Microelectronics Journal 39 (2008) 1395–1404

Contents lists available at ScienceDirect

Microelectronics Journal

journal homepage: www.elsevier.com/locate/mejo

Review

Formal verification of analog and mixed signal designs: A survey ☆

Mohamed H. Zaki [a], Sofiène Tahar [a,*], Guy Bois [b]

[a] Department of Electrical and Computer Engineering, Concordia University, Montreal, Quèbec, Canada
[b] Genie Informatique, Ecole Polytechnique de Montreal, Montreal, Quèbec, Canada

ARTICLE INFO

Article history:
Received 24 February 2008
Accepted 13 May 2008
Available online 14 July 2008

Keywords:
Formal verification
Analog and mixed signal design verification
Survey

ABSTRACT

Analog and mixed signal (AMS) designs are an important part of embedded systems that link digital designs to the analog world. Due to challenges associated with its verification process, AMS designs require a considerable portion of the total design cycle time. In contrast to digital designs, the verification of AMS systems is a challenging task that requires lots of expertise and deep understanding of their behavior. Researchers started lately studying the applicability of formal methods for the verification of AMS systems as a way to tackle the limitations of conventional verification methods like simulation. This paper surveys research activities in the formal verification of AMS designs as well as compares the different proposed approaches.

Contents

1. Introduction

The latest technological advancement of integrated circuits design and semiconductors manufacturing paved the way to the development of system on chip (SoC) designs. On the other hand, a cornerstone in embedded systems are analog and mixed signal (AMS) SoC designs (see Fig. 1), which are integrated circuits, required at the interfaces with the real world environment [1]. Among the important functionalities of AMS designs are the processing of analog signal on the front and back ends of the

☆ Expanded version of the conference paper "Formal Verification of Analog and Mixed Signal Designs: Survey and Comparison", M. Zaki, S. Tahar, and G. Bois; Proceedings of the IEEE Northeast Workshop on Circuits and Systems (NEWCAS'06), 2006, pp. 281–284.

* Corresponding author. Tel.: +1 514 848 2424 x 3114.
E-mail addresses: mzaki@ece.concordia.ca (M.H. Zaki), tahar@ece.concordia.ca (S. Tahar), guy.bois@polymtl.ca (G. Bois).

0026-2692/$ - see front matter © 2008 Elsevier Ltd. All rights reserved.
doi:10.1016/j.mejo.2008.05.013

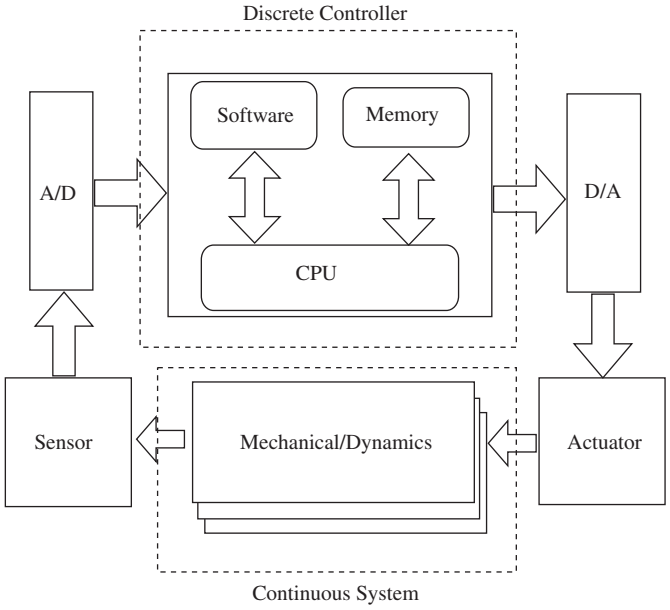

Fig. 1. Embedded system architecture.

system. Other functionalities include converting between analog and digital data representation, frequency synthesis and generating timing reference. In addition, analog circuits are used for biasing which is necessary for correct and stable operations of the system.

Computer-aided design (CAD) tools for AMS design have been proposed and developed to overcome challenges in the design process of such circuits. For instance, the need to design and improve the quality of more complex integrated systems with the tight constraints of increasingly shorter time-to-market and productivity increase. Such CAD tools and concepts are then needed to provide unique insights into the behavior and characteristics of the integrated circuits, to help the designer select best design strategies. Finally, CAD tools should tackle the crucial activity to correctly and efficiently model as well as simulate performance behavior of AMS designs.

In recent years, some breakthrough have been made in different aspects of the CAD procedure, especially in the development of hardware description languages (HDL) suitable to describe the different AMS behaviors [2]; e.g., VHDL-AMS [3] and Verilog-AMS [4]. Other advances have been made in the design procedure, namely analog synthesis and topology selections (in top-down methodologies), design related optimizations like design centering and device sizing and analog layout automation [5]. One important aspect of CAD design is analysis and verification which is a challenging task that requires lots of expertise and deep understanding of AMS behavior.

Verification challenges arise throughout the different phases of the design process. In addition, unlike digital designs, a wide difference of properties and requirements exists between the different classes of AMS designs. For instance, among the verification challenges at the implementation level of analog circuits are the functionalities defined directly in terms of continuous electrical quantities. These are usually sensitive to environment factors like signal noise, temperature, in addition to higher order physical effects like different parasitics and current leakage. As a consequence the physical implementation is subject to frequent extractions, analysis and iterated modifications.

Additionally, problems ranging from correct functionality of the integrated designs to conformance to system specification like area and power consumption need to be considered and tackled appropriately in order to deliver the correct design. For example, clock jitter is a functional verification challenge for AMS designs where PLL designs are used. Jitter can be defined as the deviations in a clock output transition from ideal positions. Clock jitter effect is apparent where the conversions between analog and digital signals are done at higher resolutions and higher data rates. Another verification issue at the functional level is the stability requirement of Δ-Σ modulators. A Δ-Σ modulator is said to be stable if the integrator output remains bounded under a bounded input signal. Gain parameters for instance affect the stability, hence careful selection of such parameters is usually required.

Verification issues arise throughout the design flow. For a consistent design flow, a compliance certificate approving the correspondence between different design levels (or different designs at a specific level) is needed to ensure correctness of the end product and its conformity to the specification. For instance, in the bottom-up design methodology (see Fig. 2), the process starts with the design of the individual blocks, which are verified individually and then combined to form the system. However, the verification can be quite expensive as the entire system is represented at the transistor level. A solution to the above-mentioned problem lies in the integration at a higher level than the implementation level, such that the analysis for the whole design can be applied. This is achieved by the development of symbolic analysis which are simplification methods developed to obtain simplified models (e.g., macromodel, behavioral models) preserving the properties of interest. To ensure correctness of the methodology, some notion of equivalence needs to be verified between the implementation and the generated models. In addition, we need to ensure that extracted models when combined preserve specification properties.

