TEST GENERATION FOR ANALOG AND MIXED-SIGNAL CIRCUITS USING HYBRID SYSTEM MODELS

Tarik NAHHAL[1] and Thao Dang[2]

[1]Department of Mathematics and Computer Science, Hassan II University, Casablanca, Morocco t.nahhal@fsac.ac.ma

[2]VERIMAG, 2 avenue de Vignate 32000 Gières, France
Thao.Dang@imag.fr

ABSTRACT

In this paper we propose an approach for testing time-domain properties of analog and mixed-signal circuits. The approach is based on an adaptation of a recently developed test generation technique for hybrid systems and a new concept of coverage for such systems. The approach is illustrated by its application to some benchmark circuits.

KEYWORDS

Hybrid System, Formal Methods in Conformance Testing, Analog and Mixed-Signal Circuit.

1. INTRODUCTION

The increasing need for analog and mixed-signal circuits has motivated the development in design and test tools for these circuits[27]. Analog and mixed signal testing is considered to be a very difficult task. Even when the area of the analog part in a mixed-signal circuit is small, the cost of its testing covers a significant proportion of the global manufacturing cost.

In comparison with the digital counterpart, the specific difficulties of analog and mixed-signal testing are the following. While digital testing can use simple fault models (such as stuck-at-faults), fault models in analog designs are often complex and reflect the process-related disturbances, such as parameter deviations or size changes which have an infinite domain of possible values. In addition, the performance measures (such as transfert curves, frequency characteristics) are more complex than patterns of 1's and 0's at the outputs in a digital circuit.

In this work, we focus on the problem of automatic test generation, which involves computing a set of input patterns that permit detecting a given fault (for example, by comparing the differences in the outputs with some predetermined tolerance thresholds). Test generation has been considered for analog circuits such as in [13], [25], [20], for mixed-signal circuits such as in [2], [15], [14], [29],[30], using a variety of techniques, such as static test generation [23],[28], sensitivity computation [13], Monte-Carlo simulation [25], [26], and optimization [5].

Following the model-based design approach, we propose a new test generation method for analog and mixed-signal circuits using hybrid system models. Hybrid systems are systems that combine

DOI : 10.5121/vlsic.2011.2302

discrete event systems and continuous systems and can naturally describe the behaviors of these circuits. Formal verification of these circuits using these models has been investigated in [7], [10]. Our test generation method is built upon on an adaptation of a recently developed test generation technique for hybrid systems, guided by a coverage measure [18]. The rest of the paper is organized as follows. First, we briefly introduce the model and important testing notions, such as conformance relation, test cases, test verdicts, coverage. We then formally state our testing problem and present our test generation algorithm. Finally, we demonstrate the application of the approach to three benchmark circuits.

2. TESTING PROBLEM

As a model for hybrid systems, we use hybrid automata [1]. In most classic versions of hybrid automata, continuous dynamics are defined using ordinary differential equations (ODEs). The behaviors of analog and mixed-signal circuits are however described using differential algebraic equations (DAEs). We thus adapt the model to capture this particularity. Mathematically, the behaviour of a (non-linear) analog circuit can be described by a set of DAEs:

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

Where $x(t) \in \mathfrak{R}^n$ denotes the state variables (internal voltages, currents, and outputs), $\dot{x}(t)$ denotes their time derivatives, $w \in W \subseteq \mathfrak{R}^m$ is the circuit parameter vector, and $u: \mathfrak{R}^+ \longrightarrow U \subseteq \mathfrak{R}^p$ is the input signal. When considering mixed-signal circuits that exhibit both logical and continuous, their behavior can be naturally modeled using hybrid automata. Before presenting the model, we remark that DAEs and ODEs are different in both theoretical and numerical properties. We shall briefly discuss the DAE solving method we use in Section IV.

2.1. Hybrid Automata

A hybrid automaton is an automaton augmented with continuous variables.

Definition 1 (Hybrid automaton). A hybrid automaton is a tuple *A= (X, Q, F, A, I, G,R)* where

- $X \subset \mathfrak{R}^n$ is the continuous state space. We denote by V (*A*) the set of continuous variables of *A*.
- *Q* is a finite set of locations (or discrete states).
- *E* is a set of discrete transitions.
- $F = \{F_q \mid q \in Q\}$ such that for each $q \in Q$, $F_q = (f_q, U_q, W_q)$ defines a DAE $f_q(x(t), \dot{x}(t), u(t), w) = 0.$
  where $w \in W_q \subseteq \mathfrak{R}^m$ is the parameter vector, and $u: \mathfrak{R}^+ \longrightarrow U_q \subseteq \mathfrak{R}^p$ is the input signal. We assume the existence and uniqueness of solutions of these differential algebraic equations. Note that during the evolution of the system, the parameter w is constant.
- $I=\{ I_q \subseteq \mathfrak{R}^n \mid q \in Q \}$ is a set of staying conditions.
- $G =\{ G_e \mid e \in E \}$ is a set of guards such that for each discrete transition $e \in$, $G_e \subseteq I_q$
- $R=\{ R_e \mid e \in E\}$ is a set of reset maps. For each $e=(q,q') \in E$, $R_e : G_q \longrightarrow 2^{Iq'}$ defines how x may change when A switches from q to q'.

A hybrid state is a pair (q, x) where $q \in X$ and the hybrid state space is $S=Q \times X$ state (q, x) can change in two ways: by continuous evolution and by discrete evolution. In location q the continuous evolution of x is governed by the DAE

$$f_q(x(t), \dot{x}(t), u(t), p) = 0.$$

Let $\phi(t, x, u(\cdot), p)$ be the solution of the DAE with the initial condition x, the parameter p and under the input $u(\cdot)$. A continuous transition: $(q,x) \xrightarrow{u(.),h} (q,x')$ where $h > 0$ is a positive real number means that $x' = \phi(h, x, u(\cdot), p)$ and for all $t \in [0,h[$ $\phi(t, x, u(\cdot), p) \in I_q$. In other words, x' is reached from x under the input $u(\cdot)$, and we say that $u(\cdot)$ is *admissible* starting at (q, x) for h time. For a state (q, x), if there exists a transition $e = (q, q') \in E$ and $x \in G_e$, then the transition e is enabled, the system can switch from location q to q' and the continuous variables can be assigned to a new value $x' \in R_e(x)$. This denoted by: $(q,x) \xrightarrow{e} (q',x')$ and we say that the discrete transition e is admissible at (q, x). We use the notation $(q,x) \rightarrow (q', x')$ to simply indicate that (q', x') is reachable from (q, x). We assume that discrete transitions are instantaneous. The hybrid automata we consider are assumed to be non-Zeno.

For simplicity of presentation, we first assume a single initial state of the automaton denoted by (q_{init}, x_{init}), and additionally all the paremeter set P_q are singletons. An extension of the framework to a set of initial states and sets of parameters will be discussed when we present our test generation algorithm.

