Hindawi Publishing Corporation
VLSI Design
Volume 2013, Article ID 239064, 14 pages
http://dx.doi.org/10.1155/2013/239064

Research Article

Verification of Mixed-Signal Systems with Affine Arithmetic Assertions

Carna Radojicic,[1] Christoph Grimm,[1] Florian Schupfer,[2] and Michael Rathmair[2]

[1] *Design of Cyber-Physical Systems, Kaiserslautern University of Technology, Postfach 3049, 67663 Kaiserslautern, Germany*
[2] *Institute of Computer Technology, Vienna University of Technology, Gusshausstraße 27-29, 1040 Vienna, Austria*

Correspondence should be addressed to Carna Radojicic; radojicic@cs.uni-kl.de

Received 3 January 2013; Revised 26 March 2013; Accepted 28 March 2013

Academic Editor: Chang-Ho Lee

Embedded systems include an increasing share of analog/mixed-signal components that are tightly interwoven with functionality of digital HW/SW systems. A challenge for verification is that even small deviations in analog components can lead to significant changes in system properties. In this paper we propose the combination of range-based, semisymbolic simulation with assertion checking. We show that this approach combines advantages, but as well some limitations, of multirun simulations with formal techniques. The efficiency of the proposed method is demonstrated by several examples.

1. Introduction

Analog/mixed-signal (AMS) systems are a crucial part of today's embedded systems. Typical AMS components such as sensors, transceivers, and signal conditioning enable interaction of embedded HW/SW systems with its physical environment. In today's embedded systems, the functionality of the analog components is tightly interwoven with the digital HW/SW system. A particular challenge of AMS systems is that parameters cannot be assumed to be fixed to a deterministic value like in a digital system.

Behavior of AMS systems cannot be assumed to be fixed for the following reasons: *variations* of parameters due to variations in the manufacturing process, but as well during operation (e.g., different temperatures, aging, and supply voltage) introduce deviations compared to an ideal reference. (*Modeling*) *uncertainties* are introduced by the fact that all models represent more or less accurate abstractions of physical reality. No model can be assumed to be absolutely accurate. Furthermore, computation with fixed-point arithmetic in the digital domain can contribute significantly to deviation from expected ideal behavior (rounding errors, quantization). In the following we refer to such deviations of a simulation run from possible real behavior in general as "*deviations*."

A communication system with typical variations and deviations is shown in Figure 1 as an example. Variations of gain, offset, or due to temperature (Figure 1, left) are compensated in software. This is done at lower layers of the software by controlling variable gain amplifier (VGA), voltage controlled oscillator (VCO). Higher layers of the software stack introduce further error correction mechanisms in software. Dependability of the overall system is defined by complex interaction of AMS parts with the software stack. While known statistical methods (e.g., Monte Carlo simulation) allow us computing other statistic properties like Bit or Packet Error Rates (BER, PER), open issues are questions such as

(i) how can we *guarantee* some system properties, for example, for safety relevant systems?

(ii) can we get information from the analysis that assists us in design and debug, such as counter examples?

This paper proposes a new methodology that for the first time combines high verification coverage of formal verification on one hand with the general applicability of simulation-based approaches on the other hand. To achieve this goal, we combine assertion checking with symbolic simulation:

(1) assertions specify required properties of a system;

(2) in an overall system model, deviations and variations are represented by symbols that capture size and correlations of the deviations, variations, respectively;

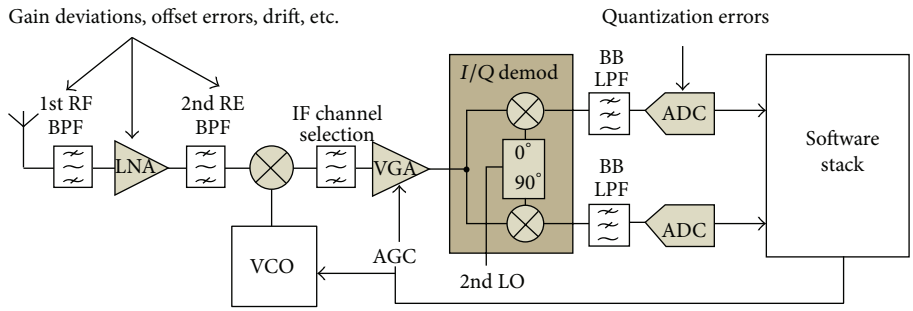


Figure 1: System model including parameter deviations.

(3) for verification of "worst case" behavior, a range-based simulation using Affine Arithmetic shows that for given ranges of inputs and deviations the required properties are valid, and that no "forbidden state" is reached.

We mostly focus on level of block diagrams, but the methodology is as well applicable on circuit simulation. We implemented it based on SystemC and its AMS extensions. Section 2 gives a review of state of the art and related work. In Section 3 Affine Arithmetic is described, and some modeling examples are given. Section 4 describes semisymbolic simulation and the verification method proposed in this work. The applicability of the verification technology is demonstrated by examples given in Section 5. Section 6 concludes the paper and identifies future work.

2. State of Art and Related Work

When verifying AMS systems with parameter deviations, application of multirun simulation techniques (Monte-Carlo, worst-case analysis) can be considered as state-of-the art. *Monte Carlo simulation* [1] is a statistical technique. However, statistical techniques do not provide dependable "worst case" results. While the number of simulation runs can be reduced by *importance sampling* [2], the number of simulation runs required may still be prohibitive for analysis of complex systems. *Corner case analysis* [3] is a more appropriate means for finding worst case performances of AMS systems. Unfortunately, the number of simulation runs grows exponentially with the number of parameters considered. However, even if all corner cases are considered, the dependability of the result cannot be guaranteed since corner cases are not necessarily worst cases. *Design of Experiments* [4] allows reducing the number of simulation runs significantly and finding worst case performances more accurately.

Even with high number of simulation runs, there is no guarantee that worst case performances are found. A drawback of multi-run simulation methods is that the dependable operation of AMS systems cannot be guaranteed under all circumstances. For safety-critical systems, for example, in aviation or automotive systems this is a major drawback and motivation for further research.

In order to find counter examples, rapidly exploring random trees [5, 6] and robust test case generation have been proposed in [7–9]. Further, the simulation techniques proposed in [10, 11] guarantee that a system is "safe" if a set of trajectories lie within certain regions defined by previously found conditions. In contrast to these approaches the techniques proposed in [12, 13] compute an overapproximation of the set of states reached by all trajectories. While these methods support debugging and introduce coverage metrics, they are not able to deal with the increased complexity of systems that with deviations and variations. An approach that enables safety verification of hybrid systems with uncertain parameters is the use of barrier certificates proposed in [14]. Those methods can verify if a set of system trajectories crosses a barrier previously defined by a barrier certificate. Finding a proper barrier certificate is not easy and makes this approach difficult for system verification.

To cope with the drawbacks of simulation-based techniques, formal verification methods were proposed. The idea of formal methods is to use the formal checkers which automatically explore all possible states and transitions in the system model to check if the desired output behaviour is met or not. Hence, in contrast to simulation-based methods which can verify only one behaviour (for only one input stimuli) per operation, the formal methods deal with the set of behaviours at a time. Approaches for formal verification were firstly applied on digital systems [15–17], and due to their efficiency they found a good way in industrial applications. Approaches that also cover analog/mixed-signal or hybrid systems are rare and still in infancy.

In [18–20] hybrid systems with linear and nonlinear dynamics are approximated by timed automata in order to simplify their analysis. Reference [21] describes a model checking tool which requires discrete and (for continuous parts) linear system descriptions. For nonlinear continuous behavior, such approximations are too simple. Linear phase-portrait approximation [22] is a general technique, because its approximation does not depend on the order of the differential equation to be approximated. There is no standard method for partitioning the state space, and therefore it seems to be complicated to find proper discrete models for strongly nonlinear models. In [23–25] focus is on nonlinear analog behavior for which discretized models in the state space are used for verification. The efficiency of these approaches seems to be limited to smaller analog systems. With increasing complexity, the number of states in the discretized model grows which leads to the state explosion problem and high run time of verification algorithms applied on this model.

