_0 \in S_0$ and such that for each $i \in 0 \ldots N-1$, $s_i \in S$ and $s_i \rightarrow s_{i+1}$. Each state s_k (with $k < N$) of σ assigns to each analog signal ξ_a^i (respectively, digital signal ξ_d^i) its constant value between t_k and t_{k+1}, i.e., $\xi_a^i[t] = \pi_a(s_k, i)$ (respectively $\xi_d^i[t] = \pi_d(s_k, i)$) for $t \in [t_k, t_{k+1}]$. The i-th *suffix* of σ is the sequence $s_i, \ldots, s_{N-1}$. An SSDES is thus an infinite-state Markov chain extended with information and operations on analog and digital signals.

3.2 Probabilistic Signal Linear Temporal Logic

We introduce the *probabilistic signal linear temporal logic* (SLTL) to reason on the set of executions of an SSDES. In the rest of the section, we assume a set of atomic propositions $\mathcal{B}$ and an SSDES $\mathcal{S} = (\mathcal{T}, S, S_0, \rightarrow, \pi_a, \pi_d, L)$ with L being a mapping from the set of states S to $2^{\mathcal{B}}$. Before introducing SLTL, we first recall the syntax and the semantics for linear temporal logic (LTL). The syntax of LTL is given by the following grammar:

$$\phi ::= \mathbf{T} \,|\mathbf{F}\, |b \in \mathcal{B}\, |\neg b\, |\bigcirc \phi\, |\, \phi_1 \vee \phi_2\, |\, \phi_1 \wedge \phi_2\, |\, \phi_1 \mathcal{U} \phi_2\, |\, \phi_1 \widetilde{\mathcal{U}} \phi_2.$$

We now present the semantics of LTL, which here is defined with respect to finite sequences of states of $\mathcal{S}$. 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 $b \in \mathcal{B}$ iff $b \in L(\sigma(0))$;
– $\omega \not\models b$ with $b \in \mathcal{B}$ iff $b \notin L(\sigma(0))$;
– $\omega \models \bigcirc \phi$ 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 $\omega^j \models \phi_1$.

In the logic defined above, negation can only be applied to atomic propositions. Negation can be extended to any formula with the help of the following relations:

$$\omega \not\models \phi_1 \mathcal{U} \phi_2 \text{ iff } \omega \models (\neg\phi_1)\widetilde{\mathcal{U}}(\neg\phi_2);$$
$$\omega \not\models \phi_1 \widetilde{\mathcal{U}} \phi_2 \text{ iff } \omega \models (\neg\phi_1)\mathcal{U}(\neg\phi_2);$$
$$\omega \not\models \bigcirc\phi_1 \text{ iff } \omega \models \bigcirc\neg\phi_1.$$

Two additional temporal operators are used. The first is $\diamondsuit$, where $\diamondsuit$ is read "eventually". The eventually operator requires that its argument eventually becomes true. Formally, we have $\diamondsuit\psi = \mathbf{T}\mathcal{U}\psi$. The second operator is $\square$, where $\square$ is read "always". This operator requires that its argument stays true. Formally, we have $\square\psi = \mathbf{F}\widetilde{\mathcal{U}}\psi$.
We can now focus on the syntax and semantics of SLTL. We use the following definition.

Definition 2 (Execution Predicate). *Let $\Sigma(\mathcal{S})$ be the set of all the executions of an SSDES $\mathcal{S}$. An* execution predicate *p for $\mathcal{S}$ is a mapping from $\Sigma(\mathcal{S})$ to $\mathcal{B}$:*

$$p : \sigma \in \Sigma(S) \mapsto p(\sigma) \in \{\mathbf{T}, \mathbf{F}\}$$

This general definition of execution predicates makes it possible to define properties that apply to complete traces rather than to single states, as is usually the case with LTL formulas. This is useful, in particular, when we need to apply a functional to one or several signals associated with an execution and characterize the properties of the result. Such a functional can be the Fourier transform, e.g., as is described in Section 4. We can now give the following definition.