To tackle analysis and verification issues, simulation and testing techniques were developed for the verification of AMS designs [5]. Traditionally, simulation was used where the evaluation of the results is often done manually in an informal fashion and the search of the state space is not complete. The simulation goal varies from DC and operating point analysis, (linearized) small signal analysis; i.e., AC, noise and distortion analysis and transient analysis used to predict the non-linear behavior of a circuit and periodic steady state analysis. However, simulation lacks the rigor needed to ensure correctness of the design as well as it does not provide the guarantees needed for correct correspondence between the implementation and the approximate models at subsequent design levels, or two models at the same level where robustness and parameter tolerances are considered. In addition, such method falls short to validate interesting properties of the design behavior such as temporal requirements.

The relative success of formal methods for digital design verification [6] has triggered research for more complex systems like real-time, control and hybrid systems [7]. In general, formal verification consists of mathematically establishing that an implementation satisfies its specification. The implementation refers to the system design which should be verified. This entity can represent a design description at any level of the system abstraction hierarchy. The specification usually refers to the property with respect to which correctness is to be determined. It can be expressed in a variety of ways, such as behavioral description, an abstract structural description, a timing diagram which reflects the behavior of the system at different time points, a temporal-logic formula, etc. Using formal methods, a decision procedure checks whether a mathematical model for the design satisfies some given properties in the specification. Formal verification techniques naturally group themselves into theorem proving methods and automated state space exploration methods.

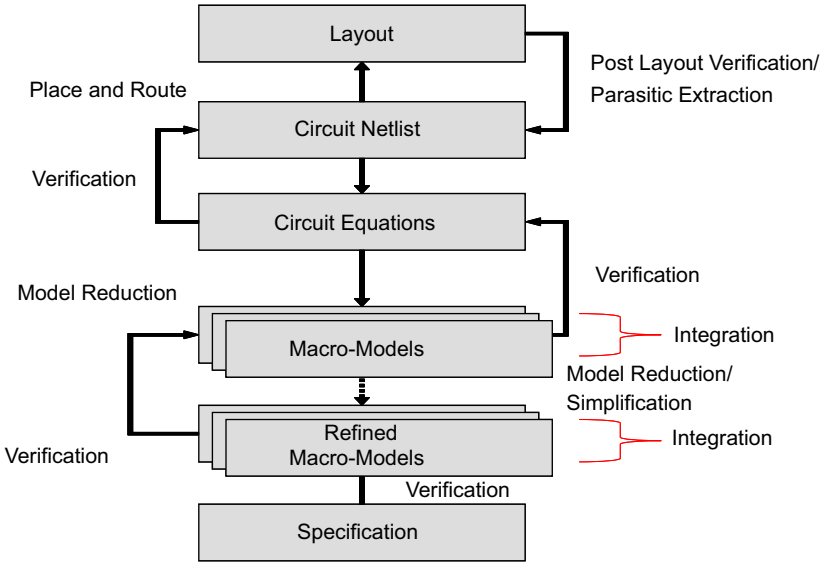

Fig. 2. Bottom-up design methodology.

In theorem proving [6], also known as proof based methods, the designer constructs a mathematical proof that a model or a structure meets their specification to be proven within the logic system, using axioms and inference rules. Thus, theorem proving is a powerful verification technique. It can provide a unifying framework for various verification tasks at different hierarchical levels. However, the task of proving complex theorems requires the aid of experts, in addition to great effort and creativity on the part of the user.

The main state space exploration methods are equivalence checking and model checking [8]. In equivalence checking, the output signals of two different models of the designs are compared for a given set of input conditions. In contrast to theorem proving, no mathematical proof needs to be developed, but the correctness of the method relies on the exploration and comparison of the reachable state spaces. In model checking, the model of the design under verification is a kind of transition system describing all its possible behaviors. The specification is a temporal-logic formula property. State exploration algorithms are then applied to check whether the model satisfies the given property or not. In case the property does not hold, a counter-example describing the failure point(s) is generated. Then the design can be corrected and reverified. The main advantage of the state exploration method is its automation, while the main drawback is the state explosion problem which is the principle limiting factor of the technology. The efficiency of state exploration and model checking methods depends heavily on the size of the reachable state space. The larger the state space, the more time and memory it takes to verify a system.

During the last two decades, formal verification has been applied to digital hardware and software systems. Recently, however, formal verification techniques have been adapted and applied to the verification of AMS systems as a way to tackle the limitations of conventional simulation techniques [9]. In addition, hybrid semi-formal techniques combining simulation and formal based methods have been developed as a way to benefit from the advantages of these methods, where logical models are used to analyze the simulation results. In this paper, we provide a survey and comparison of the research activities in the field of formal verification of AMS design. We survey formal techniques used in AMS verification along with comparison of the relevant projects. We end each section with discussions about its pros and cons along with perspectives for future directions. Equivalence checking methods applied to AMS designs are surveyed in Section 2, followed by model checking and reachability techniques in Section 3, run-time verification in Section 4 and deductive methods in Section 5, before concluding with a general discussion in Section 6.

2. Equivalence checking

2.1. Introduction

Equivalence checking is a problem where we are given two system models and are asked whether these systems are equivalent with respect to some notion of conformance, or functionally similar with respect to their input–output behavior [6]. Verification can be based on specific properties like transient or steady state response properties, in time domain or frequency domain. Such correspondence relation between designs is classically done through exhaustive testing by proving that two expressions are equivalent, which can be a difficult task for any reasonably large circuit. Instead, symbolic reasoning methods can prove or disprove equivalence using decision procedures over the whole range of inputs described symbolically. Therefore, it is possible to compare circuits on the same level of abstraction as well as on different levels, e.g., SPICE netlists versus analog behavioral models, behavioral versus macromodel, or macromodel versus device level, etc.

Consider an AMS $\mathscr{C}$ design which is represented with two different models $\mathrm{Mod}_1(\mathscr{C})$ and $\mathrm{Mod}_2(\mathscr{C})$. The models can be a transfer function if the analysis is in frequency domain or system of equations (ODE, differential algebraic equations DAE or algebraic equations) if the analysis is in the time domain. The equivalence checking problem is described as

$$\forall Ip\ \forall \mathrm{Par}.\ \mathrm{Mod}_1(\mathscr{C}) \sim \mathrm{Mod}_2(\mathscr{C})$$

where Ip is the set of input signals, Par is the set of parameters variation and $\sim$ is an equivalence relation.

An important requirement in behavior equivalence is the specification of tolerance or bounds on parameters and signals which may be needed. A failure occurs if the comparison finds that the results of both design levels are different or different beyond a certain tolerance. In the rest of this section, we survey the relevant work dealing with the equivalence checking problem.