Table 1: Tradeoffs of verification methods for AMS systems with deviations.

Verification techniques	Disadvantages	Advantages
Multirun simulations	Low verification coverage, no "guarantee"	General applicability
Formal methods	State explosion problem, no complex, heterogeneous systems	High verification coverage "guarantee," counter examples
Assertion-based techniques (MSA, AMT tool)	Only nominal behaviour verified, no "guarantee"	Well suited for complex systems, increased coverage

Due to limitation of formal methods for analog/continuous systems to small systems, simulation-based techniques are still the only way for verification of more complex analog/continuous and AMS systems. To formalize verification of AMS systems, assertions that describe typical properties of analog systems are the focus of recent research. In [26] mixed-signal assertions (MSA) were proposed to check properties of mixed-signal systems during simulation. These assertions were implemented in a separate SCAC (SystemC AMS Temporal Checker) library. This library is easily integrated in SystemC AMS simulation environment, due to its C++ based nature. In contrast to this approach, [27] features AMT, an offline tool for monitoring temporal properties of mixed-signal systems for verification. Both verification methodologies simulate and evaluate a nominal system model without taking into account any deviations caused by variations in design process.

A particular challenge in design of complex analog/mixed-signal systems is parameter variations. In any physical system, values are not implemented in an accurate way and change in a partially unpredictable way over time (e.g., due to temperature, aging, etc.). Such deviations form an ideal model change system behavior and can potentially cause malfunctions. For conventional simulation, multi-run methods as described in the first paragraph do not provide the result dependability and require a high number of simulation runs to explore the system behaviour while considering process variations. For formal approaches, such issues are still in infancy, because its applicability simply does not yet allow handling complex and heterogeneous systems such as AMS systems.

A first approach to cover deviation effects in AMS systems and compute the guaranteed worst case results at the same time was introduced in [28–31]. Deviations are modeled as ranges, superimposed on the nominal system model, and modified during system simulation to obtain the formally guaranteed range-based system quantities. For this purpose, Affine Arithmetic was applied. Using this approach the variations in parameter values are represented with deviated symbols which are traced to the system output. Hence, the contribution of all variations in the system is contained in the system response which simplifies analysis of the system robustness.

In [32] it is proposed that how Affine Arithmetic approach can be used to analyze worst case behavior of electrical circuits. Further, in [33] this methodology found its application in sizing of analog circuits. Using Affine Arithmetic the bounds on the worst case circuit behavior are calculated and the global minimum of sizing problem is determined due to inclusion isotonicity. Beside analog domain, Affine Arithmetic models can also be used in Digital Signal Processing (DSP) applications to represent errors introduced by calculations in floating-point arithmetic [34, 35].

Within this work, semisymbolic simulation based on Affine Arithmetic is combined with the assertion-based technology. Concretely, assertions based on Affine Arithmetic (AAF+A) are introduced to include range-based system quantities and allow specification and automatic verification of typical time and frequency-domain properties of systems considering variations in their parameter values.

Using the proposed verification method system verification is done during simulation.

(i) The desired output behaviour is described with assertion which is embedded into simulation process.

(ii) The assertion is verified automatically. In the case where the design requirement is not met the simulation process is stopped reporting the user about the assertion violation.

Table 1 summarizes the advantages and disadvantages of previously described verification techniques.

The verification method proposed in this work copes with the disadvantages of previous verification methods. Concretely, combining the assertion-based technology with semisymbolic simulation, which generates the dependable guaranteed result in which all output values for the considered parameter set are contained, 100% coverage can be obtained.

3. Affine Arithmetic and Its Use for Modeling Deviations

3.1. Affine Arithmetic. Affine Arithmetic (AA) is a range arithmetic that overcomes the error explosion problem of Interval Arithmetic (IA) [36]. AA keeps track of correlations between quantities represented as ranges. This in particular enables application for simulation of control systems. A feedback loop, for instance, can be simulated keeping the correlation of identical ranges. A subtraction of related ranges therefore results in a reduced range avoiding the overapproximation inherent to Interval Arithmetic [29].

An affine expression $\tilde{x}$ can be represented as

$$\tilde{x} = x_0 + \sum_{i \in \mathcal{N}_{\tilde{x}}} x_i \varepsilon_i, \quad \varepsilon_i \in [-1, 1], \tag{1}$$

where a sum of deviation terms $\sum_{i\in\mathcal{N}_{\tilde{x}}} x_i\varepsilon_i$ models the impact of independent deviations from the ideal system behavior described with the nominal value x_0. The values of deviation symbols ε_i lie in the range $[-1, 1]$ which is scaled by the numerical value x_i. Linear mathematical operations in Affine Arithmetic allow accurate symbolic computations and are defined as follows:

$$\begin{aligned}\tilde{x} \pm \tilde{y} &= (x_0 \pm y_0) + \sum_{i\in\mathcal{N}_{\tilde{x}}} (x_i \pm y_i)\,\varepsilon_i,\\ c\tilde{x} &= cx_0 + \sum_{i\in\mathcal{N}_{\tilde{x}}} cx_i\varepsilon_i,\end{aligned} \tag{2}$$

where $\mathcal{N}_{\tilde{x}}$ defines a set of natural numbers identifying all deviation terms $x_i\varepsilon_i$ in symbol $\tilde{x}$. In contrast to linear operations, nonlinear operations introduce an overapproximation of the exact solution, for example, multiplication as follows:

$$\begin{aligned}\tilde{x}\cdot\tilde{y} :=\ & (x_0 \cdot y_0) + \sum_{i\in\mathcal{N}_{\tilde{x}}} (x_0 y_i + x_i y_0)\,\varepsilon_i\\ &+ \operatorname{rad}(\tilde{x})\cdot\operatorname{rad}(\tilde{y})\,\varepsilon_{\mathcal{N}_{\tilde{x}}+1},\end{aligned} \tag{3}$$

where $\operatorname{rad}(\tilde{x})$ is equal to $\sum_{i\in\mathcal{N}_{\tilde{x}}} |x_i|$ and represents the total deviation of $\tilde{x}$. Although the multiplication operation results in an overapproximation, deviations in the result of this operation are traced to deviations contained in the quantities $\tilde{x}$ and $\tilde{y}$. The overapproximation is contained in residual term.

3.2. Modeling Examples with Affine Arithmetic. In the following we show how to model different deviations. We focus on block-diagram like representations with transfer functions as common in control theory, because this model of computation is generally applicable to a vast set of different domains, including communication systems and electronic circuits. We focus on giving some mathematical background that can be applied in C-language (as in the examples in later sections), but as well, for example, in Matlab/Simulink.

(a) Simple Example—Modeling Gain Variation. To model a gain variation of a block (e.g., a low noise amplifier (LNA)) we assume that the exact gain value is not known but lies in interval $[K_{\min}, K_{\max}]$. The range for K $[K_{\min}, K_{\max}]$ can be modeled using Affine Arithmetic as follows:

$$K_{\text{nom}} + \varepsilon K_{\text{dev}}, \quad \varepsilon \in [-1, 1], \tag{4}$$

where K_{nom} and K_{dev} correspond to the center value of the range and the maximum absolute deviation from the center value, respectively. To model the gain variation in the SystemC AMS (used as a simulation environment in this work) that extended with an abstract data type AAF and some constructors for typical deviations, it is only necessary to call the constructor of the amplifier module with K_{nom} and K_{dev} as the second and the third argument. The first argument is always the name of the module. Therefore, the amplifier can be instantiated by the following line of code:

```
amp amp_("amp_", Knom, Kdev).
```

(b) Modeling (Parameter) Uncertainties. Modeling uncertainties are due to lack of capturing absolute accurate models of "real" behavior. In order to describe simple modeling uncertainties, a more general model is assumed in which the "real" behavior can be included by parameter variations. As simple example, the following transfer function of a system block will be supposed:

$$P(s) = \frac{1}{s^2 + as + 1}. \tag{5}$$

Further, it will be supposed that the exact value of the parameter a is not known, but it is known that it lies in interval $[a_{\min}, a_{\max}]$. The range for parameter a can be modeled using Affine Arithmetic as

$$a_{\text{nom}} + \varepsilon a_{\text{dev}}, \quad \varepsilon \in [-1, 1], \tag{6}$$

where a_{nom} represents the midpoint of the range and a_{dev} the maximum absolute deviation from the midpoint. Now the transfer function $P(s)$ can be expressed as

$$P(s) = \frac{P_{\text{nom}}(s)}{1 + \varepsilon P_{\text{dev}}(s)\,P_{\text{nom}}(s)}, \quad \varepsilon \in [-1, 1], \tag{7}$$

where $P_{\text{nom}}(s)$ represents the nominal model with the parameter value a_{nom} as follows:

$$P_{\text{nom}}(s) = \frac{1}{s^2 + a_{\text{nom}}s + 1} \tag{8}$$

and $P_{\text{dev}}(s)$ is the deviation function modeled as $P_{\text{dev}}(s) = a_{\text{dev}}s$. The system block model with parameter deviation can be represented with the block diagram shown in Figure 2.