Definition 3 (SLTL Formula). *An* SLTL formula *is a Boolean combination of formulas of the form $\psi = Pr_{\geq\theta}(\phi)$, where ϕ is either a linear temporal logic formula or a Boolean combination of execution predicates for $\mathcal{S}$.*

We say that $\mathcal{S}$ satisfies ψ, which is denoted by $\mathcal{S} \models \psi$, if and only if the probability for an execution of $\mathcal{S}$ to satisfy ϕ is greater than θ. Such a semantics makes sense since one can always associate a probability with the set of executions that satisfy an LTL formula or a Boolean Combination of Boolean predicates.

Theorem 1. *Let be $\mathcal{S}$ be an SSDES and ϕ be either LTL formula or a Boolean combination of exectution predicates. One can always associate a probability with the set of executions of $\mathcal{S}$ that satisfy ϕ.*

Proof. Recall that an SSDES is an infinite-state Markov chain whose executions can be viewed as infinite executions by considering their last state to be an absorbing state, i.e., a state where the system stays forever. It is known that one can assign a probability distribution to sets of infinite executions of a Markov chain using a *probability space* and the classical notion of *basic cylinder* (see [CG04] for a survey). It is also known that this probability distribution is sufficient to assign a probability to the set of executions that satisfy an LTL formula [Var85]. We are now left with the case where ϕ is a Boolean combination of execution predicates. In such situation, we can derive from $\mathcal{S}$ another SSDES $\mathcal{S}'$ that represents the

tree expansion of all possible executions of $\mathcal{S}$, and where each state keeps an ordered list of its set of predecessors in the execution. Clearly, there is a one-to-one correspondence between the executions of $\mathcal{S}$ and $\mathcal{S}'$. We can now associate a Boolean variable with each terminal state s of $\mathcal{S}'$. This Boolean variable is true if the execution, which is recorded in s, satisfies the Boolean combination of execution predicates. It is false otherwise. We have thus reduced the problem of assigning a probability to the set of executions of $\mathcal{S}$ that satisfy a Boolean combination of execution predicates to the one of assigning a probability to the set of executions of $\mathcal{S}'$ that satisfy the Boolean variable in their last state, which is an LTL formula. Since $\mathcal{S}$ and $\mathcal{S}'$ have the same executions, the probabilities obtained from $\mathcal{S}'$ can be used for $\mathcal{S}$.

3.3 Statistical Model Checking for SLTL

We observe that SSDES and SLTL are in the scope of the class of systems and logics that can be handled with the SPRT algorithm introduced in Section 2. Indeed, the simulation of Markov chains, of which an SSDES is an instance, is common, as well as verifying LTL properties for finite executions (e.g., using off-line monitoring). Assuming also that we use only execution predicates that we can compute, an SLTL formula can be then verified for a given execution. One can thus decide whether an SSDES satisfies an SLTL formula using a statistical Model Checking approach.

4 Frequency-Domain Properties of Signals

4.1 Fourier Transform and Frequency Domain Signals

The *Fourier transform* is a widely used mathematical tool in signal processing [Smi97]. It transforms a *time-domain signal* $\xi : \mathcal{T} \to \mathbb{R}$ into a *frequency-domain* signal $F(\xi) : \mathcal{F} \to \mathbb{C}$ where the *frequency set* $\mathcal{F}$ is a subset of $\mathbb{R}$. For all ν in $\mathcal{F}$,

$$F(\xi)[\nu] = \int_{\mathcal{T}} \xi[t] e^{-\mathbf{i}2\pi\nu t} dt. \tag{2}$$

The *inverse Fourier transform* makes it possible to "reconstruct" the function ξ from $F(\xi)$. If we let $\hat{\xi} = F(\xi)$, we have:

$$\xi[t] = F^{-1}(\hat{\xi}) = \int_{\mathcal{F}} \hat{\xi}[\nu] e^{\mathbf{i}2\pi\nu t} d\nu. \tag{3}$$