Note that this model is non-deterministic. Indeed, in continuous dynamics the non-determinism is represented by the set of admissible input functions. The discrete transitions are non-deterministic because there might be continuous states at which the system can either continue with the same continuous dynamics or make a transition. Also, multiple transitions can be enabled at the same continuous states, and additionally, the reset maps could also be set-valued. This non-determinism is useful for describing disturbances from the environment and imprecision in modeling and implementation. To define our testing framework, we need the notions of inputs and observations.

Inputs

An input of the system which is controllable (by the tester) is called control input; otherwise, it is called disturbance input. We consider the following inputs:

- **Continuous inputs**. All the continuous inputs of the system are controllable. Since we want to implement the tester as a computer program, we are interested in piecewise-constant continuous input functions. Hence, a *continuous control action*, such as $(\bar{u}_q, h)$ where $\bar{u}_q$ is the value of the input and *h* is the time step, specifies that the system continues with the dynamics F_q under the input $u(t) = \bar{u}_q$ for exactly *h* time. We say that $(\bar{u}_q, h)$ is admissible at (q, x) if $u(t) = \bar{u}_q$ is admissible starting at (q, x) for h time.
- **Discrete inputs.** The discrete transitions are partitioned in controllable and uncontrollable discrete transitions. Those are controllable correspond to discrete control inputs, and the others to discrete disturbance inputs. The tester emits a discrete control action to specify whether the system should take a controllable transition (among the enabled ones) or continue with the same continuous dynamics. In the latter case, it can also control the values assigned to the continuous variables by the associated reset map.

We denote a discrete control action of this type by the corresponding transition, such as (q, q').

In the remainder of the paper, for simplicity of explanation, we use the following assumption: a continuous control action is of higher priority than any discrete input actions. This means that after a continuous control action ($\bar{u}_q$, h) is applied, no discrete transition can occur during h time, i.e. until the end of that continuous control action. This assumption is not restrictive, from a modeling point of view. As we shall see later, by considering all the possible values of h we can capture the cases where a discrete transition can occur before the termination of a continuous control action.

Definition 2 (Admissible Input Sequence). For a state (q, x), a sequence of input actions $i_0, i_1 \ldots i_k$ is admissible starting at (q, x) if the following conditions are satisfied:

- The first input action i_0 is admissible at (q, x).
- For each i = 1, 2, 3, . . . , k:

  - i_i is admissible at (q_i, x_i) where (q_i, x_i) is the state such that $(q_{i-1}, x_{i-1}) \xrightarrow{i_{i-1}} (q_i, x_i)$.

Intuitively, this means that an admissible control sequence does not cause the automaton A to be blocked.

Definition 3 (Trace). Given an admissible input sequence $\gamma = i_0, i_1 \ldots i_k$, the trace starting at (q, x) under γ is defined as follows:

$$\tau((q,x),\gamma) = \{(q_0,x_0),\ldots,(q_k,x_k) \mid (q_0,x_0) = (q,x) \wedge \forall i \in \{1,2,\ldots,k\} : (q_{i-1},x_{i-1}) \xrightarrow{\iota_{i-1}} (q_i,x_i) \}. \quad (1)$$

We use the notation $(q,x) \xrightarrow{\gamma} (q',x')$ to indicate that (q', x') is reachable from (q, x) after γ.

Admissible Control Sequences

It follows from the above assumption that uncontrollable discrete transitions cannot occur during a continuous control action. However, they can occur between control actions. Hence, the result of applying a control action is nondeterministic. Given a state (q, x) and a control action c, let $\sigma = \sigma_0, \sigma_1, \ldots \sigma_k$ be an input sequence that satisfies the following:

- $\sigma_0 = c$ and all the other σ_i with $i > 0$ are disturbance input actions.
- σ is an admissible input sequence starting at (q, x).

Let $\sum(c, (q, x))$ be the set of all sequences like σ. We can now define the set of traces reachable from (q, x) after a control action c as:

$$Tr((q,x),c) = \{\tau((q,x),\sigma) \mid \sigma \in \Sigma(c,(q,x))\}.$$

We also define the set of successors of (q, x) after a control action c as

$\text{Reach}((q, x), c) = \{(q', x') \mid \exists\, \sigma \in \sum(c, (q, x)) \ (q,x) \xrightarrow{\sigma} (q',x')\}$.

We can now proceed to define an *admissible control action sequence* (or admissible control sequence for short). Given a sequence of two control actions $\gamma = c_0, c_1$, we define

$$\Sigma(\gamma,(q,x)) = \{\sigma_0 \oplus \sigma_1 \mid \sigma_0 \in \Sigma(c_0,(q,x)) \wedge \exists (q',x') \in Reach((q,x),c_0) : \sigma_1 \in \Sigma(c_1,(q',x'))\},$$

where $\oplus$ is the concatenation operator. Note that if c_0 is admissible at (q, x) then Reach((q, x), c_0) is not empty; therefore, we say that $\gamma = c_0, c_1$ is admissible starting at (q, x) if $\sum(\gamma, (q, x))$ is not empty.

For a sequence γ of more than two control actions, $\sum(\gamma, (q, x))$ can be defined similarly. We denote by $S_C(A)$ the set of all admissible control action sequences. We can also define the set of traces starting at (q, x) after an admissible control sequence γ as follows:

$$Tr((q,x),\gamma) = \{\tau((q,x),\sigma) \mid \sigma \in \Sigma(\gamma,(q,x))\}.$$

Observations

In this work, we focus on behavior of analog signals and thus the properties of interest involve only continuous variables. We assume a set $V_o(A) \subseteq V(A)$ of continuous variables of the hybrid automaton that are observable by the tester. We also assume that any change in location can be observed by the tester. Given a hybrid state s = (q, x), cont(s) gives the continuous component of s. The definition can be extended to a sequence of states $\tau = (q_0, x_0), \ldots, (q_k, x_k)$ as follows: $cont(\tau) = x_0, \ldots, x_k$. The projection of a continuous state x on $V_o(A)$, denoted by $\pi(x, V_o(A))$, is called an observation.

Definition 4 (Observation Sequence). Let $\gamma = c_0, c_1, \ldots, c_k$ be an admissible control sequence. Let (q_{init}, x_{init}) be the initial state of A. The set of observation sequences associated with γ is

$$S_{\mathcal{O}}(\mathcal{A},\gamma) = \{\pi(cont(\tau), V_o(\mathcal{A})) \mid \tau \in Tr((q_{init}, x_{init}),\gamma)\}.$$

2.2. Specification and System under Test

We assume that the specification is modeled as a hybrid automaton A and the system under test (such as an implementation) by another hybrid automaton As such that:

- $V_o(A) \subseteq Vo(As)$ and
- $S_C(A) \subseteq S_C(A_s)$.