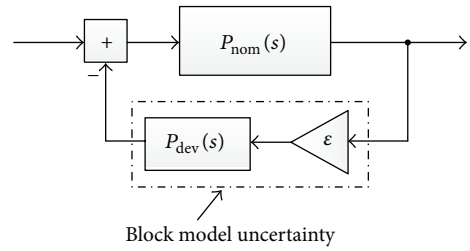


Figure 2: Block-diagram level representation of model transfer function with uncertain parameters.

(c) Variation of Time Delay, Jitter. Time delays are often varying, even in digital systems ("jitter"). A time delay can be modeled by the following transfer function:

$$P(s) = e^{-\tau s}P_{\text{nom}}(s), \tag{9}$$

where $P_{\text{nom}}(s)$ models an ideal behavior of a block (without time delay) and τ represents a time delay for which it is supposed that its exact value is not known, but it is known that it lies in interval $[0, \tau_{\max}]$. The time delay causes deviation of the block from its ideal behavior $P_{\text{nom}}(s)$. This deviation will be modeled using Affine Arithmetic. In order to do this the exponential function $e^{-\tau s}$ will be approximated using the first-order Taylor polynomial:

$$\begin{aligned}e^{-\tau s} &:= e^{-\tau_{\text{nom}}s} + \left.\frac{\left(e^{-s\tau}\right)'}{1!}\right|_{\tau=\tau_{\text{nom}}} * \varepsilon\tau_{\text{dev}}\\ &= e^{-\tau_{\text{nom}}s} + (-s) * e^{-\tau_{\text{nom}}s} * \left(\varepsilon\tau_{\text{dev}}\right),\end{aligned} \tag{10}$$

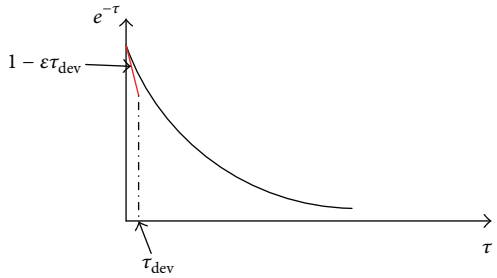


Figure 3: Linearization of exponential function $e^{-\tau}$.

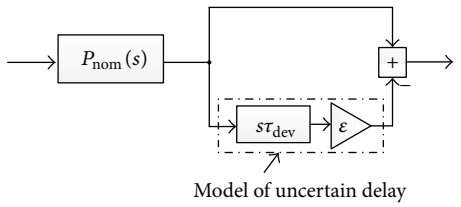


Figure 4: Block-diagram representation of model of uncertain delay respectively jitter.

where τ_{nom} represents time delay in ideal conditions whose value is zero. The symbol τ_{dev} represents the maximum absolute deviation of time delay from its nominal value, and ε is a real number whose value lies in interval $[-1, 1]$. Substituting $\tau_{\text{nom}} = 0$ in the previous equation we get

$$e^{-\tau s} := 1 + (-s) * \varepsilon\tau_{\text{dev}}, \quad \varepsilon \in [-1, 1]. \tag{11}$$

Since the approximation of the exponential function $e^{-\tau s}$ with the first-order Taylor polynomial represents the linearization of $e^{-\tau s}$ around τ_{nom}, this polynomial is actually tangent line on the function at point τ_{nom}, as it can be seen in Figure 3. Replacing $e^{-\tau s}$ in (9) with this approximation the transfer function $P(s)$ can be approximated with

$$\begin{aligned} P(s) &:= P_{\text{nom}}(s)\left(1 + (-s) * \varepsilon\tau_{\text{dev}}\right) \\ &= P_{\text{nom}}(s)\left(1 + \varepsilon P_{\text{dev}}\right), \quad \varepsilon \in [-1, 1], \end{aligned} \tag{12}$$

where $P_{\text{nom}}(s)$ models the block ideal behavior and $P_{\text{dev}}(s)$ models the deviation from the ideal behavior. The block diagram corresponding to this model is shown in Figure 4.

3.3. Abstraction of Accurate Model with Affine Arithmetic. For verification of accurate system models in the presence of parameter deviations, a high number of simulation runs is required to achieve a sufficient verification coverage. To deal with this drawback "semisymbolic" approach based on Affine Arithmetic is introduced. Using this method an abstract system model is created in which the accurate system behavior is included. Concretely, abstraction of system model gets an overapproximation of accurate models and therefore gives a guaranty that if the abstract model satisfies desired specifications, the accurate model will also meet them.

To create the abstract model it will be supposed that there is a small change of the input voltage signal v_d around DC operating point (I_D, V_D) Δv_d (see Figure 5(b)). This change will be modeled using Affine Arithmetic as follows:

$$v_d = V_D + \varepsilon\Delta v_d, \quad \varepsilon \in [-1, 1]. \tag{13}$$

The accurate model of a diode can be described with the following equation:

$$i_d = I_s\left(e^{v_d/\eta V_T} - 1\right). \tag{14}$$

To get the abstract model of a diode which is more simple for analysis, the accurate model will be approximated linearizing the nonlinear equation around DC operating point (I_D, V_D). This linearization will be performed using the first-order Taylor's series as follows:

$$\begin{aligned} i_d &= I_D + \frac{\partial i_d}{\partial v_d}(V_D)(v_d - V_D) + \text{lin\_error} \\ &= I_D + I_s e^{V_D/\eta V_T}\frac{1}{\eta V_T}\varepsilon\Delta v_d + \text{lin\_error}, \quad \varepsilon \in [-1, 1], \end{aligned} \tag{15}$$

where the symbol lin_error assigns linearization error which is added to enclose the accurate model in the abstracted one. The absolute value of this error represents the maximum absolute value of the Lagrange remainder:

$$\begin{aligned} |\text{lin\_error}| &= \max\left(\frac{1}{2}\left|\frac{\partial^2 i_d}{\partial v_d^2}(\xi)\right|(v_d - V_D)^2\right) \\ &= \max\left(\frac{1}{2}\left|\frac{\partial^2 i_d}{\partial v_d^2}(\xi)\right|\right)\left(\Delta v_d^2\right), \end{aligned} \tag{16}$$

where ξ can take any value from $\xi \in [V_D - \Delta v_d, V_D + \Delta v_d]$. To include the accurate model, linearization error will be represented as