A comparison between these work is outlined in the end of the section.

2.2. Relevant work

In [10], the authors proposed a method for applying equivalence checking between two designs (e.g., specification and implementation) of analog systems described by their linear transfer function. The verification idea is based on the discretization of the transfer functions to the Z-domain using bilinear transformation, thereby, the design can be represented in terms of discrete-time components and encoded into FSM representation like binary decision diagrams (BDDs). The verification problem can be stated as follows: the transient behavior of the implementation mimics that of the specification iff for any initial state of the specification, there exists a state in the implementation such that the FSMs representing the two circuits produce identical output sequences for all input sequences.

The discretization of the behavior raises issues like the error analysis which must be accounted for tolerance between the output sequences for both models must be specified. Another issue is state space explosion when the inherited discretization of the design is encoded. This is largely due to the large word size used to encode real signals. Finally, the methodology is only practical for linearized systems as transfer function generation for non-linear circuits is very difficult in general.

Realizing the coefficient of a transfer function exactly using actual components and devices is not always possible as the tolerance region around nominal characteristic must be taken into account. The ideas in [10] have been extended in [11] in the following way. Given the transfer function description of both the specification and implementation, verify the conformance of the magnitude and phase response of the implementation against the specification over a desired frequency range. The equivalence verification problem is modeled in [11] as an optimization problem by ensuring that the implementation response is bounded within an envelope around the specification under the influence of parameter variation.

The conformance in [11] is defined using the notion of different frequency bands product response functions (PRFs) of both design models and which serve as objective functions for the global optimization routine. Such definition allows s-domain verification, hence avoiding loss of precision due to the bilinear transformation used in [10].

Conformance checking with parameters variation was also investigated in [12], where the authors present an equivalence checking for linear analog circuits to prove that an actual circuit fulfills a specification in a given frequency interval for all parameter variations. Linear analog circuits can be described by transfer functions, extracted from the netlist by symbolic analysis methods (in case of implementation), resulting in a parameterized description of the circuit behavior. The main idea of the procedure is to compare by inclusion the value sets of the transfer functions of specification and implementation. To ensure soundness, the authors chose an over-approximation for the implementation transfer function while an under-approximation is chosen for the specification transfer function.

Comparing [10] with [12], we see that in the first work, the authors trade-off accuracy for practicality. They adapt the developed technology based on BDD equivalence checking for verification of analog systems. This comes at the cost of precision which is affected by the discretization process. In contrast, the authors in the second work insist on soundness by checking that the implementation of the behavior is included in the specification behavior.

While the above-mentioned work are concerned with frequency domain verification, others tend to focus on verification in time domain. For instance, in [13], the authors proposed an equivalence checking approach based on qualitative comparison between two representations of the non-linear analog system. However, direct comparison of vector fields for non-linear systems is usually not possible. Therefore, the authors propose to apply non-linear transformations on the sample state spaces to make the comparison possible. The difference between the evaluations of the sampled equations is then calculated allowing the identification of behavior similarity between the two designs under verification by giving an explicit error measure. Unfortunately, finding the correct transformations is a non-trivial task and automation is not possible, leading to the introduction of some heuristics to analyze and approximate qualitative behaviors of the circuits, but affecting the soundness of the methodology. The authors applied their methodology for comparison verification of two CMOS inverters with different parameters as well as the verification of an opamp against its specification.

Another equivalence checking verification approach was proposed in [14] for verifying VHDL-AMS designs. The idea is based on combining equivalence checking, rewriting systems and simulation into a verification environment. The verification methodology consists of partitioning the specification and implementation codes into digital, analog and data converter components. Digital components are verified using classical equivalence checking, while analog specification and implementation are simplified using rewriting rules and pattern matching. Furthermore, the outputs are fed to comparators to be verified by using simulation. This syntactic method can only be performed on simple designs where rewriting techniques can be easily applied. While the presented methodology is practical, it ignores the coupling between the analog and digital parts.

Such syntactic verification for analog circuits can only be applied when the designs are treated at higher level (architectural or behavioral and functional levels) as at low level, non-linear behavior makes such approaches impractical for verification. Instead of direct simulation, advanced verification techniques mentioned earlier can be used to compare analog model behaviors.

2.3. Discussion and perspectives

In general, the nature of analog circuits, most notably the presence of tolerance margins, makes equivalence verification a difficult problem. However, with careful definition of bounds on the parameters as well as the signals, certain compliance relations can be checked. In addition, in contrast to equivalence checking for digital systems where a canonical representation allows easy comparison of two functions representation, no such form exists for analog systems and all the methods presented are design driven in the sense that a priori knowledge of the qualitative and quantitative properties of the design under verification is a requirement for the methodology development. In summary, error analysis and unified equivalence theory for AMS are important candidates for the success of such verification methodology. Table 1 draws a brief comparison among the above-mentioned projects. It describes the class of system verified, the models used, the analysis regions and domains, the adopted analysis techniques, the tool used, and the case studies verified. A future application for such verification can be developed for post layout extraction. Layout extraction is the translation of an integrated circuit layout back into the electrical circuit (netlist). This extracted circuit is needed for various analysis including circuit simulation, power analysis, etc. For high frequency designs,

Table 1
Equivalence checking techniques

	[10]	[11]	[12]	[13]	[14]
Type of systems	Linear analog	Linear analog	Linear analog	Non-linear analog	Non-linear AMS
Models	Transfer function	Transfer function	Transfer function	ODE–DAE	ODE–DAE FSM
Analysis regions	Transient response	Transient response	Transient response	Near operating point	N/A
Analysis domain	Z-domain	S-domain	S-domain	Time	Time
Techniques and analysis	OBDDs comparisons	Global optimization	Interval analysis	Qualitative analysis	Rewriting, SAT simulation
Tools	N/A	Matlab	MAPLE	MAPLE	M-CHECK
Case studies	Low pass filter	Filters opamp	Band pass filter, opamp	CMOS inverter opamp	D/A converter

layout-induced parasitics significantly change the transfer function of a design, and thus modify the AC and transient behavior of a system. For example, layout parasitics are responsible for offsets of opamps and transconductors. Formal methods can play an important role for low level verification. As a complement to conventional simulation results, formal methods can be developed for compliance checking between the design circuit and the extracted one.

3. Model checking and reachability analysis

3.1. Introduction

Sometimes, we are interested in global properties connected to the dynamic behavior of the AMS system. For example, we might be interested in reachability properties, like, "From the initial states, can we reach a state where a certain condition holds?" Temporal logics [8] are logical formalisms designed for expressing such properties.