There are many operations that are easier to perform in the Fourier domain than in the time domain, e.g., convolution and differentiation. For these operations, it is then 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 which are by nature easier to analyze in the frequency domain than in the time

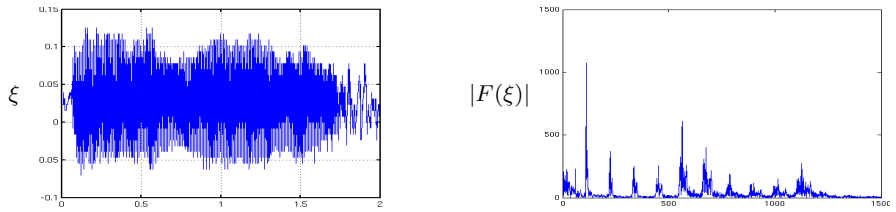

Fig. 1. Example of a Fourier transform. The signal was obtained by recording a human voice. Its Fourier transform lies in the interval $[0Hz, 1500Hz]$ (its value is 0 outside this interval).

domain (e.g., the sound of a voice as shown in Figure 1).

In our context, a signal ξ is sampled on the finite set of time points $\mathcal{T} = \{t_0, t_1, \ldots, t_{N-1}\}$, with $t_{i+1} - t_i = \delta t$. Assuming that ξ takes the zero value outside the interval $[t_0, t_{N-1}]$ and since it takes constant values on each interval $[t_i, t_{i+1}]$, $F(\xi)$ can be approximated by:

$$F(\xi)[\nu] = \delta t \sum_{k=0}^{N-1} \xi[t_k] e^{-\mathbf{i}2\pi\nu t_k}. \tag{4}$$

An efficient algorithm, known as the *Fast Fourier Transform (FFT) algorithm*, can compute this quantity. Given the N values of the signal ξ on $\mathcal{T}$, the algorithm returns the Fourier transform $F(\xi)$ on a set $\mathcal{F} = \{\nu_0, \ldots, \nu_{N-1}\}$, where $\nu_{i+1} - \nu_i = \delta\nu > 0$ with a computational complexity of $\mathcal{O}(N \log N)$ (see, e.g., [FJ97]).

4.2 Frequency-Domain Predicates

In the previous section, we defined an SLTL formula to be a Boolean combination of terms of the form $Pr_{\geq\theta}(\phi)$ where ϕ is either an LTL formula or a Boolean combination of execution predicates. The latter provides a means to specify and to verify a wide range of properties involving the values of the signals induced by σ, including properties in the frequency domain. To explicitly deal with Fourier transforms of signals in σ, we extend the Fourier Transform functional as follows. Let

$$F_a^i(\sigma) = F(\pi_a(\sigma, i)) \text{ (resp. } F_d^i(\sigma) = F(\pi_d(\sigma, i))),$$

for each $i \in \{1, \ldots, n_a\}$ (respectively for each $i \in \{1, \ldots, n_a\}$). Each $F_a^i(\sigma)$ (or $F_d^i(\sigma)$) is a frequency-domain signal on which standard operations (such as the sum, the product or the maximum) and comparison operators can be applied to define a Boolean predicate.

5 A Class of Mixed-Signal Circuits: $\Delta - \Sigma$ Modulators

This section briefly recalls the principles of $\Delta - \Sigma$ *modulation* and the related design issues. The reader can consult [MPVRV01] for more details on this advanced topic in Signal Processing.

A $\Delta - \Sigma$ 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 everywhere, which motivates their study.