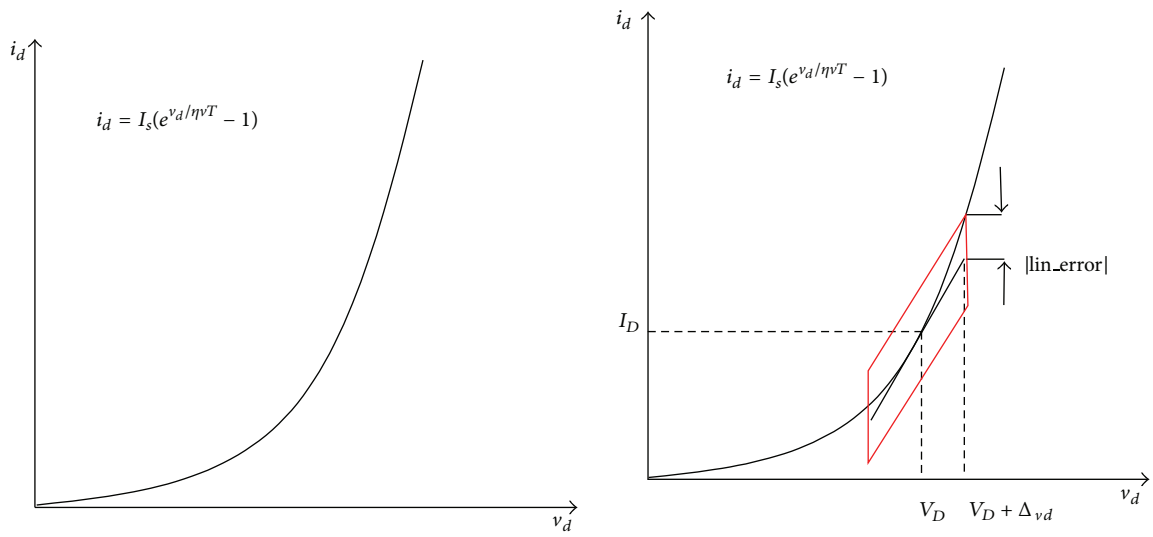
$$\text{lin\_error} = \varepsilon' |\text{lin\_error}|, \quad \varepsilon' \in [-1, 1]. \tag{17}$$

Figure 5(b) shows approximation of nonlinear diode (from Figure 5(a)) at the operating point.

3.4. Time-Domain Properties Modeled with Affine Arithmetic. To analyze system behavior it is necessary to specify values of its properties. In the following there will be given a list of properties in time, but also in frequency domain whose specified values can be modeled using Affine Arithmetic approach.

3.4.1. Settling Time Property. This time is defined as the maximum time necessary for the output signal to settle within the error band, usually symmetrical around the value of the output signal asymptote, from the time at which an ideal step input is applied. The specified values for the settling time and the error band can be modeled with Affine Arithmetic as follows:

$$\begin{aligned} \text{spec}(t_s) &= \frac{t_s}{2} + \varepsilon\frac{t_s}{2}, \quad \varepsilon \in [-1, 1], \\ \text{error\_band} &= y_{\text{asimp}} + \varepsilon\delta, \quad \varepsilon \in [-1, 1], \end{aligned} \tag{18}$$



(a) Accurate diode model

(b) Approximation of accurate model

FIGURE 5: Forward diode characteristic.

where δ represents an allowed tolerance from the asymptote y_{asimp}.

(d) Operational Range Specification. This property defines allowed swinging in the output voltage, which does not cause a system make distortions on its output. This range will be represented with Affine Arithmetic as

$$\text{spec}(\text{swing}) = SW + \varepsilon\beta \quad \varepsilon \in [-1, 1], \tag{19}$$

where SW represents the nominal value of range defining the swing property and β the allowed deviation from the nominal value.

3.5. Frequency-Domain Properties Modeled with Affine Arithmetic. The properties determining system behavior in frequency domain, whose specified values can be modeled with Affine Arithmetic, refer to low pass filter design specifications.

3.5.1. The Allowed Ripple in the Pass Band. This property defines maximum allowed deviation from the DC filter gain $S(0)$. Using Affine Arithmetic this requirement can be represented as

$$S(f) \in S(0) + \varepsilon_p \delta_p, \quad 0 \le f \le f_p, \tag{20}$$

where $\varepsilon_p \in [-1, 1]$, $S(0)$ is a DC gain in the pass band, δ_p is the maximum allowed deviation from the DC gain, and f_p is the pass band edge frequency.

3.5.2. The Allowed Ripple in the Stop Band. This design specification defines the minimum allowed attenuation in the stop band and can be modeled in the similar way as

$$S(f) \in \frac{\delta_n}{2} + \varepsilon_n \frac{\delta_n}{2}, \quad f \ge f_n, \tag{21}$$

where $\varepsilon_n \in [-1, 1]$, δ_n represents the minimum attenuation, and f_n represents the stop band edge frequency.

4. Semisymbolic Simulation and Assertion-Based Verification

In order to reduce high number of simulation runs required for simulation of systems with parameter variations, semisymbolic simulation is introduced. The idea of this approach is to model parameter deviations using Affine Arithmetic and simulate systems including these deviations. The simulation result is range-based system response which can be obtained in only one simulation run. Semisymbolic simulation can be performed on two levels: system level and circuit level.

4.1. Semisymbolic Simulation on System and Circuit Level. For semisymbolic simulations on system level SystemC AMS environment is used. Affine Arithmetic is implemented in a separate library which is due to SystemC AMS C++ based nature very easily integrated.

Beside system level analog circuits can also be simulated on lower level like transistor level. For this purpose, a semisymbolic circuit simulator has been developed [37]. Circuit simulation is performed in two steps.

(1) A netlist of a simulated system is converted into the according system of differential algebraic equations (DAE) using the Modified-Node-Analysis (MNA):

$$F\left(\underline{x}(t), \underline{x}'(t), \underline{p}(t), t\right) = \underline{0}, \tag{22}$$

where the vector $\underline{x}(t)$ represents the vector of time dependable state variables and $\underline{p}(t)$ represents the vector of time dependable circuit parameters.

Table 2: AAF+A operators.

AAF+A operators	Symbols	Operator meaning
Arithmetic operators	$\ominus \in \{+, -, *, /, \hat{}\}$	Symbols for the usual arithmetic operations
Relation operators	$\bullet \in \{<, >, \leq, \geq, ==\}$	The usual relation operations
Logic operators	$\oslash \in \{\&\&, \|\|, \rightarrow\}$!	Logic and, or, implies not
Affine analog operator	IN	Compares two affine terms a and b
Affine analog frequency operator	AFO $\in$ {FIN, GFIN} min, max	Compares two affine terms a and b within specified frequency interval, FIN assigns eventually; GIN assigns always Min finds frequency component with minimum amplitude value within specified frequency interval; max finds frequency component with maximum amplitude value within specified frequency interval
Slope operator	DV	Calculates slope of a signal
Temporal operators	$\square \in \{G, F\}$	Always, eventually
Affine analog time interval	Time $\in$ AAF, time $= t_0 + \varepsilon\Delta t, t_0, \Delta t \in \mathbb{R}$	Specified signal time interval of affine analog signal
Verification time interval	$[t], t \in \mathbb{R}$	Specified time within which the assertion is verified

(2) The equation system is passed through a numerical equation solver which performs DC, AC, and Transient Analysis [37]. The solver uses numerical methods as forward, backward Euler or trapezoidal methods to solve the equations.

Since at transistor level analog circuits are usually described with nonlinear differential algebraic equations the numerical solving is followed by linearization of equation system in the operating point with respect to variables and parameters. To deal with affine terms the algorithm [37] with the following steps is applied in the simulator.

In the first step the nominal solution is computed using Newton-Raphson method. In the second step the equation system is linearized in the operating point. Result of linearization is the linear dependency of variables according to the parameter deviations. In the case of linear system with constant parameters and variable inputs the result of linearized equation system is exact affine solution and algorithm ends. As the equation system usually contains nonlinear expressions, the affine solution of the linearized system is usually an underestimation of the exact solution, and therefore it has to be extended to include the exact area. This is done in the third step of algorithm.

4.2. Assertion-Based Verification with Affine Arithmetic. The verification technology proposed in this work is based on assertions which use Affine Arithmetic to model specified values of system properties as ranges. As simulation and verification environment SystemC AMS is used. The assertions representing specifications are verified within simulation run. In the case where the specification is violated the specification violation is reported and simulation run is stopped. To describe specifications with AAF+A, the set of operators defining the syntax of these assertions is used. Table 2 summarizes available operators whose meaning will be briefly described in the following.