Model checking [8] is a powerful technique for the automatic verification of dynamic properties of the system. Model checking was initially developed for discrete finite-state systems and has been successful in validating communication protocols and hardware circuits. Model checking approaches completely explore the whole state space to check whether the system satisfies the desired specification. A model-checking algorithm determines whether a mathematical model of a system meets a specification that is given as a temporal-logic formula. The model checking problem is defined as follows: given a model M of a design and a property P expressed in temporal logic, check $M \models P$, i.e., check if P holds in the model M.

In recent years [15], model-checking algorithms have also been developed for real-time systems that are described by discrete programs with real-valued clocks as well as for hybrid systems. Model checking and reachability analysis of AMS designs have the potential of validating designs over a range of parameters and for all possible input signals all at once such that none of them drives the system into a bad state. An important issue is the solution of the system of differential equations; that is, the collection of continuous-time trajectories starting from a set of initial states where in practice the initial conditions are usually not known exactly but only known to lie within some range. However, the effectiveness of model checking is severely constrained by the state space explosion problem and even undecidability limitations when the systems are described by differential equations [16]. It is not always possible to generate a computational model representing all possible executions (behaviors) of a program as well as all its possible execution environments. In such cases, abstraction techniques are usually required in order to achieve the verification task [17]. The goal of the abstraction is to transform uncomputable problems to computable ones or computable problems into more efficiently computable ones. The main idea of model abstraction is to find a map between the actual set of values of state variables and a small set of abstract values such that a simulation relation exists between the original transition system and the newly created one preserving the desired properties. The model checking problem thus becomes the following: given a model M and a temporal logic property P, compute an abstraction M^* of the model and an abstraction P^* of the property and check whether $M^* \models P^*$. In the remaining, we describe several ideas concerned with using model checking techniques for the verification of AMS designs.

3.2. Relevant work

The first effort in applying model checking for electronic designs is the work in [18], where the authors proposed verification of digital designs at the transistor level. Given a circuit, they construct a finite-state discrete abstraction by partitioning the continuous state space representing the characteristics of transistors into fixed size multidimensional cubes. Heuristics methods are then used to predict possible transitions between these cubes. The final constructed model is then encoded into an automata that is verified subsequently against some properties using conventional model checking techniques.

In a series of papers [19–21], the authors tried to overcome the expensive computational method in [18], by using discretization and projection techniques of the state space into category of geometric polygons called projectahedra (projected polyhedra) [22]. Such models have the property of reducing the dimension of the state space, while maintaining an over-approximation of the dynamic behavior of the design. While this method results in less precise analysis due to projection, it still allows sound verification. Variant approaches of polyhedral based analysis were adapted in [23,24].

In [23], the authors used techniques developed for hybrid system verification to verify AMS designs. For systems described using differential equations, they use the tool d/dt [25] to over-approximate the reachability analysis. In order to tackle the state explosion problem for the class of discrete-time AMS designs, they proposed to use techniques from optimal control (i.e., hybrid constrained optimization) in order to find bounds of the reachability. The idea is to reformulate bounded time reachability analysis as a hybrid constrained based optimization problem that can be solved by techniques such as mixed-integer linear programming (MILP) [26]. The basic idea is to compute a set of worst case trajectories which implies the safety of all other trajectories.

In [24], the authors use the Checkmate tool [27] for the verification of AMS designs. The tool is based on constructing abstractions of the continuous dynamics, using flow pipes approximations, which are sequences of polyhedra that follow the natural contour of the vector field. Therefore, the state space is partitioned along the waveforms that the system can generate for the given set of initial conditions and there is no need to discretize the entire state space. Checkmate specifications to be verified can

be provided as ACTL formulas. For the verification of systems like Δ-Σ modulator, which is described by discrete-time components, a modification of the tool to support discrete-time analysis was proposed [24].

The work in [24] has been extended further in [28] for the PHAver tool [29]. In this work, the authors proposed a refinement process for the state space, which is carried out using iterations between forward and backward reachability. Such technique as claimed in [28] allows generating more precise bounds for the reachable states.

In [30], the authors proposed a more precise approximation for the reachable states of the AMS designs, where state space exploration algorithms are handled with Taylor approximations over interval domains. Such modeling allows the computation over continuous quantities while avoiding the unsoundness inherent in the numerical Taylor approximation. They use the techniques along with symbolic manipulation analysis for the bounded model checking of safety and liveness properties. The approach is applied for both discrete- and continuous-time AMS designs.

In [31], the authors proposed modeling analog designs using timed hybrid Petri nets (THPN), which is an extension of Petri nets for real-time and hybrid systems. They proposed two methods for the generation of the THPNs verification model. In the first method, they translate the circuits differential equation into THPNs. This is done by first discretizing the state space as in [32,33] and then encoding the state space into THPNs. Additionally, they developed an algorithm in [34], to generate THPNs from simulation data. Over-approximation based analysis is applied on the generated model. In [35], the authors compared verification using their methodology in [31] against simulation results, by examining the effect of variable delays caused by parasitic capacitances and interconnect capacitances on the performance and functionality of the circuits. In [36], they enhanced their methodology in [31] by using a variant of Petri nets named labeled hybrid Petri nets (LHPNs), that offer a more efficient representation. BDD based symbolic algorithms and satisfiability modulo theories (SMT) [37] techniques are then applied in [38,39] to check for properties of the design.

In contrast to the on-the-fly techniques mentioned above, a priori state space division has been explored as a way to obtain abstractions of the analog behavior of the systems. In [32,33], the authors proposed to use an automatic state space subdivision method, by discretizing the whole continuous state space into variable sized regions where each of these regions represents a homogeneous part of the state space and is treated as a discrete state of the simplified system. Some kind of estimation techniques are then proposed to describe possible transitions between partitions under the condition of retaining the essential non-linear behavior of the analog system. Different criteria take care of the resulting error during discretization and try to automatically minimize the error by choosing a suitable subdivision of the state space. The discretized state space is then encoded and CTL based model checking is applied. The proposed approach was implemented in a tool called *Amcheck* [9]. In [40], the authors proposed extending their previous work for the verification of time constraints of analog signals like rise and fall time. The presented extensions are based on developing a specification language ASL [41] tailored to represent properties of interest in analog circuit design, such as offset, gain, rise time, and slew rate.

In [42], the authors developed a bounded model checking tool (*Property-Checker*) for the verification of the quasi-static behavior of AMS designs. The basic idea is based on validity checking of first order formulas over a finite interval of time steps using SAT-modulo theories. In contrast to other approaches, the work presented in [42] trades-off accuracy with efficiency by basing the analysis on rational numbers rather than real numbers.

3.3. Discussions and perspectives

As a summary, we can classify model checking and reachability analysis into three categories:

- On-the-fly model checking: Where the set of reachable states corresponds to the over-approximate solution of the system equations, which is obtained for a bounded period of time.
- Abstract model checking: The whole state space is subdivided into regions and then heuristic rules define the transitions between states. Conventional model-checking algorithms are applied on the new abstract model of the system, which is generally described as a finite-state automaton.
- Model checking as a symbolic decision procedure: The verification problem is turned into a satisfiability or constraint solving problem that is defined for bounded steps.