5.1 Analog to Digital Conversion via $\Delta - \Sigma$ Modulation

The challenge of Analog-to-Digital conversion is to represent the uncountable set of analog values using the finite set of digital values $\mathcal{D}$. The direct approach, which is called *quantization*, consists in assigning each value in $\mathcal{D}$ to an interval in $\mathbb{R}$. The *quantization error* is given by $\delta = v - u$

Example 1. Assume that $\mathcal{D}$ is encoded with 2 bits and contains the values $\{-3/4, -1/4, 1/4, 3/4\}$. The relation between the quantized value v and the analog input u can be given by:

$$v = \begin{cases} -3/4 & \text{if } u \in [-\infty, -1/2] \\ -1/4 & \text{if } u \in [-1/2, 0] \\ 1/4 & \text{if } u \in [0, 1/2] \\ 3/4 & \text{if } u \in [1/2, +\infty] \end{cases}$$

In this example, if u is inside the interval $[-1, 1]$, then the value of the quantization error δ never exceeds $1/4$.

One way to decrease the quantization error is to increase the number of bits used to encode $\mathcal{D}$ and thus the number of possible digital values. This directly reduces the error in the time domain. Another approach, which is the one implemented in $\Delta - \Sigma$ modulation, is to process the quantization error through a feedback loop. Basically, a $\Delta - \Sigma$ modulator sums the quantization errors as time evolves and subtracts the result from the input[1]. Intuitively, this process reduces the mean of the quantization error but the benefit of the approach is more 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 the quantization. The feedback loop in the $\Delta - \Sigma$ 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. An illustration of this principle is given in Fig.2. The plots show the Fourier transform of the digital output of a $\Delta - \Sigma$ modulator for the signal of Fig. 1. The quantization error is pushed toward frequencies higher

[1] Hence the Σ for the summation of errors and the Δ for the difference in the feedback loop.

than $1500Hz$ where it can be safely removed since in this case, the original signal contains no information in this range. Note that $\Delta - \Sigma$ modulators can achieve good performance using a limited number of bits (only one in the case

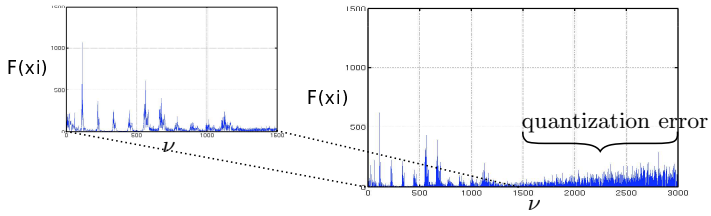

Fig. 2. Example of the behavior of the $\Delta - \Sigma$ modulator for the signal of Fig.1. The Fourier transform of the output signal (b) matches the Fourier transform of the input signal (a) on the interval $[0, 1500Hz]$. The quantization error is pushed toward frequencies higher than $1500Hz$.

5.2 Verification Issues

The design of a $\Delta - \Sigma$ modulator involves the implementation and the tuning of its feedback loop[2]. In particular, it contains *integrators* that are used to store the feedback quantization errors. The *order* of a modulator is given by the number of integrators it uses. Higher order modulators can exhibit better performance, but also introduce a *stability* issue. An integrator memorizes its input and adds it to the sum of all the previously read inputs during the execution. 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 a limited capacity, the values of these states would then reach a *saturation level*. Saturation can compromise the quality of the analog-to-digital conversion. The stability analysis of the feedback loop implemented by $\Delta - \Sigma$ modulators is made difficult by the nonlinearity (in this case, a discontinuity) induced by the quantizer. This invalidates the direct application of classical linear stability theory which makes the stability analysis of high order (greater or equal to three) $\Delta - \Sigma$ modulators a challenging problem.

In [DDM04] and [GKR04] reachability techniques developed in the area of hybrid systems were used to guarantee that for *every* input signal in a given range, the integrator state would never saturate. While this approach is clearly sound for proving stability, its computational cost is prohibitive. As an example,

[2] The details of this process is outside the scope of this paper since we are only interested in the verification of a given circuit.