Note that we do not assume that we know the model A_s. For a given observation sequence O of A, the operator of projecting O on a set V of variables is defined componentwise, denoted by $\pi(O, V)$. Again, we can naturally extend this definition to a set of observation sequences $\pi(S, V) = \{\pi(O, V) \mid O \in S\}$.

2.3. Testing Problem

The goal of testing is to make statements about the relation between the traces of a system under test and a specification. The system under test A_s often operates within some environment. In our testing problem, the tester plays the role of the environment and it performs experiments on A_s in order to study the relation between A and A_s. The tester works as follows. It emits an admissible control sequence to the system under test and measures the resulting observation sequence in order to produce a verdict $v \in \{P, F, I\}$. The verdict P means 'pass' (the observation sequence is allowed by the specification), F means 'fail' (the observation sequence is not allowed by the specification), and I means 'inconclusive' (neither a 'pass' nor a 'fail' verdict can be assigned). The observations are measured at the end of each continuous control action and after each discrete (disturbance or control) action. Conformance is an important property of the relation between the system under test and the specification.

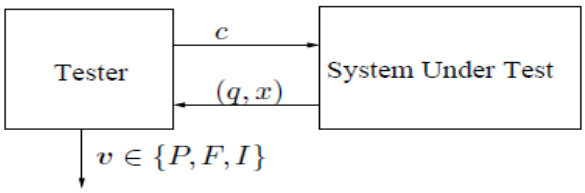

Figure 1. Test architecture.

Definition 5 (Conformance). We say that the system under test A_s is conform to the specification A, denoted by $A \approx A_p$, iff for all γ $S_C(A) : \pi(S_O(A_s, \gamma), V_o(A)) \subseteq S_O(A, \gamma)$.

Note that since $S_C(A) \subseteq S_C(As)$, a control sequence which is admissible for A is also admissible for A_s.

2.4. Test case

A test case is represented by a (finite) tree where each node is associated with an observation and each path from the root with an observation sequence. Each edge of the tree is associated with a control action.

The observation sequences of the trees are grouped into three disjoint sets: the set O_p of observation sequences that cause a 'pass´ verdict, the set O_f that cause a 'fail´ verdict, and the set O_i that cause an 'inconclusive´ verdict. Note that an observation sequence must cause a unique verdict.

Since a hybrid automaton might have an infinite number of infinite traces of a hybrid automaton; however, the tester can only perform a finite number of test cases in finite time. Therefore, we need to select a finite portion of the input space of A and test the conformance of A_s with respect to this portion. The selection is done using a coverage criterion that we formally define in the next section. Hence, our testing problem is formulated as to automatically generate a set of test cases from the specification model to satisfy this coverage criterion.

2.5. Test Coverage

Test coverage is a way to evaluate testing quality. More precisely, it is a way to relate the number of tests to carry out with the fraction of the system's behaviors effectively explored. As mentioned earlier, the classic coverage notions mainly used in software testing, such as statement coverage and if-thenelse branch coverage, path coverage (see for example [24], [22]), are not appropriate for the trajectories of continuous and hybrid systems defined by differential equations. However, the geometry of the hybrid state space can be exploited to define a coverage measure which, on one hand, has a close relationship with the properties to verify and, on the other hand, can be efficiently computed or estimated.

In circuit testing, fault coverage is an important concern. A fault is said to be detected by a test input pattern if, when applying the input pattern to the circuit, different output patterns can be observed, for the reference (non-faulty) circuit and the faulty circuit. Generally, faults can be categorized into catastrophic and parametric faults. Examples of catastrophic faults include a change in the circuit topology, a global deviation of the circuit behavior. Parametric faults refer to small changes in the parameters that do not affect the circuit functionality. For example, a band-pass filter which has a frequency response with the correct shape but it passes a larger range of frequencies. It is often assumed that beyond a deviation of 10% is considered to have caused faults.

For this reason, we are motivated in defining a coverage measure that describes how 'well' the visited states represent the set of all reachable states of the system. This measure is defined using the star discrepancy notion in statistics, which characterises the uniformity of the distribution of a point set within a region. We first briefly recall the star discrepancy. The star discrepancy is an important notion in equidistribution theory as well as in quasi-Monte Carlo techniques (see for example [4]).

Star Discrepancy: Let P be a set of k points inside

B= $[l_1, L_1]\times\ldots\times[l_n, L_n]$. Let Γ be the set of all subboxes J of the form:

$$J = \prod_{i=1}^{n} [l_i, \beta_i]$$

with $\beta_i \in [\, l_i, L_i]$ (see Figure 2 for an illustration). The local discrepancy of the point set P with respect to the subbox J is defined as follows :

$$D(P, J) = |\frac{A(P, J)}{k} - \frac{\lambda(J)}{\lambda(\mathcal{B})}|$$

where A(P, J) is the number of points of P that are inside J, and $\lambda(J)$ is the volume of the box J. The star discrepancy of P with respect to the box B is defined as:

$$D^*(P, \mathcal{B}) = sup_{J\in\Gamma} D(P, J) \qquad (2)$$

Note that $0 < D^*(P, B) \leq 1$. Intuitively, the star discrepancy is a measure for the irregularity of a set of points. A large value $D^{\llcorner}(P, B)$ means that the points in P are not much equidistributed over B. When the region is a hyper-cube, the star discrepancy measures how badly the point set estimates the volume of the cube. Since a hybrid system can only evolve within the staying sets of the locations, we are interested in the coverage with respect to these sets. For simplicity we assume that all the staying sets are boxes.

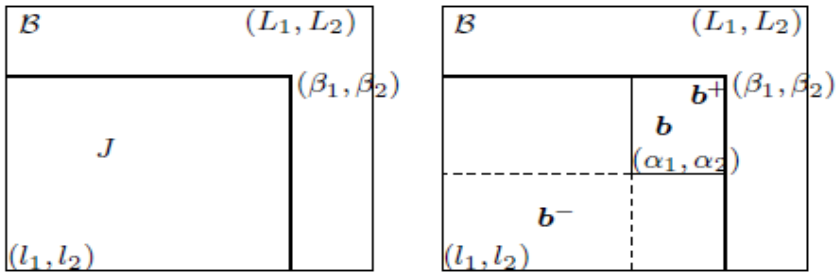

Figure 2. Illustration of the star discrepancy notion.

Definition 6 (Test Coverage). Let P = {(q, P_q) | q ∈ Q ∧ Pq ⊂ Iq} be the set of states. The coverage of P is defined as:

$$\dot{Cov}(\mathcal{P}) = \frac{1}{||Q||} \sum_{q\in Q} 1 - D^*(P_q, \mathcal{I}_q)$$

Where ||Q|| is the number of locations in Q.