In the first two types, the generated state space can be encoded symbolically such that discrete model-checking algorithms can be applied. Time bounded ACTL properties are usually the property language of choice. One drawback of abstraction is its over-approximation of behavior. If more behaviors are added and a universally quantified property is *True*, then it will still be *True* on a concrete system. This method works well with safety properties (e.g., *something bad will never happen*) but often leads to false negatives when reasoning about liveness properties (e.g., *eventually, something good will happen*). The consequence of the over-approximation of the possible executions is that non-existing executions are considered, some of which are erroneous, which leads to false alarms (also called false positives). A false alarm corresponds to the case when the abstract semantics intersects the forbidden zone while the concrete semantics does not intersect this forbidden zone. So a potential error is signaled which can never occur in reality. The imprecision problem can often be solved by choosing a more refined abstract model, which is more precise and so often more complex, which results in larger computation costs. The approach used in [42], while it avoids the over-approximation issue, is limited to simplified models of AMS design. In fact, the approach does not support systems described using differential equations, however, it is more suitable for systems described using difference equations. Tables 2 gives a comparison between the work presented in this section. They describe the class of system verified, the models used, the analysis regions and domains, the adopted analysis and state space partitioning techniques, the tools used, and the case studies verified.

Other successful trends in the related hybrid system theory that can be explored in the model checking of AMS is the use of sound methods like constraint solving techniques and logical based methods like quantifier elimination. In addition, model checking in the frequency domain has yet to be developed.

4. Run-time verification

4.1. Introduction

Model checking of AMS circuits is computationally expensive and therefore suffers from the state-space explosion problem that makes exhaustive verification very hard and poses limitations in memory and/or time resources. In order to cope with these problems, *run-time verification* (logical monitoring) methods were developed where no computational model is needed prior to the verification, avoiding state space explosion [43]. By employing logical monitors, an efficient analysis of the results is achieved,

Table 2
Model checking techniques

	[18]	[19,20,22]	[24]	[23]
Type of systems	Non-linear	Non-linear	Non-linear	Non-linear
Models	ODE	ODE	HA/ ODE–DAE	HA/ODE–DAE
Analysis regions	No restriction	No restriction	No restriction	No restriction
Analysis domain	Time	Time	Time	Time
Techniques and analysis	Simulation lang. containment	Projection numerical appro.	Numerical approx.	Numerical approx., MILP
State space partitions	Fixed size hypercubes	Projectahedra	Convex polyhedra	Orthogonal polyhedra
Temporal logic	N/A	–	ACTL	–
Verification method	Abstract model checking	On-the-fly reachability	On-the-fly model checking	On-the-fly reachability
Tools	COSPAN	Matlab/Coho	Checkmate	d/dt
Case studies	Interlock circuits	Van der Pool oscillator, toggle circuit	Tunnel diode Δ-Σ mod	Low pass filter Δ-Σ mod

	[9,32,33,40,41]	[31,34–36,38,39]	[42]	[28]	[30]
Type of systems	Non-linear	Non-linear	AMS	Non-linear	Non-linear
Models	ODE, DAE	THPN/ODE	Piecewise linear automaton	Piecewise ODE	ODE/SRE
Analysis regions	No restriction	No restriction	Steady state	Steady state	No restriction
Analysis domain	Time	Time	Time	Time	Time
Techniques and analysis	Numerical analysis	Numerical approx.	Bounded MC	Numerical approx.	Numerical and symbolic approx.
State space partitions	Hypercubes	Convex polygons	–	Convex polygons	Taylor models
Temporal logic	ASL/CTL-AT	ACTL	FOL	–	LTL
Verification method	Abstract model checking	On-the-fly/symbolic model checking	Symbolic model checking	On-the-fly reachability	On-the-fly/symbolic model checking
Tools	Amcheck	ATACS/LEMA	SVC, property checker	PHAver	Mathematica
Case studies	Schmidt trigger, opamp, VCO	Tunnel diode PLL	Sequential AMS circuit	Tunnel diode VCO	Oscillators Δ-Σ mod

avoiding exhaustive inspection, by testing whether a given behavior satisfies a property [44]. Monitors for hybrid systems have been developed in [44], where the authors developed tools for monitoring real-time and hybrid systems. Timed and linear hybrid automata can be used to monitor real-time and hybrid behavior, respectively.

Property monitoring of AMS designs is performed in general using assertions and tests. The monitoring can be described in general as follows: the AMS design under verification is simulated by attaching it to a testbench which provides the inputs necessary to drive while monitoring its output. Assertions have the property that they are always checked, regardless of what tests are running. An assertion is a piece of code that continually observes one or more signals and raises a fault when it detects an error condition. They can be placed in the models or in the circuit where they check that the design is being used correctly. The monitor could be as simple as observing a current or voltage, or could be more complicated, taking several signals, processing them and then compare against the expected results.

The main challenges in this technique is the development of adequate monitors. This process can be performed in two different fashions: namely, *offline* and *online* monitoring [45]. *Offline monitoring* starts after the whole sequence is given. *Online monitoring* is interleaved with the process of reading the sequence and is similar to the way the sequence is read by an automaton. The two types of monitoring have their strengthes and weaknesses. Offline monitoring allows the verification of more complex properties like those described backward in time (e.g., using past operators). However, offline monitoring requires the gathering of simulation or execution data in advance which can cost lots of time and memory resources. In addition, violations are not detected as soon as they happen but only after simulation is finished. On the other hand online monitoring is more practical when simpler properties are needed to be verified and violations are identified as soon as they occur. In the following we survey the main projects concerned with monitoring AMS designs.

4.2. Relevant work

In [46], the authors proposed an offline methodology for monitoring the simulation of continuous signals described by differential equations. This work is based on extending the PSL (property specification language) [47] logic to support monitoring analog signals, by defining the syntax and semantics of metric timed linear temporal logic (MTL) [48] and extending it with predicates over reals to define the signal temporal logic (STL) [46]. STL is then synthesized into timed automata [45,49] which monitor simulation traces to check for property violation in an online fashion. The approach was implemented in [50]. No techniques for test case generation is proposed.

A different effort for using PSL properties to monitor AMS designs was proposed in [51], where the authors generated observers from PSL properties to monitor the behavior of discrete-time designs. While the approach limits the classes of circuits to be verified, it has the advantage of using the standard PSL languages making it attractive to be incorporated in the design flow.