6 Model and Experiments

In this section, we detail some experiments we conducted on a $\Delta - \Sigma$ Modulator. The model we consider is a discrete-time Simulink model of a third order $\Delta - \Sigma$ modulator (Fig. 4), the same as in [DDM04]. This model was obtained by using the standard MATLAB `delsig` toolbox[3] [Sch03].

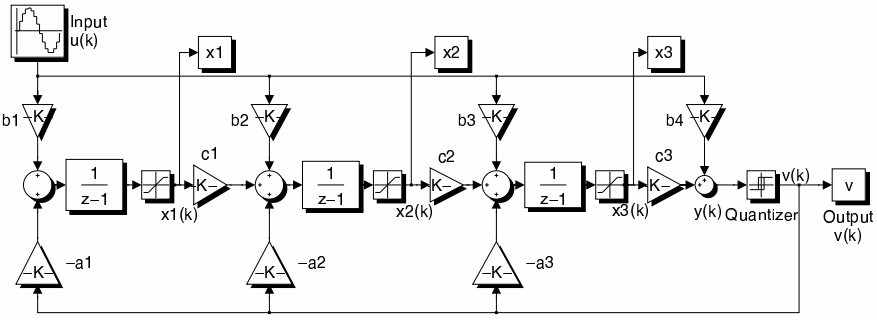

Fig. 4. Simulink model of a third order $\Delta - \Sigma$ modulator.The three blocks $\frac{1}{z-1}$ followed by saturation blocks represent the saturated integrators. The coefficient gains a_i, b_i and c_i were obtained using the `delsig` toolbox.

6.1 The SSDES Model

Our modulator can be represented with an SSDES $\mathcal{S} = (\mathcal{T}, S, S_0, \rightarrow, \pi_a, \pi_d, L)$ as follows.

- **Time.** We set $\mathcal{T} = \{t_0, t_1, .., t_{N-1}\}$ with $t_0 = 0.$, $t_{N-1} = 3$ and $\delta t = t_{i+1} - t_i = \frac{1}{8000}$, thus $N = 24000$.

- **Set of States.** The Simulink model contains three integrator blocks which contain each one real-valued (or analog) variable. A state $s \in S$ can thus be described as a tuple (u, x_1, x_2, x_3, v), where
  - x_1, x_2 and x_3 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 $n_a = 4$ and the number of digital signals $n_d = 1$. The integrator blocks are saturated, meaning that their states cannot go beyond certain values. In practice, $x_i \in [-1, 1]$ for $i \in \{1, 2, 3\}$ and $-1, 1$ are the *saturation values*. Assuming also that $u \in [-u_{\max}, u_{\max}]$,

[3] This toolbox provides practical models used by actual $\Delta - \Sigma$ modulator designers.

we get $A_s = [-1,1]^3 \times [-u_{\max}, u_{\max}]$ and $D_s = \{-1,1\}$. We simplify the notation and, given an execution $\sigma = s_0 s_1 \dots s_{N-1}$, we use $u(k) = \pi_a(s_k, 1)$, $x_1(k) = \pi_a(s_k, 2)$, $x_2(k) = \pi_a(s_k, 3)$, $x_3(k) = \pi_a(s_k, 1)$ and $v(k) = \pi_d(s_k, 1)$. For all $k \in 0 \dots N-1$, we have $\xi^u[t_k] = u(k)$ and $\xi^v[t_k] = v(k)$;

- **Transition relation.** When $u(k)$ is given, the model deterministically computes $x_1(k+1)$, $x_2(k+1)$, $x_3(k+1)$ and $v(k+1)$. Thus the probability distribution $Pr(s_k \rightarrow s_{k+1})$ for all $(s_k, s_{k+1}) \in S \times S$ is induced by the probability distribution of the input value $u(k+1)$. For our experiments, we consider uniform random inputs: for all k, $u(k)$ is chosen in a set $[-u_{\max}, u_{\max}]$ with a uniform random distribution;

- **Set of initial states.** Initially, the value of the integrator states are 0 and the digital output $v(0)$ is set to 1. The input value $u(0)$ is chosen randomly with a uniform distribution in $[-u_{\max}, u_{\max}]$ as specified above;

- **Boolean variables.** We define a boolean variable Sat which is true iff one of the analog values, i.e., either the input or an integrator state, saturates. Formally, $L(s) = \mathbf{T}$ iff there exist i in $\{1, \dots, 4\}$ such that $\pi_a(s, i) = 1$ or -1, $L(s) = \mathbf{F}$ otherwise.