If a staying set I_q is not a box, we can take the smallest oriented box that encloses it, and apply the star discrepancy definition in (2) to that box after an appropriate coordination change. We can see that a large value of Cov(P) indicates a good space-covering quality. If P is the set of states visited by a test suit, our objective is to maximize Cov(P).

3. TEST GENERATION

Our test generation is a combination of the Rapidly exploring Random Tree (RRT) algorithm (a successful robot motion planning technique [17], [32]) and a guiding tool used to achieve a good coverage of the system's behaviors we want to test. We call the resulting algorithm gRRT. In this section, we present only the main ideas of gRRT, and a detailed description of the algorithm can be found in [18].

The algorithm constructs a tree denoted by *T*, the root of which corresponds to the initial state $s_0 = (q_0, x_0)$. The construction of the tree is summarized in Algorithm 1. In each iteration, a hybrid state $s_{goal} = (q_{goal}, x_{goal})$ is sampled to indicate the direction towards which the tree is expected to evolve. Expanding the tree towards sgoal is done by making a continuous step as follows:

- First, a starting state $s_{near} = (q_{near}, x_{near})$ for the current iteration is determined. It is natural to choose s_{near} to be a state near s_{goal}. The distance between two hybrid states is defined as an average length of the traces from s_{near} to s_{goal} (see [18] for more detail).
- Next, the procedure **CONTINUOUSSTEP** tries to find the input $\bar{u}_{qnear}$ to take the system from s_{near} towards s_{goal} as closely as possible after one time step *h*. This results in a new continuous state x_{new}. A new edge from s_{near} to $s_{new} = (q_{near}, x_{new})$, labeled with the associated continuous control action is then added to the tree. To find s_{new}, when the set *U* is not finite it can be sampled, or one can solve a local optimal control problem.

Then, from s_{new}, we compute its successors by all possible discrete transitions. Each time we add a new edge, we label it with the associated control or disturbance action. The algorithm terminates after reaching a satisfactory coverage value or after some maximal number of iterations. To extract a test case from the tree, we project the states at the nodes on the observable variables of A. In Algorithm 1, the function SAMPLING plays the role of guiding the exploration, which will be described later. To deal with a set of initial states, we can sample a finite number of initial states and thus the tree can have more than one root. Similarly, to deal with parameter sets, we can consider a finite number of parameter values associated and associate them with each initial state. Along a path from each root, the parameter vector remains constant and is used to determine the corresponding continuous dynamics.

Algorithm 1 gRRT - Test generation algorithm

$\mathcal{T}.init(s_0)$, $j = 1$ ▷ s_0: initial state
repeat
  $s_{goal} = \text{SAMPLING}(\mathcal{S})$ ▷ $\mathcal{S}$: hybrid state space
  $s_{near} = \text{NEIGHBOR}(\mathcal{T}, s_{goal})$
  $(s_{new}, \bar{u}_{q_{near}}) = \text{CONTINUOUSSTEP}(s_{near}, h)$
  $\text{DISCRETESTEPS}(\mathcal{T}, s_{new})$, $j++$
until $j \geq J_{max}$

Coverage Estimation

To evaluate the coverage of a set of states, we estimate a lower and upper bound of the star discrepancy of a point set (exact computation is not an easy problem). These bounds as well as the information obtained from their estimation are used to decide which parts of the state space have been 'well explored' and which parts need to be explored more. Let us briefly describe this estimation method. Let $B=[l_1, L_1]\times\ldots\times[l_n, L_n]$. We define a box partition of B as a set of boxes

$\Pi=\{b^1,\ldots,b^m\}$ such that :

$$\bigcup_{i=1}^{m} b^i = B$$

and the interiors of the boxes b^i do not intersect. Each such box is called an *elementary box* . Given a box $b = [\alpha_1, \beta_1]\times\ldots\times[\alpha_n, \beta_n] \in \Pi$, we define $b^+ = [l_1, \beta_1]\times\ldots\times[l_n, \beta_n]$ and $b^- = [l_1, \alpha_1]\times\ldots\times[l_n, \alpha_n]$ (see Figure 2 for an illustration).

For any finite box partition Π of B, the star discrepancy D*(P,B) of the point set P with respect to B satisfies: $C(P, \Pi) \leq D^*(P,B) \leq B(P, \Pi)$ where the upper and lower bounds are:

$$B(P,\Pi) = \max_{b\in\Pi} \max\left\{\frac{A(P,b^+)}{k} - \frac{\lambda(b^-)}{\lambda(\mathcal{B})}, \frac{\lambda(b^+)}{\lambda(\mathcal{B})} - \frac{A(P,b^-)}{k}\right\} \text{ and}$$

$$C(P,\Pi) = \max_{b\in\Pi} \max\left\{\left|\frac{A(P,b^-)}{k} - \frac{\lambda(b^-)}{\lambda(\mathcal{B})}\right|, \left|\frac{A(P,b^+)}{k} - \frac{\lambda(b^+)}{\lambda(\mathcal{B})}\right|\right\}.$$

Coverage-Guided Sampling

In each iteration, we want to bias the goal state sampling distribution according to the current coverage of the visited states. More concretely, we first sample a discrete location and then a continuous state. Let $P = \{(q, P_q) \mid q \in Q \wedge P_q \subset I_q\}$ be the current set of vi sited states. The discrete location sampling distribution depends on the current continuous state coverage of each location:

$$Pr[q_{goal} = q] = \frac{D^*(P_q, \mathcal{I}_q)}{\sum_{q'\in Q} D^*(P_{q'}, \mathcal{I}_{q'})}.$$

We now show how to sample x_{goal}, assuming that we have already sampled a discrete location $q_{goal} = q$. The sampling process consists of two steps. In the first step, we sample an elementary box b_{goal} from the set Π; in the second step we sample a point x_{goal} in b_{goal} uniformly. The elementary box sampling distribution in the first step is biased in order to optimize the coverage. The intuition behind this guiding strategy is to favor the selection of an elementary box such that a new point x added in this box results in a reduction of the lower and upper bounds. In words, the essential idea of this guiding method is that we use the information about the current coverage in order to improve it (see [18]). An important remark is that when the propery of interest involves only a subset of continuous variables, we can define a coverage measure with respect to the projection of the state space on these variables, and use it to guide the sampling process.

Before continuing, we recall that the gRRT algorithm preserves the probabilistic completeness of the RRT algorithm.The proof can be found in [18]. Roughly speaking, the probabilistic completeness property states that if the trace we seach for is feasible, then the probability that the algorithm finds it approaches 1 as the number k of iterations approaches infinity. This property is a way to explain a good spacecovering property of the RRT algorithm and its successes in solving practical motion planning problems [17]. Our test generation algorithm has been tested on a number of control applications, which proved its scalability to high dimensional systems [18].