Table 3: Time and frequency domain formulas (TBF and FBF).

TBF	FBF
$s \bullet d \in$ TBF	$\{\text{GFIN, FIN}\}(f_1, f_2, \text{FFT}\langle N\rangle(\beta), \gamma) \in$ FBF
s - affine signal, $d \in \mathbb{R}$	$f_1, f_2 \in \mathbb{R}$, $N \in \mathbb{N}$, β-affine signal, $\gamma \in$ AAF
$\text{DV}(s) \bullet d \in$ TBF	$\text{IN}(\zeta, \psi) \in$ FBF
s, affine signal, $d \in \mathbb{R}$	$\zeta \in$ FF, $\psi \in$ AAF
$\text{IN}\{[\text{time}]\}(\varphi, \vartheta) \in$ TBF	$c \bullet d \in$ FBF
$\varphi \in \{s, \text{DV}(s)\}$, time $\in$ AAF, $\vartheta \in$ AAF	$c \in \text{FF} \wedge d \in \mathbb{R}$

The label AAF in the table assigns the set of affine terms. The set of assertions based on Affine Arithmetic (AAF+A) is comprised of two sets. The first set defines Boolean formulas checking validation of properties in time-domain TBF and the second one in frequency domain FBF. The operators from Table 2, comprising, respectively, the sets TBF and FBF, are given in Table 3.

In order to simplify the description of the FBF set, the new set of frequency formulas FF is introduced. This set is determined with

$$\{\min, \max\}\{[f_1], [f_1 \quad f_2]\}(\text{FFT}\langle N\rangle(\beta)) \in \text{FF} \wedge \{\min, \max\}(\text{FFT}\langle N\rangle(\beta)) \in \text{FF}, \tag{23}$$

where $\{f_1, f_2\} \in \mathbb{R}$, $f_1 < f_2$, β is an affine signal, and N is an integer number representing the length of Fast Fourier Transform (FFT). Note that for the operators min, max the frequency interval $[f_1, f_2]$ or the frequency f_2 does not need to be specified. In that case default values for f_1 and f_2 are 0 Hz and $f_s/2$ (f_s represents a sampling frequency), respectively.

Table 4: The meaning of operators in AAFA.

Operator	Explanation
IN(s, h)	Satisfied at time t' in which $(s(t') \leq h))$
IN[time](s, h)	Satisfied at time $t'((t' \in$ time) in which $(s(t') \leq h))$
FIN$(f_1, f_2, \text{FFT}(s), \beta)$	Satisfied if $\exists f \in [f_1, f_2] \Rightarrow \text{FFT}(s)(f) \leq \alpha$
GFIN$(f_1, f_2, \text{FFT}(s), \beta)$	Satisfied if $\forall f \in [f_1, f_2] \Rightarrow \text{FFT}(s)(f) \leq \alpha$
$G\{[t]\}(h), F\{[t]\}(h)$, $h \in$ AAFA+A	Satisfied if the formula h holds always or at least once during simulation, respectively

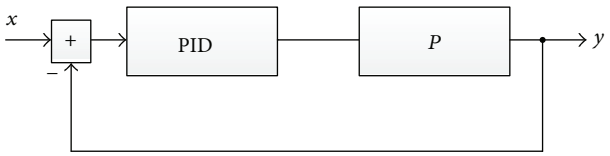


Figure 6: Block diagram of a system with PID controller.

The sets TBF and FBF comprise the smallest set of AAF+A as follows:

$$\text{TBF} \cup \text{FBF} \subset \text{AAF} + \text{A},$$

$$(\alpha \in \text{AAF+A} \wedge t \in \mathbb{R}) \rightarrow \Box(\alpha) \in \text{AAF+A} \wedge \Box[t](\alpha) \in \text{AAF} + \text{A},$$

$$(\alpha, \beta \in \text{AAF+A}) \rightarrow \alpha \oslash \beta \in \text{AAF+A} \wedge !(\alpha) \in \text{AAF+A}.$$

The following table (Table 4) gives a brief description of operators given in Table 2. If α and β represent the ranges modeled with Affine Arithmetic, then the operator $\leq$ in Table 4 assigns that the first range α lies in the second range β ($\alpha \subset \beta$). If β is a real value, then this operator assigns that the upper bound of α is lower or equal to the real variable β.

5. Demonstration Examples

Within this work the applicability of the proposed verification method will be shown through several examples. The examples are chosen to demonstrate ability to handle typical challenges for symbolic simulation like feedback, nonlinearities, and discontinuities. Note that complexity itself is not a challenge by itself. As the first example a closed loop control system composed of a PID controller and a plant is chosen. Its block diagram is shown in Figure 6. Further, as the second example also a feedback system, which needs to set a room temperature to a certain value considering variation in the external temperature, is chosen. The third example through which the performance of the method will be illustrated is a PLL (Phase-Locked Loop) circuit containing the loop and nonlinear elements like a phase detector or a voltage controlled oscillator.

5.1. A Control System including a Parameter Uncertainty of a Plant. For the control system from Figure 6 a plant with the following transfer function will be considered:

$$P(s) = \frac{1}{s^2 + as + 1}, \tag{24}$$

where for the parameter a it will be supposed that it lies in the interval $[0.4, 0.8]$. This range is modeled using Affine Arithmetic as $a = 0.6 + \varepsilon 0.2$ where $\varepsilon \in [-1, 1]$.

Substituting the affine form of the parameter a, $P(s)$ can be rewritten as

$$P(s) = \frac{P_{\text{nom}}(s)}{1 + \varepsilon P_{\text{dev}}(s) P_{\text{nom}}(s)} \quad \varepsilon \in [-1, 1], \tag{25}$$

where $P_{\text{nom}}(s) = 1/(s^2 + 0.6s + 1)$ and $P_{\text{dev}}(s) = 0.2s$. For a PID controller model it is supposed that a noise filter for the derivative term is included, yielding to the following controller structure:

$$C(s) = K_p \left(1 + \frac{1}{T_i s} + \frac{T_d s}{(T_d/20)\, s + 20}\right), \tag{26}$$

where the proportional gain K_p is 1.8, the integral time $T_i = 0.38$ s, and the derivative time $T_d = 0.095$ s. Between the integral and derivative times the ratio of 4 ($T_i = 4 * T_d$) is chosen. In [38] it is shown that this ratio is appropriate for many industrial processes.

In order to behave appropriately whether in time or frequency domain, it is required for a control system to satisfy the certain number of specifications. One of the most important specifications on control systems is the stability of the closed loop system. The gain and phase margin of the closed loop system are typical stability criteria. The gain margin is the maximum amount of the gain which is allowed to increase in the loop before a closed loop system becomes unstable, and the phase margin tells how much the phase lag must increase to make the system unstable. Since it is necessary to specify both margins to ensure appropriate behavior of a system, they can be replaced by a single parameter named the stability margin M_s. This parameter is defined as the shortest distance between the Nyqvist curve of the loop transfer function and the critical point −1. This distance is actually the inverse of the maximum sensitivity. The loop transfer function is determined with $L(s) = C(s)P(s)$ where $C(s)$ and $P(s)$ are the controller transfer function and the plant transfer function, respectively. Mathematically, the stability margin can be expressed as

$$\begin{aligned} M_s &= \inf_{\omega} \left|-1 - L(j\omega)\right| = \inf_{\omega} \left|1 + L(j\omega)\right| \\ &= \left[\sup_{\omega} \left|\frac{1}{1 + L(j\omega)}\right|\right]^{-1} = \left[\sup_{\omega} \left|S(j\omega)\right|\right]^{-1}, \end{aligned} \tag{27}$$

where $S(j\omega) = 1/(1+L(j\omega))$ is the sensitivity function. In particular, the sensitivity function represents the disturbances amplification at the output of the plant by the closed loop system. Recommended values for the stability margin M_s lie in the range of $[0.5, 0.75]$ [38].