In [52,53], the authors use an extended temporal logic, AnaCTL (CTL for analog circuit verification), for monitoring the transient behavior of non-linear analog circuits. The transient response of a circuit under all possible input waveforms is represented as an FSM created by means of repeated SPICE simulations, bounding and discretizing the continuous state space of an analog circuit. Exhaustive simulation is again a drawback as the created FSM is not guaranteed to cover the total transient behavior leading to soundness problem.

An online monitoring technique was proposed in [54], where the authors used linear hybrid automata as template monitors for

Table 3
Run-time verification techniques

	[45,46,49,50]	[52]	[54]	[55]	[51]
Type of systems	Non-linear	Non-linear	Piecewise affine	Non-linear	Non-linear
Models	ODE	ODE	ODE	ODE	SRE
Monitors	STL	Ana CTL	Linear HA	TCTL, timed automata	PSL observers
Monitoring type	Offline/online	Offline	Online	Online	Offline
Analysis	No	Transient	No	No	No
Regions	Restriction	Response	Restriction	Restriction	Restriction
Analysis domain	Time	Time	Time	Time	Time
Techniques	Numerical simulation	Numerical simulation	Numerical approx.	Numerical approx.	Numerical simulation
Tools	AMT & Matlab	Spice simulator	PHAver	AWA	Matlab
Case studies	Sine wave signals, memory	VCO opamp	Tunnel diode circuit	Tunnel diode circuit	PLL Δ-Σ mod

Table 4
Theorem proving

	[56]	[57–59]	[60]
Type of systems	Piecewise linear	Piecewise linear	Piecewise non-linear
Modeling	Set of predicates over real	Set of predicates over real	Generalized recurrence equations
Domain analysis	Time	Time	Time
Verification method	Deduction	Deduction and constraint solving	Rewriting, induction constraint solving
Tool	PVS	N/A	Mathematica
Case studies	Analog receiver transmitter	TTL	DSM modulators

time domain features of oscillatory behaviors, such as bounds on signal amplitude and jitter. For the automata with an error state, the reachability computation can be stopped as soon as this state is reachable. The monitors are used within the PHAver tool where non-linear circuit equations are modeled with piecewise affine differential inclusions.

In [55], the authors propose an online monitoring methodology for analog systems. They present a run-time verification methodology based on monitoring the behavior (solution flow) of analog circuits validated using interval analysis. Given the system description and its specification described by non-linear differential equations and timed computational temporal logic (T-CTL) formulas, respectively, the authors build a timed automata monitor which can detect bad behavior within a specified time period of the interval arithmetics simulation.

4.3. Discussion and perspectives

Run-time verification, although considered only a partial verification technique, combines desirable properties from simulation and formal verification while avoiding the undesirable ones. No computational model needs to be generated prior to the verification, avoiding state space explosion. By employing logical monitors, an efficient analysis of the results is achieved, avoiding exhaustive inspection by engineers. In addition, guiding the testcase generation by the monitors, enhances the code coverage.

Although appealing, several issues must be addressed to make run-time verification useful. Among these issues are monitoring properties in frequency domain, synthesizing monitors from the specification as well as developing testcase generation approaches to guide the verification. An important requirement in monitoring analog behavior is the specification of tolerance or bounds which may be needed when comparing analog signals. A failure occurs if the comparison finds that the actual and expected results are different or different beyond a certain tolerance. Table 3 summarizes the main characteristics of the described projects. It describes the class of systems verified, the models used, the monitors language, the monitoring methods, analysis regions and domains, the adopted analysis techniques, the tools used, and the case studies verified.

5. Proof based and symbolic methods

5.1. Introduction

Theorem provers are formal systems that were developed to prove design properties using formal deduction based on a set of inference rules [6]. Even though these deductive methods are not constrained by any decidability frontiers, their application requires expertise and significant human intervention which makes their application to complex systems very difficult. A lot of research has been focusing on extending theorem provers with decision procedures for verification assistance and automation, as well as formalizing important theories like the real analysis theory. Some primary efforts on verifying AMS systems using theorem provers started recently. In addition to deductive based methods, induction and symbolic based methods were also proposed to prove properties of some classes of AMS designs.

5.2. Relevant work

In [56], the authors used the PVS theorem prover to formally prove the functional equivalence between behavioral specification of VHDL-AMS designs and approximated linearized models of their synthesized netlist. The verification was applied for DC and small signal analysis. The ideas presented can be considered as a starting point for a methodology to verify analog designs, yet important extensions should be studied more, like avoiding informal linearization, in addition to tackling more complex verification issues especially related to AC analysis.

Similar but more elaborate research was done in [57]. The author proposed an approach for specifying and reasoning about implementations of digital systems that are described at the analog level of abstraction. The approach relies upon specifying the behaviors of analog components (such as transistors) by

conservative approximation techniques based on piecewise-linear predicates on voltages and currents. Theorem proving was initially used to check for the implication relation between the implementation and the specification [58]. In order to automate the verification process, the author proposed afterwards using constraint based techniques instead [59].

In [60], the authors propose a new symbolic verification methodology for proving the properties of AMS designs using Mathematica. Starting with an AMS description and a set of constraint properties described in the form of generalized recurrence equations, an induction based verification strategy is applied to prove the correctness of the properties. The procedure is iterative in the sense that if the proof is obtained, then the property is verified. Otherwise, generated counterexamples are analyzed and constraint refinement is applied and the verification is repeated until the property is verified or a concrete counterexample is identified. Such methodology is suitable for AMS systems that can be described using discrete-time models.

5.3. Discussion and perspectives

Proof based methods for AMS designs are still premature and the proposed techniques have been only explored for the verification of basic properties. Challenging verification issues including analysis in time and frequency domains, AC analysis, error analysis are yet to be done. One main strength of such techniques is the possibility of the integration of theorem provers and computer algebra which can help establish new verification techniques for AMS designs and overcome limitation of the approximated methods like model checking and compliance verification. In Table 4, we highlight some main points of the work surveyed. It describes the class of system verified, the models used, the analysis domains, the adopted analysis techniques, the tools used, and the case studies verified.

6. Conclusion

In summary, the AMS design process must ensure, with a high degree of confidence, the proper functionality in all possible situations and to be able to meet its performance requirements. Therefore, precise constraints and properties identification along with verification from the behavioral level through functional and circuit levels are needed. This motivates the necessity of using formal verification methodologies throughout the design process.

In this paper, we summarized the research activities in the application of formal methods for the verification of AMS systems. We tried to be as exhaustive as possible in collecting the different related work as well as giving comparisons among the research proposed.