4. APPLICATION TO BENCHMARK CIRCUITS

We have implemented a test generation tool using the above described algorithm. For circuit applications, we use the well-known RADAU algorithm for solving DAEs [12]. Before presenting the experimental results obtained for three benchmark circuits, we recall that solving high index1 and stiff DAEs is computationally expensive, and in order to evaluate the efficiency of the test generation algorithm, we have chosen two practical circuits with DAEs of this type. The three circuits we treated are: a transistor amplifier, a voltage controlled oscillator, and a Delta-Sigma modulator circuit.

4.1. Transistor Amplifier

The first example is a transistor amplifier model, taken from [12]. Its diagram is shown in Figure 3, where the variable y_i is the voltage at node i; U_e is the input signal and $y_8 = U_8$(node 8) is the output voltage. The circuit equations are a system of DAEs of index 1 with 8 continuous variables. The matrix M is given by:

$$\begin{pmatrix} -C_1 & C_1 & 0 & 0 & 0 & 0 & 0 & 0 \\ C_1 & -C_1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -C_2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -C_3 & C_3 & 0 & 0 & 0 \\ 0 & 0 & 0 & C_3 & -C_3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -C_4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -C_5 & C_5 \\ 0 & 0 & 0 & 0 & 0 & 0 & C_5 & -C_5 \end{pmatrix}.$$

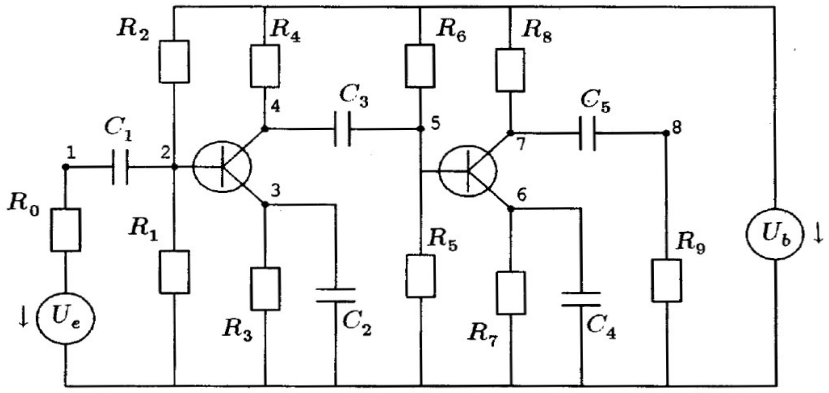

Figure 3. Transistor amplifier circuit [12].

The function *f* is given by:

$$\begin{pmatrix} -U_e/R_0 + y_1/R_0 \\ -U_b/R_2 + y_2(1/R_1 + 1/R_2) - (\alpha - 1)g(y_2 - y_3) \\ -g(y_2 - y_3) + y_3/R_3 \\ -U_b/R_4 + y_4/R_4 + \alpha g(y_2 - y_3) \\ -U_b/R_6 + y_5(1/R_5 + 1/R - 6) - (\alpha - 1)g(y_5 - y_6) \\ -g(y_5 - y_6) + y_6/R_7 \\ -U_b/R_8 + y_7/R_8 + \alpha g(y_5 - y_6) \\ y_8/R_9 \end{pmatrix}.$$

The circuit parameters are: $U_b = 6$; $U_F = 0.026$; $R_0 = 1000$; $R_k = 9000$, k = 1, . . . , 9; $C_k = k10^{-6}$; $\alpha = 0.99$; $\beta = 10^{-6}$.The initial state $y_{init} = (0, U_b/(R2/R1 + 1), U_b/(R2/R1 + 1), U_b, U_b/(R_6/R_5+1), U_b/(R_6/R_5+1), U_b, 0)$

[1]The differential index of a DAE is a measure of the singularity of the DAE. It characterizes the difficulty in numerically solving the equation.

We are interested in studying the influence of circuit parameter uncertainty on the transient properties, such as overshoot, stabilisation time. The uncertainty we consider is indeed a perturbation in the function describing the relation between the current through the source of the two transistors and the voltages at the gate and source

$$I_S = g(U_G - U_S) = \beta(e^{\frac{U_G - U_S}{U_F}} - 1) + \epsilon,$$

With $\epsilon \in [\epsilon_{min}, \epsilon_{max}] = [-5e-5, 5e-5]$.

The input signal $U_e(t) = 0.1\sin(200\ \boldsymbol{\pi}\ t)$. We used the gRRT algorithm to generate a test case, which indicates the presence of traces with overshoots (the acceptable interval of U_8 in the non-perturbed circuit is [−3.01, 1.42]). Figure 4 shows the generated observation sequences projected on the variable U_8 over time.

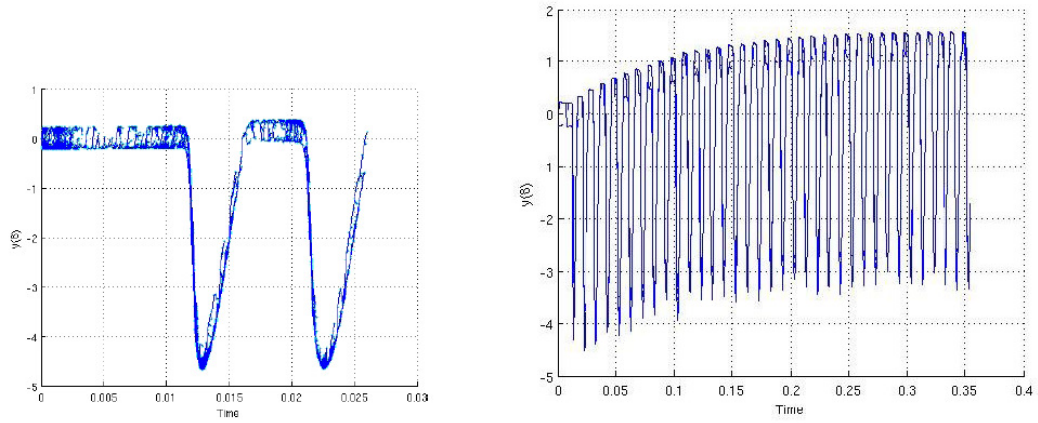

Figure 4. Test generation result for the transitor amplifier (first figure: zoom in the first 0.03s).

4.2. Voltage Controlled Oscillator

The second circuit we examined is a voltage controlled oscillator (VCO) circuit [11], shown in Figure 5. The behavior of this circuit can be represented by a system of DAEs with 55 continuous variables. We are interested the oscillating frequency of the variables v_{C1} and v_{C2} . Indeed, this frequency is a linear function of the input voltage. We study the influence of a time-variant perturbation in C2 on the frequency. This perturbation is modeled as an input signal. In this example we show that, in addition to conformance relation, using this framework, we can test a property of the input/output relation. More precisely, for a given input sequence, we want compare the observation sequences of the specification A and those of the system under test A_s, with respect to a property.