Following the discussion at the end of Section 5, we chose a uniform distribution for the input signals. Thus the only assumption that we have imposed on the nature of the input signals was their maximum amplitude $u_{\max}$. The properties that we verified in our experiments were tested against different values for this maximum amplitude. As is intuitively expected, the circuits performance decreases as this amplitude increases. We implemented the SPRT algorithm described in Section 2 in the MATLAB environment, as well as the procedure to verify SLTL properties.

6.2 Experiments on Saturation

We first checked the formula $Pr_{\geq\theta}(\Diamond Sat)$, i.e., we evaluated the probability that the system saturates for different values of $u_{\max}$. Note that this corresponds to the probabilistic version of the exhaustive verification that was performed in [DDM04] and was successful up to only $N = 30$. We used an indifference region of size .02, and $\alpha = \beta = .01$. For each experiment, we generated 100 simulations [4] to get a first coarse approximation of the probability. Then we picked a θ according to these results and applied the SPRT algorithm to test $H_0 : p \geq \theta$. The results are summarized in Table 2. In the first column, we report the value of $u_{\max}$. The second column is the number of executions satisfying $\Diamond Sat$ over 100 executions. Column 3 gives the value of θ tested to check $Pr_{\geq\theta}(\Diamond Sat)$ with the SPRT algorithm. Column 5 and 6 report the answer obtained with the algorithm and the number of executions to get the result. More executions means that θ was closer to the actual probability of $\Diamond Sat$.

[4] The number 100 was arbitrarily chosen.

$u_{\max}$	Number of **T** on 100 exec.	Probability θ checked	SPRT result	SPRT number of exec.	Computational Time (s).
0.0	0	0.	H_0 false	459	57
0.1	0	0.	H_0 false	459	58
0.2	1	0.	H_0 true	57	7
0.3	56	.5	H_0 true	827	102
0.4	100	.99	H_0 true	228	28
0.5	100	1.	H_0 true	463	59

Table 1. Table of results for $Pr_{\geq\theta}(\Diamond Sat)$.

These results can be compared with those reported in [DDM04]. In particular, we confirmed the fact that for signals with a maximum amplitude of .1, the circuit never saturates whereas if $u_{\max}$ is .5, the circuit always does. In our case, though, the length N of the executions considered was much larger ($N = 24000$ to be compared to $N = 30$ in [DDM04]).

6.3 Verifying a Frequency Domain Predicate

For the second set of experiments, we checked the formula $Pr_{\geq\theta}(p_F)$ where p_F is a frequency-domain execution predicate defined as follows. Let d_F be a metric on frequency-domain signals such that for two signals $\hat{\xi}_1$ and $\hat{\xi}_2$,

$$d_F(\hat{\xi}_1, \hat{\xi}_1) = \frac{1}{N} \sum_{0 \leq k \leq N-1} |\hat{\xi}_1[\nu_k] - \hat{\xi}_2[\nu_k]|. \tag{5}$$

The truth value for the execution predicate p_F for an execution σ is given by

$$p_F(\sigma) = \mathbf{T} \text{ iff } d_F(\hat{\xi}^u_{|_{[0,\nu]}}, \hat{\xi}^v_{|_{[0,\nu]}}) \leq \epsilon.$$

With $\nu = 100Hz$ and $\epsilon = .05$, the predicate efficiently discriminates between executions for which the digital output has a correct Fourier transform (as in Figure 2) against executions when this is not the case (as in Figure 3).