Within this paper it will be verified if the control system meets the stability margin specification. In order to satisfy the stability margin specification the system stability margin must lie in the recommended range. This range representing the specification for M_s will be modeled with Affine Arithmetic:

$$\text{spec}(M_s) = M_s' + \varepsilon\delta, \quad \varepsilon \in [-1, 1], \tag{28}$$

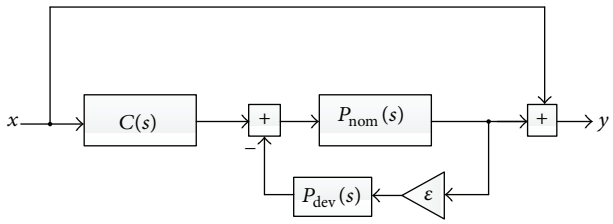


FIGURE 7: M_s calculation block diagram for parameter uncertainty.

where M_s' represents the center value of the specified range and δ represents the maximum absolute distance from the center value. Concretely, since the specified range is [0.5, 0.75], spec(M_s) is equal to

$$\text{spec}\left(M_s\right) = 0.625 + \varepsilon 0.125, \quad \varepsilon \in [-1, 1]. \tag{29}$$

Including the parameter uncertainty in the plant, the stability margin M_s can be rewritten as

$$\begin{aligned} M_s &= \inf_{\omega}\left(\inf_{\varepsilon}\left|1 + C\left(j\omega\right)P\left(j\omega\right)\right|\right) \\ &= \inf_{\omega}\left(\inf_{\varepsilon}\left|1 + C\left(j\omega\right)\frac{P_{\text{nom}}\left(j\omega\right)}{1 + \varepsilon P_{\text{dev}}\left(j\omega\right)P_{\text{nom}}\left(j\omega\right)}\right|\right). \end{aligned} \tag{30}$$

This equation expresses that for every frequency it is necessary to determine the shortest distance of the loop transfer function from the critical point −1. It is very easy to be seen that the shortest distance will be obtained for $\varepsilon = 1$, which corresponds to the lower bound of range $1 + C(j\omega)P(j\omega)$.

A control system including uncertainties meets the stability margin specification if its stability margin M_s lies in the range [0.5, 0.75]. Since M_s is crucial to verify the control system against the given specification, the proposed verification method will use the system given in Figure 7 to calculate the stability margin M_s.

Using the proposed verification method the specification to be verified will be described with AAF+A assertion which will be verified during simulation run. In order to calculate the stability margin following Expression 4 the minimum value of $|1 + C(s)P(s)|$ with respect to frequency will be determined. This value can be found using AAF+A frequency operator min. One has

$$\begin{aligned} M_s &= \min\left(\left|1 + C\left(j\omega\right)P\left(j\omega\right)\right|\right) = \min\left(\left|H\left(j\omega\right|\right)\right) \\ &= \min\left(\text{FFT}\left\langle N\right\rangle\left(h\left(t\right)\right)\right), \end{aligned} \tag{31}$$

where $h(t)$ represents the response of the system with transfer function $1 + C(j\omega)P(j\omega)$ (Figure 7) on Dirac impulse. The operator FFT assigns the Fast Fourier Transform, and N is the number of points for which the Fourier is calculated. In this work N will be set to 2048.

It is important to note that the values calculated by FFT operator are range-based values, because the uncertainty of the plant model is modeled with Affine Arithmetic. The infimum with respect to frequency can be determined as the minimum of all FFT frequency components min(FFT). The stability margin corresponds to the lower bound of the range representing min(FFT) ($\varepsilon = 1$) (Expression 4). As it is said, in order to meet design specification, it is required from the control system that its M_s lies in the specified range spec(M_s). Using the proposed method this requirement can be written as AAF+A assertion which will be verified during simulation as follows:

$$G\left(\text{IN}\left(\min\left(\text{FFT}\left\langle 2048\right\rangle\left(h\left(t\right)\right)\right), 0.625 + \varepsilon 0.125\right)\right). \tag{32}$$

This assertion expresses that infimum with respect to frequency determined with min(FFT⟨2048⟩($h(t)$) must always (assigned with operator G) lie in the range modeling the stability margin specification ($0.625 + \varepsilon 0.125$) during simulation.

5.2. A Room Heating Control System. As the second demonstration example a system which controls a room temperature is chosen. The block diagram of the system is shown in Figure 8.

For room modeling the model including the thermal resistance of the wall between the room and the ambient R_{ra}, and the thermal capacitance of the room C_r is used. One part of heat, brought into the room, leaves the room through the resistor with R_{ra} and the other part is stored in the capacitor with thermal capacity C_r. This mathematical model can be described with the following equation:

$$q = \frac{\theta_r - \theta_e}{R_{ra}} + C\frac{d\theta_r}{dt}. \tag{33}$$

Using Laplase transformation the equation can be transformed into

$$q = \frac{\theta_r - \theta_e}{R_{ra}} + Cs\theta_r, \tag{34}$$

where q is the heat bringing into the system, θ_r is the temperature of the room, and θ_e is the external temperature. To determine the value of the thermal capacitance C_r, the certain number of factors needs to be considered (the heat capacity of the stuff in the room, the air in the room...). Within this work the value $C_r = 10^7$ (J/K) is chosen (Appendix C in [39]). For the thermal resistance R_{ra} the value of $R_{ra} = 0.0846 * 10^{-6}$ (°C/W).

The controller used for this system is a PID controller with the following coefficients:

$$\begin{aligned} C\left(s\right) &= c_1 + c_2 s + \frac{c_3}{s} \\ &= 0.1 * 10^8 + 0.15 * 10^8 s + \frac{0.15 * 10^8}{s}. \end{aligned} \tag{35}$$

According to (34) the room temperature θ_r can be calculated as

$$\theta_r = q\frac{R_{ra}}{1 + sC_rR_{ra}} + \frac{\theta_e}{1 + sC_rR_{ra}}. \tag{36}$$

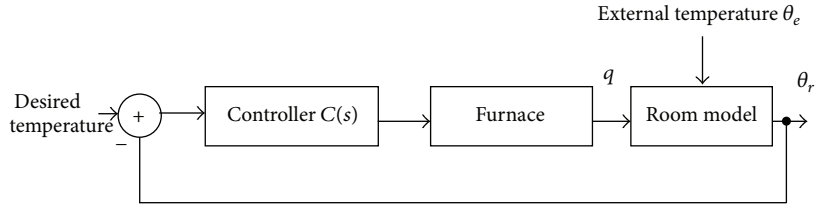


Figure 8: The block diagram of a room heating control system.

Table 5: Simulation results.

Example	Time without AAFA (s)	Time with AAFA (s)	Overhead (s)
The control system with parameter uncertainty	17.6	19.5	1.9
The room heating control system	9.3	10.8	1.5
PLL circuit	7.3	Simulation run stopped	—

For the external temperature θ_e it will be supposed that its value varies and lies in the range $[-2, 1]$. This range will be modeled using Affine Arithmetic as

$$[-2, 1] = -0.5 + \varepsilon 1.5, \quad \varepsilon \in [-1, 1]. \tag{37}$$

Considering the variation in the external temperature it will be verified

(1) if the room temperature θ_r within 5 seconds reaches the value varying within 3% of the final value,

(2) if the final value is reached.

These two requirements will be described using Affine Arithmetic Assertions (AAF+A) and verified during simulation. The final value of the temperature is supposed to be 30°C. The assertion corresponding to the first requirement can be written as

$$F\left(\mathrm{IN}\left[\frac{t_s}{2} + \varepsilon_1 \frac{t_s}{2}\right](\theta_r, \theta_{\text{final}} + \varepsilon_2 \delta) \longrightarrow G\left(\mathrm{IN}\left(\theta_r, \theta_{\text{final}} + \varepsilon_2 \delta\right)\right)\right), \tag{38}$$

where t_s is 5 s, θ_{final} is 30°C, and $\delta = 0.03 * 30 = 0.9$°C and $\varepsilon_1, \varepsilon_2 \in [-1, 1]$. The second requirement can be described with the following assertion:

$$F\left(\theta_r == 30\right). \tag{39}$$

The simulation results are given in Table 5 in Section 5.4.