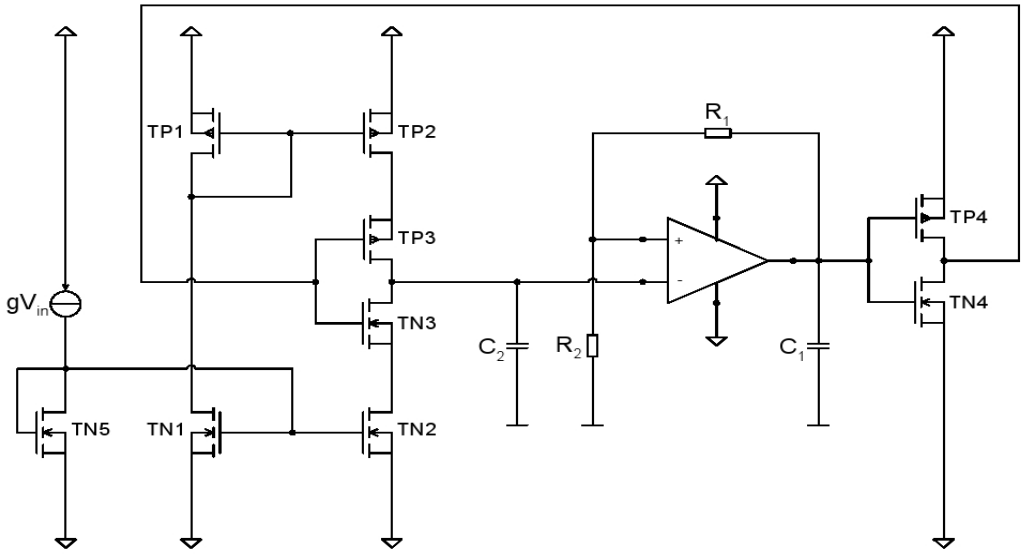

Figure 5. Voltage controlled oscillator (VCO) circuit.

As an example, the oscillating period $T \pm \boldsymbol{\delta}$ of a variable $x_1 = v_{C2}$ can be expressed using a simple automaton with one clock y in Figure 6 (similar to a monitor automaton in [10]). The question is to know if given an oscillating trace in A, its corresponding trace in A_s is also oscillates with the same period. This additional automaton can be used to determine test verdicts for the traces in the computed test cases. If we are additionally interested in a property which states that all traces of the system under test oscillate with the period $T \pm \boldsymbol{\delta}$. If an observation sequence corresponds to a path entering the 'Error´ location, then it causes a 'fail´ verdict.

Since we cannot use finite traces to prove a safety property, the set of observation sequences that cause a 'pass´ verdict is empty, and therefore the remaining observation sequences (that do not cause a 'fail´ verdict) causes a 'inconclusive´ verdict. We consider a constant input voltage $u_{in} = 1.7$. The generated test case shows that after the transient time, under a time variant deviation of C2 which ranges within ±10% of the value of $C_2 = 0.1e{-}4$, the variables v_{C1} and v_{C2} oscillate with the period $T \in [1.25, 1.258]s$ (with $\varepsilon = 2.8e - 4$). This result is consistent with the result presented in [11]. The number of generated states was 30000 and the computation time as 14mn. In this experiment, the coverage measure was defined on the projection of the state space on v_{C1} and v_{C2} . Figure 7 shows the explored traces of v_{C2} over time.

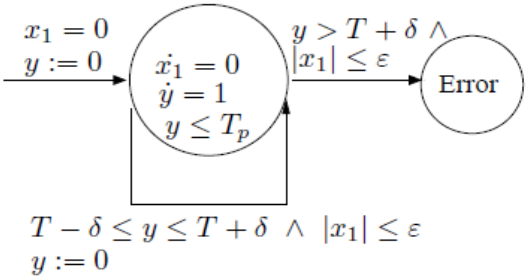

Figure 6. Automaton for an oscillation specification.

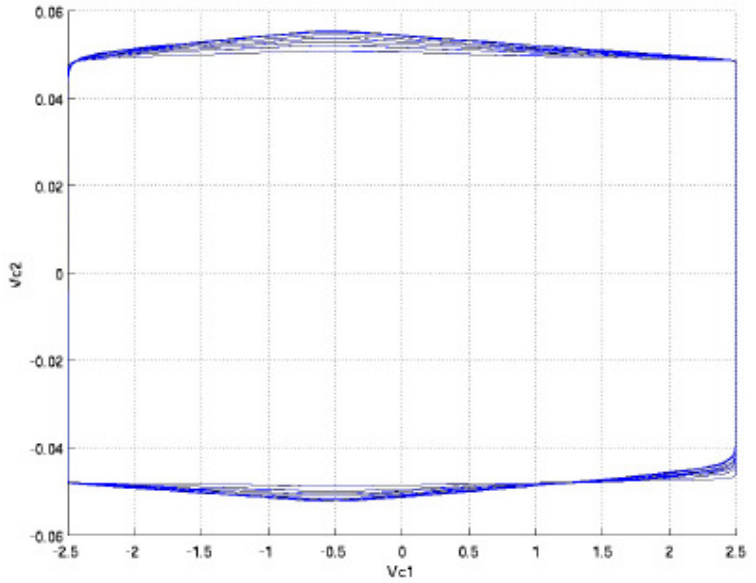

Figure 7. Variable vC2 over time

4.3. Delta-Sigma Circuit

The third example is a Delta-Sigma modulator [19], which is a mixed-signal circuit. The Delta-Sigma modulation is a very popular technique to perform analog to digital conversion. Its principle can be briefly explained as follows. When the input sinusoid is positive and its value is less than 1, the output takes the +1 value more often and the quantization error which is the difference between the input and the output of the quantizer is fed back with negative gain and 'accumulated' in the integrator $\frac{1}{z-1}$.

Then, when the accumulated error reaches a certain threshold, the quantizer switches the value of the output to −1 for some time, which reduces the mean of the quantization error. A third-order Delta-Sigma modulator is modeled as a hybrid automaton, shown in Figure 9. It is called third order since it uses a third order filter to process noise. Higher-order modulators achieve better performance but induce stability problems. A modulator is said to be stable if under a bounded input, the states of its integrators are bounded. Stability analysis for such circuits is still a challenging problem [19], due to the presence of two sources of non-linearities: saturation and quantization. The discrete time dynamics of the system is as follows:

$$x(k+1) = Ax(k) + bu(k) - sign(y(k))a, \quad (3)$$
$$y(k) = c_3x_3(k) + b_4u(k), \quad (4)$$

Where matrix A, vectors a and b are constants depending on the various gains of the model, $x(k) \in \mathfrak{R}^3$ represents the integrator states, $u(k) \in \mathfrak{R}$ R is the input, $y(k) \in \mathfrak{R}$ is the input of the quantizer. Thus, its output is v(k) = sign(y(k)), and one can see that whenever v remains constant, the system dynamics is affine continuous.