These experiments show that, in general, saturation does not imply a wrong behavior. That the absence of saturation is necessary for p_F to be true, which was assumed in [DDM04], thus seems to be an overly conservative assumption.

Finally, we performed some experiments about computational times with respect to the strength parameters α and β, and the size of the indifference region given by $p_0 - p_1$. They allow to draw two main observations: the number of computed trajectories increases logarithmically with respect to the decrease of α and β and linearly with respect to the decrease of $p_0 - p_1$ (see Table 3). This indicates that one can verify that $P(\mathcal{S} \models \phi) \geq \theta$ with a very low probability of error whereas it is more difficult to estimate precisely the actual value of p by narrowing the indifference region. These results experimentally confirm theoretical and practical results reported in [You05].

$u_{\max}$	Number of **T** on 100 exec.	Probability θ checked	SPRT result	SPRT number of exec.	Computational Time (s).
0.9	96	0.9	H_0 true	1280	169
0.92	85	0.8	H_0 true	4547	622
0.94	49	0.4	H_0 true	277	38
0.96	22	0.2	H_0 true	333	47
0.98	6	0.1	H_0 false	3818	542
1.	0	.0	H_0 true	24	4

Table 2. Table of results for $Pr_{\geq\theta}(p_F)$

Test strength $\alpha(=\beta)$	Number of executions	—— ——	Indifference region $p_0 - p_1$	Number of executions
$1e^{-2}$	347	——	.1	44
$1e^{-4}$	496	——	0.05	117
$1e^{-6}$	873	——	0.02	239
$1e^{-8}$	1322	——	.01	614
$1e^{-10}$	1449	——	0.005	1080

Table 3. Number of trajectories against α (β was set equal to α) and against the size of the indifference region $p_0 - p1$ for $u_{\max} = 0.94$ and the property $Pr_{\geq 0.4}(p_F)$

7 Future Work

This paper presents the first attempt to apply the general statistical techniques introduced in [You05,YS06] to verifying non-trivial properties of mixed-signal circuits. A direction for future work is to generalize our results to an unbounded version of SLTL. This would relax the hypothesis that the size of the executions we consider is bounded. Such an extension would require modification of Younes' procedure, which is clearly designed to work with bounded properties. Our work requires the ability to monitor properties of discrete-time signals, which can be done easily with existing techniques [LS06,dR]. In a series of recent papers [NM07,MNP08], Nickovic et al. proposed techniques for monitoring properties of *dense-time* analog signals. Another interesting direction would be to adapt Younes' procedure to work in this latter, more demanding context. Finally, it would also be of interest to consider extensions of SLTL. As an example, one could consider formulas of the form $Pr_{\geq\theta}(\phi)$, where ϕ is a LTL formula whose atomic propositions could either be Boolean propositions or execution predicates.

Acknowledgement

We thank H. Younes for answering many email questions on his work and C. J. Langmead for fruitful discussions and proof-reading drafts of the paper.

References

BHHK03. Christel Baier, Boudewijn R. Haverkort, Holger Hermanns, and Joost-Pieter Katoen. Model-checking algorithms for continuous-time markov chains. *IEEE Trans. Software Eng.*, 29(6):524–541, 2003.

BRV04. Doron Bustan, Sasha Rubin, and Moshe Vardi. Verifying omega-regular properties of markov chains. In *CAV*, volume 3114 of *Lecture Notes in Computer Science*, pages 189–201. Springer, 2004.

CB06. F. Ciesinski and C. Baier. Liquor: A tool for qualitative and quantitative linear time analysis of reactive systems. In *QEST*, pages 131–132. IEEE, 2006.

CG04. F. Ciesinski and M. Größer. On probabilistic computation tree logic. In *Validation of Stochastic Systems*, LNCS, 2925, pages 147–188. Springer, 2004.