The formal verification of AMS designs is a relatively young research field and still under-developed, which is a bad and a good sign at the same time. It is bad because this shows the lack of extensive research which is due mainly to the complexity of the verification process and the challenging problems mostly inherited from the hybrid systems. Also, it is due to the different scientific backgrounds between AMS engineers, control engineers and computer scientists. However, this can motivate interdisciplinary collaborations. The good news is that room for exploration is yet wide open. Among the interesting direction is developing an AMS theory with high-order logic, process algebraic languages for AMS designs and formalizing the AMS theory within a formal theory like abstract interpretation, and developing specification logics for frequency properties among others. Another important direction is incorporating formal verification within the design flow, hence complementing simulation, testing and symbolic analysis. Also, the problem of extending classical temporal logics to derive suitable descriptions of analog properties is of great interest.

As the field of research did not reach the maturity phase yet, standard aspects for comparisons of the various projects are not well defined and there is a lack of a coherent framework and criteria that allows a theoretical analysis and comparison of the methods. We made some efforts in this direction by categorizing and comparing the available state of art projects in several aspects we believe are important to identify the qualitative strengths and weaknesses of each project.

One drawback of our comparison is the lack of testing of the several approaches. This is due to different reasons. First the public unavailability of the prototypes developed in the various projects. Second the lack of benchmarks required for comparison. We hope that in the future, we overcome these two obstacles so that more insights can be gained about the available methodologies for AMS formal verification.

References

[1] K. Kundert, H. Chang, D. Jefferies, G. Lamant, E. Malavasi, F. Sendig, Design of mixed-signal systems-on-a-chip, IEEE Trans. Computer-Aided Design Integrated Circuits Syst. 19 (12) (2000) 1561–1571.

[2] F. Pecheux, C. Lallement, A. Vachoux, VHDL-AMS and Verilog-AMS as alternative hardware description languages for efficient modeling of multi-discipline systems, IEEE Trans. Computer-Aided Design Integrated Circuits Syst. 24 (2) (2005) 204–225.

[3] VHDL-AMS Language Reference Manual, 2004 ⟨http://www.eda.org/vhdl-ams/⟩.

[4] Verilog-AMS Language Reference Manual, 2004. Available from: ⟨http://www.accellera.org⟩.

[5] R.A. Rutenbar, G.G. Gielen, B.A. Antao, Computer-Aided Design of Analog Integrated Circuits and Systems, IEEE Press, New York, 2002.

[6] T. Kropf, Introduction to Formal Hardware Verification, Springer, Berlin, 2000.

[7] R. Alur, T.A. Henzinger, P.-H. Ho, Automatic symbolic verification of embedded systems, IEEE Trans. Software Eng. 22 (3) (1996) 181–201.

[8] E.M. Clarke, O. Grumberg, D.A. Peled, Model Checking, MIT Press, Cambridge, MA, 2000.

[9] W. Hartong, R. Klausen, L. Hedrich, Formal Verification for Nonlinear Analog Systems: Approaches to Model and Equivalence Checking, Advanced Formal Verification, Kluwer, Dordrecht, 2004, pp. 205–245.

[10] A. Balivada, Y.V. Hoskote, J.A. Abraham, Verification of transient response of linear analog circuits, in: IEEE VLSI Test Symposium, 1995, pp. 42–47.

[11] S. Seshadri, J.A. Abraham, Frequency response verification of analog circuits using global optimization techniques, J. Electron. Test. 17 (5) (2001) 395–408.

[12] L. Hedrich, E. Barke, A formal approach to verification of linear analog circuits with parameter tolerances, in: IEEE/ACM Design, Automation and Test in Europe, 1998, pp. 649–654.

[13] L. Hedrich, E. Barke, A formal approach to nonlinear analog circuit verification, in: IEEE/ACM International Conference on Computer Aided Design, 1995, pp. 123–127.

[14] A. Salem, Semi-formal verification of VHDL-AMS descriptions, in: IEEE International Symposium on Circuits and Systems, 2002, pp. 333–336.

[15] T.A. Henzinger, P.-H. Ho, H. Wong-Toi, HyTech: a model checker for hybrid systems, Software Tools Technol. Transfer 1 (1–2) (1997) 110–122.

[16] P. Kopke, T. Henzinger, A. Puri, P. Varaiya, What's decidable about hybrid Automata?, in: ACM Symposium on Theory of Computing, 1995, pp. 372–382.

[17] R.P. Kurshan, Computer-Aided Verification of Coordinating Processes: The Automata-Theoretic Approach, Princeton University Press, Princeton, NJ, 1995.

[18] R.P. Kurshan, K.L. McMillan, Analysis of digital circuits through symbolic reduction, IEEE Trans. Computer-Aided Design 10 (11) (1991) 1350–1371.

[19] M.R. Greenstreet, Verifying safety properties of differential equations, in: Computer Aided Verification, Lecture Notes in Computer Science, vol. 1102, Springer, Berlin, 1996, pp. 277–287.

[20] M.R. Greenstreet, I. Mitchell, Integrating projections, in: Hybrid Systems: Computation and Control, Lecture Notes in Computer Science, vol. 1386, Springer, Berlin, 1998, pp. 159–174.

[21] C. Yan, M. Greenstreet, Circuit-level verification of a high-speed toggle, in: IEEE International Conference on Formal Methods in Computer-Aided Design, 2007, pp. 199–206.

[22] M.R. Greenstreet, I. Mitchell, Reachability analysis using polygonal projections, in: Hybrid Systems: Computation and Control, Lecture Notes in Computer Science, vol. 1569, Springer, Berlin, 1999, pp. 103–116.

[23] T. Dang, A. Donze, O. Maler, Verification of analog and mixed-signal circuits using hybrid system techniques, in: Formal Methods in Computer-Aided Design, Lecture Notes in Computer Science, vol. 3312, Springer, Berlin, 2004, pp. 14–17.