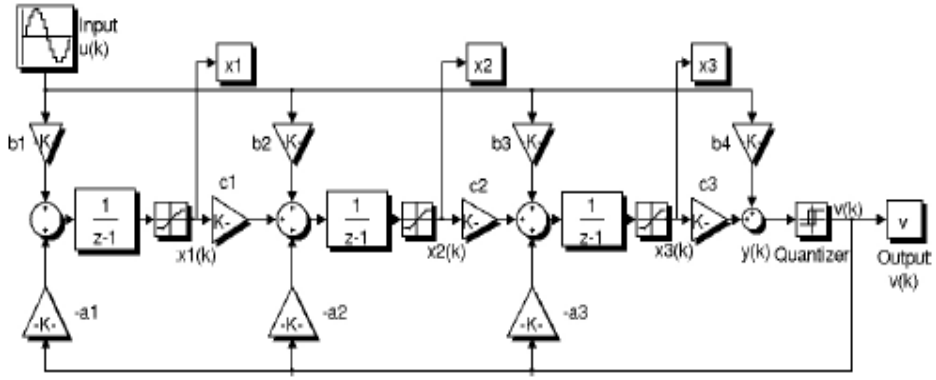

Figure 8. Model of a third-order modulator: Saturation blocks model saturation of the integrators.

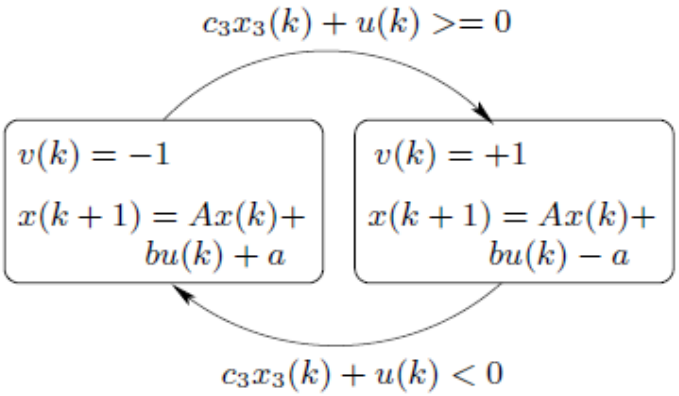

Figure 9. Third-order Delta-Sigma modulator.

The test generation algorithm was performed for the initial state x(0) inside the set $[-0.01, 0.01]^3$ and the input values $u(k) \in [-0.5, 0.5]$. After exploring only 57 states, saturation was already detected. The computation time was less than 1s. Figure 10 shows the values of $(\sup x1(k))_k$ as a function of the number k of time steps. We can see that the sequence $(\sup x1(k))_k$ leaves the safe interval $[-x_1^{sat}, x_1^{sat}] = [-0.2, 0.2]$, which indicates the instability of the circuit. This instability for a fixed finite horizon was also detected in [7] using an optimization-based verification method.

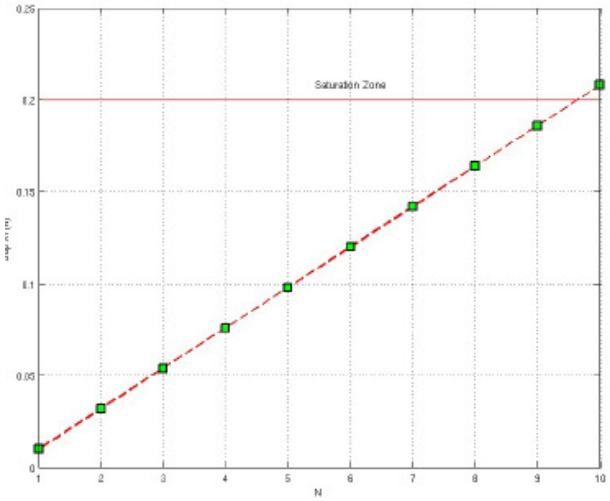

Figure 10. Test generation result for the Delta-Sigma circuit.

5. RELATED WORK

Related work in analog and mixed-signal circuit testing was already discussed in the introduction. Here, we only discuss related work in hybrid systems testing. Classical model-based testing frameworks using finite labeled transition systems has been applied to digital circuits, communication protocols and software. They have recently been extended to hybrid systems. The paper [21] proposed a framework for generating test cases from simulation of hybrid models specified using the language CHARON. A probabilistic test generation approach, similar to ours, was presented in [9]. This paper also proposed a coverage measure based on a discretized version of dispersion.While in our work, the coverage measure is used to guide the simulation, in [9] it is used as a termination criterion. The RRT algorithms have also been used to solve other problems such as hybrid systems control and verification [8], [6].

6. CONCLUSIONS

To conclude, in this paper we described a test generation technique for analog and mixed signal circuits using a hybrid system framework. The application of the proposed technique was demonstrated with a number of benchmark circuits. A number of directions for future research can be identified.First, we are interested in enriching our framework to capture partial observability and measurement imprecisions. In order to facilitate the application to practical circuits, we need a tool for automatic generation of hybrid automata from commonlyused circuit descriptions, such as Spice netlists. On the other hand, as mentioned earlier, an efficient and reliable simulation method is a key ingredient in our approach. The state of the art SPICE simulator is prone to convergence problems when the dynamics of the circuit has fast variations caused by components with stiff characteristics. In collaboration with researchers at INRIA Rhône-Alpes, a topic of our undergoing research is to integrate in our test generation tool a simulation algorithm based on the non-smooth approach [3].

REFERENCES

[1] R. Alur, C. Courcoubetis, N. Halbwachs, T.A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. Theoretical Computer Science, 138:3–34, 1995.

[2] B. Ayai, N. Ben-Hamida, and B. Kaminska. Automatic test vector generation for mixed-signal circuits. In Proc. European Design and Test Conference.,pages 458-463, March 1995.

[3] V. Acary and F. Pérignon. SICONOS: A software platform for modeling, simulation, analysis and control of Non Smooth Dynamical system. Proc. Of MATHMOD, ARGSIM Verlag, 2006.

[4] J. Beck and WWL. Chen. Irregularities of Distribution. Cambridge Univ. Press, 1987.

[5] B. Burdiek Generation of Optimum Test Stimuli for Nonlinear Analog Circuits Using Nonlinear Programming and Time-Domain Sensitivities. In Proc.DATE, pages 603-609, 2001.

[6] M. S. Branicky, M. M. Curtiss, J. Levine, and Stuart Morgan. Sampling-based reachability algorithms for control and verification of complex systems.Proc. Thirteenth Yale Workshop on Adaptive and Learning Systems, New Haven, CT, 2005.