5.3. A PLL Circuit including Parameter Uncertainties. As the second demonstration example a phased-locked loop (PLL) circuit is chosen. Due to high number of applications phased-locked loops found their place in analog-mixed-signal (AMS) systems. Some of these applications are listed in the following:

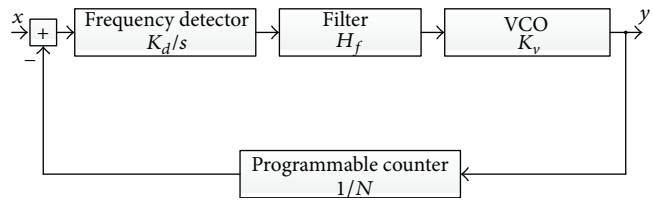


Figure 9: PLL circuit on block level.

(i) in radio transmitters it is used to synthesize frequencies, which are a multiple of a reference frequency;

(ii) clock generators which multiply a low-frequency reference clock to higher operating frequencies of microprocessors;

(iii) in communication systems for coherent demodulation and demodulation of frequency and phase modulated signals.

In this paper a PLL circuit as a frequency synthesizer is considered. Its block diagram is shown in Figure 9. This PLL model models the behavior of the system in ideal conditions. In reality certain numbers of deviations influence the behavior of the PLL causing design specifications to violate from their values previously defined by design. Since the PLL is often the part of more complex AMS systems (e.g., in the role as a frequency synthesizer it is embedded into communication systems to generate carrier frequencies) it is of a great importance to verify if the desired output behaviour in the presence of parameter deviations is still met. Within this work a time delay of the filter from Figure 9 will be considered and added to the PLL model.

As it can be seen from the figure the PLL model is comprised of a frequency detector, a filter, and a voltage controlled oscillator (VCO). The filter with the following transfer function is used:

$$H_f(s) = K_f \frac{(bs + 1)}{as} e^{-\tau s}, \tag{40}$$

where K_f is the filter gain, and within this work it will be supposed that its value is one. For parameters a and b the values $2e^{-3}$ and $680 * 0.5 * 10^{-6}$ are supposed, respectively. The parameter τ represents time delay of the filter, and for this example it will be supposed that its value lies in the range $[0, 100\,\mu s]$. In Section 3.2 it is shown that time delay causes

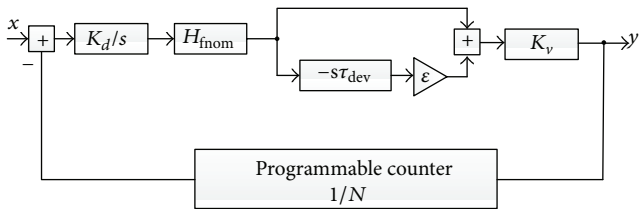


Figure 10: PLL model with the filter time delay.

deviation from the nominal filter behavior approximating the filter transfer function $H_f(s)$ with

$$\begin{aligned} H_f(s) &:= H_{f,\text{nom}}(s)\left(1 + (-s) * \varepsilon\tau_{\text{dev}}\right) \\ &= H_{f,\text{nom}}(s)\left(1 + \varepsilon H_{f,\text{dev}}\right) \quad \varepsilon \in [-1, 1], \end{aligned} \tag{41}$$

where $H_{f,\text{nom}}$ represents the filter transfer function in ideal conditions (time delay τ is zero). Substituting the values for parameters K_f, a, and b, $H_{f,\text{nom}}$ can be rewritten to

$$H_{f,\text{nom}}(s) = \frac{\left(680 * 0.5 * 10^{-6}s + 1\right)}{2 * 10^{-3}s}. \tag{42}$$

The parameter τ_{dev} represents the maximum value of time delay which is for this example 100 μs. The block diagram of the PLL model considering the filter time delay is shown in Figure 10.

For this PLL model it will be verified if the lock time of the PLL to switch from one frequency to within 5 kHz of another frequency is not greater than 1 ms. The PLL parameters such as f_{step}, the minimum obtained value of the output frequency $f_{o,\min}$ and the maximum obtained value of the output frequency $f_{o,\max}$, are supposed to be 100 KHz, 2 MHz, and 3 MHz, respectively. The detector gain K_d and the gain of the voltage controlled oscillator K_v are supposed to be

$$\begin{aligned} K_d &= 0.111\frac{V}{\text{rad}}, \\ K_v &= 11.2 * 10^6\frac{\text{rad}}{Vs}. \end{aligned} \tag{43}$$

The loop gain of the system from Figure 10 is equal to

$$L(s) = \frac{K_d}{s} * H_f(s) * K_v * \frac{1}{N}. \tag{44}$$

From this formula it can be seen that for the maximum ratio of a programmable counter the loop gain will have the minimum value causing maximum lock time. Thus, only this ratio value will be considered. For considered PLL circuit this value is equal to

$$N_{\max} = \frac{f_{o,\max}}{f_{\text{step}}} = \frac{3\,\text{MHz}}{100\,\text{kHz}} = 30. \tag{45}$$

Using the verification method proposed in this work the specification to be verified will be described with AAF+A assertion. This assertion will be verified during simulation as follows:

$$\begin{aligned} &F\left(\text{IN}\left[\frac{t_s}{2} + \varepsilon_1\frac{t_s}{2}\right]\left(f_o, f_{\text{steady}} + \varepsilon_2\delta\right)\right. \\ &\left.\longrightarrow G\left(\text{IN}\left(f_o, f_{\text{steady}} + \varepsilon_2\delta\right)\right)\right), \end{aligned} \tag{46}$$

where the values of ε_1 and ε_2 lie in the range $[-1, 1]$. The operator F in the assertion assigns that the output frequency f_o must eventually within the settling time t_s enter the error band around the value of the steady state $f_{\text{steady}} + \varepsilon_2\delta$ and stay there (assigned with operator G).

We consider only the maximum ratio of the programmable counter, and therefore the frequency value in the steady state is $f_{\text{steady}} = N * f_{\text{step}} = 30 * 100\,\text{KHz} = 3\,\text{MHz}$. The other parameters are according to desired specification equal to

$$\begin{aligned} t_s &= 1\,\text{ms}, \\ \delta &= 5\,\text{kHz}. \end{aligned} \tag{47}$$

Substituting these values in the previous assertion we have

$$\begin{aligned} &F\left(\text{IN}\left[0.5 * 10^{-3} + \varepsilon_1 0.5 * 10^{-3}\right]\left(f_o, 3 * 10^6 + \varepsilon_2 5 * 10^3\right)\right. \\ &\left.\longrightarrow G\left(\text{IN}\left(f_o, 3 * 10^6 + \varepsilon_2 5 * 10^3\right)\right)\right). \end{aligned} \tag{48}$$

5.4. Experimental Results. The SystemC AMS is used as simulation and verification environment. The control system considering the filter parameter uncertainty was simulated and verified for 10^6 s with the sampling rate $T_s = 1$ s. The specification (Assertion 6) passed and the stability margin calculated for the control system is shown in Figure 11. The symbol k in the figure assigns the k frequency component of Fast Fourier Transform which was used to determine the stability margin with respect to frequency.

Using the sampling period $T_s = 0.5$ s the heating control system was simulated for $2 * 10^5$ s. The system met desired requirements and both assertions (Assertions 8 and 9) passed. The signal representing the room temperature θ_r is shown in Figure 11.

The PLL circuit was simulated and verified for 20 s with the sampling period $T_s = 0.1$ ms. It was verified if the lock time of the PLL to switch from 2.9 MHz to within 5 kHz of 3 MHz is less than 1 ms. The Assertion 10 failed, and simulation run was stopped reporting the information about the specification violation. The fact that the PLL output frequency did not (within 1 ms) set to the desired value within the specified tolerance does not imply that it will not do so after some time. To prove this the PLL simulation result is given in Figure 12.