[24] S. Gupta, B.H. Krogh, R.A. Rutenbar, Towards formal verification of analog designs, in: IEEE/ACM International Conference on Computer Aided Design, 2004, pp. 210–217.
[25] E. Asarin, T. Dang, O. Maler, The d/dt tool for verification of hybrid systems, in: Computer Aided Verification, Lecture Notes in Computer Science, vol. 2404, Springer, Berlin, 2002, pp. 365–370.
[26] A. Bemporad, M. Morari, Verification of hybrid systems via mathematical programming, in: Hybrid Systems: Computation and Control, Lecture Notes in Computer Science, vol. 1569, Springer, Berlin, 1999, pp. 31–45.
[27] A. Chutinan, B.H. Krogh, Computational techniques for hybrid system verification, IEEE Trans. Autom. Control 48 (1) (2003) 64–75.
[28] G. Frehse, B.H. Krogh, R.A. Rutenbar, Verifying analog oscillator circuits using forward/backward abstraction refinement, in: IEEE/ACM Design, Automation and Test in Europe, 2006, pp. 257–262.
[29] G. Frehse, PHAVer: algorithmic verification of hybrid systems past HyTech, in: Hybrid Systems: Computation and Control, Lecture Notes in Computer Science, vol. 3414, Springer, Berlin, 2005, pp. 258–273.
[30] M. Zaki, G. Al Sammane, S. Tahar, G. Bois, Combining symbolic simulation and interval arithmetic for the verification of AMS designs, in: IEEE International Conference on Formal Methods in Computer-Aided Design, 2007, pp. 207–215.
[31] S. Little, D. Walter, N. Seegmiller, C.J. Myers, T. Yoneda, Verification of analog and mixed-signal circuits using timed hybrid Petri nets, in: Automated Technology for Verification and Analysis, Lecture Notes in Computer Science, vol. 3299, Springer, Berlin, 2004, pp. 426–440.
[32] W. Hartong, L. Hedrich, E. Barke, Model checking algorithms for analog verification, in: ACM/IEEE Design Automation Conference, 2002, pp. 542–547.
[33] W. Hartong, L. Hedrich, E. Barke, On discrete modelling and model checking for nonlinear analog systems, in: Computer Aided Verification, Lecture Notes in Computer Science, vol. 2404, Springer, Berlin, 2002, pp. 401–413.
[34] S. Little, D. Walter, K. Jones, C.J. Myers, Analog/mixed-signal circuit verification using models generated from simulation traces, in: Automated Technology for Verification and Analysis, Lecture Notes in Computer Science, vol. 4762, Springer, Berlin, 2007, pp. 114–128.
[35] C.J. Myers, R.R. Harrison, D. Walter, N. Seegmiller, S. Little, The case for analog circuit verification, Electron. Notes Theor. Comput. Sci. 153 (3) (2006) 53–63.
[36] S. Little, N. Seegmiller, D. Walter, C. Myers, T. Yoneda, Verification of analog/mixed-signal circuits using labeled hybrid Petri nets, in: International Conference on Computer-Aided Design, 2006, pp. 275–282.
[37] L. Mendonça de Moura, B. Dutertre, N. Shankar, A tutorial on satisfiability modulo theories, in: Computer Aided Verification, Lecture Notes in Computer Science, vol. 4590, Springer, Berlin, 2007, pp. 20–36.
[38] D. Walter, S. Little, N. Seegmiller, C. Myers, T. Yoneda, Symbolic model checking of analog/mixed-signal circuits, in: IEEE Asia and South Pacific Design Automation Conference, 2007, pp. 316–323.
[39] D. Walter, S. Little, C. Myers, Bounded model checking of analog and mixed-signal circuits using an SMT solver, in: Automated Technology for Verification and Analysis, Lecture Notes in Computer Science, vol. 4762, Springer, Berlin, 2007, pp. 66–81.
[40] D. Grabowski, D. Platte, L. Hedrich, E. Barke, Time constrained verification of analog circuits using model-checking algorithms, Electron. Notes Theor. Comput. Sci. 153 (3) (2006) 37–52.
[41] S. Steinhorst, A. Jesser, L. Hedrich, Advanced property specification for model checking of analog systems, in: Analog'06, 2006, pp. 63–68.
[42] M. Freibothe, J. Schönherr, B. Straube, Formal verification of the quasi-static behavior of mixed-signal circuits by property checking, Electron. Notes Theor. Comput. Sci. 153 (3) (2006) 23–35.
[43] J. Yuan, C. Pixley, A. Aziz, Constraint-Based Verification, Springer, Berlin, 2006.
[44] L. Tan, J. Kim, I. Lee, Testing and monitoring model-based generated program, Electron. Notes Theor. Comput. Sci. 89 (2) (2003) 128–148.
[45] O. Maler, D. Nickovic, A. Pnueli, Real-time temporal logic: past, present, future, in: Formal Modelling and Analysis of Timed Systems, Lecture Notes in Computer Science, vol. 3829, Springer, Berlin, 2005, pp. 2–16.
[46] O. Maler, D. Nickovic, Monitoring temporal properties of continuous signals, in: Formal Modelling and Analysis of Timed Systems, Lecture Notes in Computer Science, vol. 3253, Springer, Berlin, 2004, pp. 152–166.
[47] Accellera Property Specification Language Reference Manual, 2004. Available from: ⟨http://www.accellera.org⟩.
[48] P. Thati, G. Rosu, Monitoring algorithms for metric temporal logic specifications, Electron. Notes Theor. Comput. Sci. 113 (2005) 145–162.
[49] O. Maler, D. Nickovic, A. Pnueli, From MITL to timed automata, in: Formal Modelling and Analysis of Timed Systems, Lecture Notes in Computer Science, vol. 4202, Springer, Berlin, 2006, pp. 274–289.
[50] D. Nickovic, O. Maler, AMT: a property-based monitoring tool for analog systems, in: Formal Modelling and Analysis of Timed Systems, Austria, Lecture Notes in Computer Science, vol. 4763, Springer, Berlin, 2007, pp. 304–319.
[51] G. Al Sammane, M. Zaki, Z.J. Dong, S. Tahar, Towards assertion based verification of analog and mixed signal designs using PSL, in: Languages for Formal Specification and Verification, Forum on Specification and Design Languages, 2007.
[52] T.R. Dastidar, P.P. Chakrabarti, Verification system for transient response of analog circuits using model checking, in: IEEE International Conference on VLSI, 2005, pp. 195–200.
[53] T.R. Dastidar, P.P. Chakrabarti, A verification system for transient response of analog circuits, ACM Trans. Design Autom. Electron. Syst. 12 (3) (2007) 1–39.
[54] G. Frehse, B. Krogh, R. Rutenbar, O. Maler, Time domain verification of oscillator circuit properties, Electron. Notes Theor. Comput. Sci. 153 (3) (2006) 9–22.
[55] M.H. Zaki, S. Tahar, G. Bois, A practical approach for monitoring analog circuits, in: IEEE/ACM Great Lakes Symposium on VLSI, 2006, pp. 330–335.
[56] A. Ghosh, R. Vemuri, Formal verification of synthesized analog circuits, in: ACM/IEEE International Conference on Computer Design, 1999, pp. 40–45.
[57] K. Hanna, Reasoning about analog-level implementations of digital systems, Formal Methods Syst. Design 16 (2) (2000) 127–158.
[58] K. Hanna, Reasoning about real circuits, in: Theorem Proving in Higher Order Logics, Lecture Notes in Computer Science, vol. 859, Springer, Berlin, 1994, pp. 235–253.
[59] K. Hanna, Automatic verification of mixed-level logic circuits, in: IEEE International Conference on Formal Methods in Computer-Aided Design, Lecture Notes in Computer Science, vol. 1522, Springer, Berlin, 1998, pp. 133–166.
[60] G. Al Sammane, M. Zaki, S. Tahar, A symbolic methodology for the verification of analog and mixed signal designs, in: IEEE/ACM Design Automation and Test in Europe, 2007, pp. 249–254.