[7] T. Dang, A. Donzé, and O. Maler. Verification of analog and mixed-signal circuits using hybrid systems techniques. In FMCAD, LNCS, Springer, 2004.

[8] A. Bhatia and E. Frazzoli. Incremental Search Methods for Reachability Analysis of Continuous and Hybrid Systems. HSCC, LNCS 2993, pages 142-156, Springer, 2004.

[9] J.M. Esposito, J. Kim, and V. Kumar. Adaptive RRTs for validating hybrid robotic control systems. In Int. Workshop on the Algorithmic Founddations of Robotics , 2004.

[10] G. Frehse, B. Krogh, R. Ruttenbar, and O. Maler. Time domain verification of oscillator circuit properties. Proceeding of Workshop on Formal Verification of Analog Circuits (ETAPS Satellite Event), Edinburgh, ENTCS 153(3): 9-22, 2005.

[11] D. Grabowski, D. Platte, L. Hedrich and E. Barke Time Constrained Verification of Analog Circuits using Model-Checking Algorithms. Proceeding of Workshop on Formal Verification of Analog Circuits (ETAPS Satellite Event), Edinburgh, ENTCS 153(3): 37-52, 2005.

[12] E. Hairer, C. Lubich and M. Roche. The Numerical Solution of Differential-Algebraic Systems by Runge Kutta Methods. Lecture Notes in Mathematics 1409, Springer-Verlag, 1989.

[13] N. B. Hamida, K. Saab, D. Marche, B. Kaminska, and G. Quesnel. LIMSoft: Automated Tool for Design and Test Integration of Analog Circuits. In Proc. International Test Conference, 1996.

[14] N. Ben-Hamida, K. Saab, D. Marche, and B. Kaminska. A perturbation based fault modeling and simulation for mixed-signal circuits. In Proc. IEEE Asian Test Symposium, pages 182-187, 1997.

[15] H. Kerkhoff, R. Tangelder, H. Speek, and N. Engin. MISMATCH: A Basis for Semi-automatic Functional Mixed-Signal Test-Pattern Generation. In Proc. IEEE Int. Conf. on Electronics, Circuits, and Systems, pages 1072-1075, 1996.

[16] KuffnerLaValle00 J. J. Kuffner and S. M. LaValle. RRT-connect: An efficient approach to single query path planning. In Proc. IEEE Int. Conf. On Robotics and Automation, 995–1001, 2000.

[17] S. M. LaValle and J. J. Kuffner. Rapidly-exploring random trees: Progress and prospects. In Algorithmic and Computational Robotics: New Directions, pages 293–308. A K Peters, Wellesley, MA, 2001.

[18] T. Nahhal and T. Dang. Test Coverage for Continuous and Hybrid Systems In Computer Aided Verification CAV'07, LNCS, Berlin.

[19] B. Pérez-Verdú and F. Medeiro and A. Rodríguez-Vázquez. Top-Down Design of High-Performance Sigma-Delta Modulators, chapter 2. Kluwer Academic Publishers, 2001.

[20] R. Shi and W.T. Tian. Automatic Test Generator of Linear Analog Circuits Under Parameters Variations. In Proc. Asian and South Pacific Design Automation Conference, 1998.

[21] L. Tan, J. Kim, O. Sokolsky and I. Lee. Model-based Testing and Monitoring for Hybrid Embedded Systems. In IRI, 487-492, 2004.

[22] J .Tretmans. Testing Concurrent Systems: A Formal Approach. In Int. Conference on Concurrency Theory CONCUR, LNCS 1664, Springer, 1999.

[23] W. Verhaegen, G. Van der Plas, and G. Gielen. Automated test pattern generation for analog integrated circuits. In Proc. IEEE VLSI Test Symposium, pages 296-301, 1997.

[24] H. Zhu, P.A.V. Hall, and J.H.R. May. Software Unit Test Coverage and Adequacy. In ACM Computing Surveys, 29, 4. Dec. 1997.

[25] M. Zwolinski, S.J. Spinks, C.D. Chalk, and I.M. Bell. Generation and Verification of Tests for Analog Circuits Subject to Process Parameter Deviations.In IEEE Int. Symp. Defect and Fault Tolerance in VLSI Systems, Paris, October 1997.

[26] Afaq Ahmad. A Simulation Experiment on a Built-In Self Test Equipped with Pseudorandom Test Pattern Generator and Multi-Input Shift Register (MISR).In VLSICS, December 2010, Volume 1, Number 4.

[27] S. L. Haridas1 and N. K. Choudhari2. A Low Power VITERBI Decoder Design with Minimum Transition Hybrid Register Exchange Processing For Wireless Applications. In VLSICS, December 2010, Volume 1, Number 4.

[28] Subashri T, Arunachalam R, Gokul Vinoth Kumar B and Vaidehi. Pipelining Architecture of AES Encryption and Key Generation with Search Based Memory. In VLSICS, December 2010, Volume 1, Number 4.

[29] Mouna Karmani, Chiraz Khedhiri and Belgacem Hamdi. Design and test challenges in Nano-scale analog and mixed CMOS technology. In VLSICS, June 2011, Volume 2, Number 2.

[30] Chiraz Khedhiri1, Mouna Karmani1 and Belgacem Hamdi. A BIST Generator CAD Tool for Numeric Integrated Circuits. In VLSICS, June 2011, Volume 2, Number 2.

[31] Shweta S. Meshram1 and Ujwala A. Belorkar. Design Approach for Fault Tolerance in FPGA Architecture. In VLSICS, March 2011, Volume 2, Number 1.

[32] S. Karaman and E. Frazzoli. Sampling-based Algorithms for Optimal Motion Planning. In Journal of Robotics Research, June 2011, Volume 30, Pages 846--894.

Authors

Tarik NAHHAL received his M.Sc. in Computer Science from Evry University of science and Technology, France, in 2004 and his PhD in Computer Science from Joseph Fourier University, France 2007. He is currently an Assistant Professor in the Department of Mathematics and Computer Science at the Hassan II University in Casablanca, Moroco.His research interests include formal methods for testing and monitoring of Embedded Systems.

Thao Dang received her diplome d'Ingénieur and her M.Sc. in Electrical Engineering in 1996 from Ecole Nationale Supérieured'Ingénieurs Electriciens of Grenoble.She received her PhD in automatic Control in 2000 from the VERIMAG laboratory in Grenoble In 2001 she worked as a postdoctoral research associate at the Department of Computer and Information Science of the University of Pennsylvania. Since 2002, she is a research scientist at the CNRS and a membre of the VERIMAG laboratory. Her research interests are modeling, verification and control of hybrid system and their application in design and analysis of embedded real-time systems.