CY95. Costas Courcoubetis and Mihalis Yannakakis. The complexity of probabilistic verification. *Journal of the ACM*, 42(4):857–907, 1995.

DDM04. T. Dang, A. Donze, and O. Maler. Verification of analog and mixed-signal circuits using hybrid systems techniques. In Alan J. Hu and Andrew K. Martin, editors, *FMCAD'04 - Formal Methods for Computer Aided Design*, LNCS 3312, pages 21–36. Springer-Verlag, 2004.

dR. Marcelo d'Amorim and Grigore Roşu. Efficient monitoring of ω-languages. In *CAV*, LNCS 3576, pages 364 – 378. Springer.

FJ97. Matteo Frigo and Steven G. Johnson. The fastest Fourier transform in the west. Technical Report MIT-LCS-TR-728, Massachusetts Institute of Technology, September 1997.

GKR04. Smriti Gupta, Bruce H. Krogh, and Rob A. Rutenbar. Towards formal verification of analog designs. In *ICCAD*, pages 210–217, 2004.

KNP04. M. Z. Kwiatkowska, G. Norman, and D. Parker. Prism 2.0: A tool for probabilistic model checking. In *QEST*, pages 322–323. IEEE, 2004.

LS06. Andreas Bauer 0002, Martin Leucker, and Christian Schallhart. Monitoring of real-time properties. In *FSTTCS*, volume 4337 of *LNCS 4337*, pages 260–272. Springer, 2006.

MNP08. Oded Maler, Dejan Nickovic, and Amir Pnueli. Checking temporal properties of discrete, timed and continuous behaviors. In *Pillars of Computer Science*, pages 475–505, 2008.

MPVRV01. Fernando Medeiro, Belen Pérez-Verdú, and Angel Rodríguez-Vázquez. *Top-Down Design of High-Performance Sigma-Delta Modulators*, chapter 2. Kluwer Academic Publishers, 2001.

NM07. Dejan Nickovic and Oded Maler. Amt: A property-based monitoring tool for analog systems. In *FORMATS*, pages 304–319, 2007.

Sch03. Richard Schreier. The delta-sigma toolbox version 6.0, January 2003.

Smi97. Steven W. Smith. *The scientist and engineer's guide to digital signal processing*. California Technical Publishing, San Diego, CA, USA, 1997.

SVA04. Koushik Sen, Mahesh Viswanathan, and Gul Agha. Statistical model checking of black-box probabilistic systems. In *CAV*, LNCS 3114, pages 202–215. Springer, 2004.

SVA05. Koushik Sen, Mahesh Viswanathan, and Gul Agha. On statistical model checking of stochastic systems. In *CAV*, LNCS 3576, pages 266–280, 2005.

Var85. Moshe Y. Vardi. Automatic verification of probabilistic concurrent finite-state programs. In *FOCS*, pages 327–338, 1985.

Wal45. A. Wald. sequential tests of statistical hypotheses. *Annals of Mathematical Statistics*, 16(2):117–186, 1945.

YKNP06. Håkan L. S. Younes, Marta Z. Kwiatkowska, Gethin Norman, and David Parker. Numerical vs. statistical probabilistic model checking. *STTT*, 8(3):216–228, 2006.

You05. Håkan L. S. Younes. *Verification and Planning for Stochastic Processes with Asynchronous Events*. PhD thesis, Carnegie Mellon, 2005.

You06. Håkan L. S. Younes. Error control for probabilistic model checking. In *VMCAI*, LNCS 3855, pages 142–156. springer-verlag, 2006.

YS02. Håkan L. S. Younes and Reid G. Simmons. Probabilistic verification of discrete event systems using acceptance sampling. In *CAV*, LNCS 2404, pages 223–235. Springer, 2002.

YS06. Håkan L. S. Younes and Reid G. Simmons. Statistical probabilistic model checking with a focus on time-bounded properties. *Information and Computation*, 204(9):1368–1409, 2006.