Since the assertion failed and the simulation run was stopped, the PLL response from Figure 12 is the result of simulation in the case where the assertion is omitted. From the figure it can be concluded that the output frequency converges to its final value and deviated terms converge

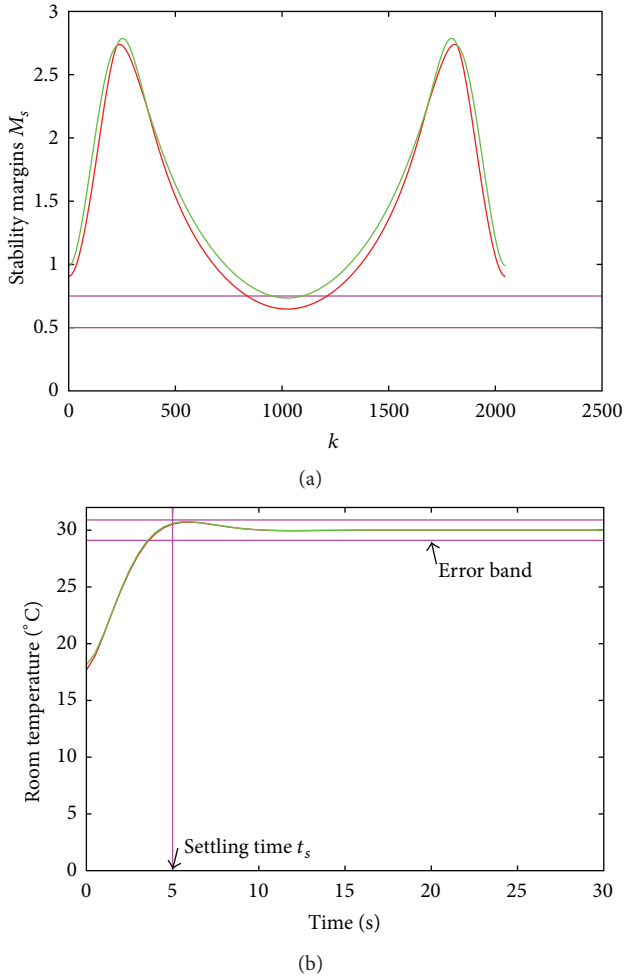


FIGURE 11: Simulation results. The stability margin of the control system M_s and the room temperature θ_r.

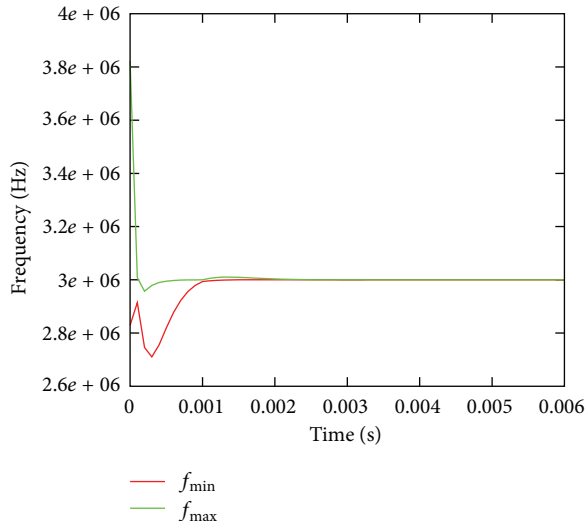


FIGURE 12: The PLL output frequency.

to zero. This fact is one of the main advantages of the Affine Arithmetic approach. Its ability to identify the correlation between system quantities reaches its maximum in the systems with feedback loops like the case with our demonstration examples.

Hence, even in the case where the loop contains the nonlinear elements the additional terms (which are the result of overapproximation introduced by nonlinear operations) will have negligible values, and the output will converge to its finite value. To free the memory of unnecessary variables, the authors in [28] propose the (cleanup) method. Concretely, all terms under some user specified value are replaced with only two symbols, the one representing the sum of all terms with a positive and the other with a negative sign. In this way the safe inclusion of the result is kept, and the number of terms is drastically decreased.

Table 5 summarizes the simulation times necessary for all designs in the case where AAF+A assertions were included into design simulation and when they were omitted. It can be noted that in the case where the assertions were satisfied, the proposed verification method generated additional simulation time, but the overhead was not high.

The simulation time of the PLL in the case where the assertion was embedded into simulation was omitted. The reason lies in the fact that the assertion failed stopping the simulation process, and hence the time required for simulation was much lower than for the case in which the assertion was not included.

6. Conclusion and Future Work

This work introduces a methodology that enables verification of analog/mixed-signal systems including deviations. The verification method is combined with symbolic simulation which generates the worst case dependable response adding deviations to a system model and modeling them as ranges. Since the generated output behaviour contains all possible traces for the considered parameter set the proposed assertion-based technology can provide formal verification result using simulation-based techniques. The assertions use Affine Arithmetic to model allowed or forbidden areas of typical system properties as ranges. The specified ranges are further combined with Boolean logic, frequency operators, and temporal logic which allows us to verify the system behaviour in time, but also in the frequency domain. The assertions are embedded into simulation, and as soon as the assertion violation is detected, the simulation run is stopped, and information about the assertion harm is reported.

Overapproximation is a challenge that can become a problem for strongly nonlinear systems. The first step to deal with this problem was proposed in [28] and found its applicability in the systems containing the loops. The further step towards the problem solution is to modify Affine Arithmetic in the way that we keep the second order terms in symbolic representation and in this way that we reduce overapproximation.

Furthermore, the AAF+A assertions (Table 2) will be extended from rather formal operators to libraries of application-specific properties that are close to requirement specifications found in various application domains. Also,

the method up to now verifies the system against the specifications for which time and frequency requirements must be known in advanced. One interesting direction in the future would be to extend the method to extract the information about lower and upper bounds of time or frequency for which the system behaviour is still desirable under the considered set of deviated parameters.

Acknowledgment

This work has been funded by the Vienna Science and Technology Fund (WWTF) through Project ICT08_012.

References

[1] R. Y. Rubinstein, *Simulation and the Monte Carlo Method*, John Wiley & Sons, New York, NY, USA, 1981.

[2] D. E. Hocevar, M. R. Lightner, and T. N. Trick, "Study of variance reduction techniques for estimating circuit yields," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 2, no. 3, pp. 180–192, 1983.

[3] K. Antreich, H. Gräb, and C. Wieser, "Practical methods for worst-case and yield analysis of analog integrated circuits," *International Journal of High Speed Electronics and Systems*, vol. 4, no. 3, pp. 261–282, 1993.

[4] M. Rafaila, C. Grimm, C. Decker, and G. Pelz, "Sequential design of experiments for effective model-based validation of electronic control units," *e&i Elektrotechnik Und Informationstechnik*, vol. 127, pp. 164–170, 2010.

[5] M. S. Branicky, M. M. Curtiss, J. A. Levine, and S. B. Morgan, "RRTs for nonlinear, discrete, and hybrid planning and control," in *Proceedings of the 42nd IEEE Conference on Decision and Control*, pp. 657–663, December 2003.

[6] A. Bhatia and E. Frazzoli, "Incremental search methods for reachability analysis of continuous and hybrid systems," in *Proceedings of the International Workshop Hybrid Systems: Computation and Control (HSCC '04)*, vol. 2993 of *Lecture Notes in Computer Science*, pp. 142–156, Springer, 2004.

[7] A. Julius, G. Fainekos, M. Anand, I. Lee, and G. Pappas, "Robust test generation and coverage for hybrid systems," in *Proceedings of the International Workshop Hybrid Systems: Computation and Control (HSCC '07)*, vol. 4416 of *Lecture Notes in Computer Science*, pp. 329–342, Springer, 2007.

[8] J. M. Esposito, "Randomized test case generation for hybrid systems: metric selection," in *Proceedings of the 36th Southeastern Symposium on System Theory*, pp. 236–240, 2004.

[9] Q. Zhao, B. H. Krogh, and P. Hubbard, "Generating test inputs for embedded control systems," *IEEE Control Systems Magazine*, vol. 23, no. 4, pp. 49–57, 2003.

[10] J. Kapinski, B. H. Krogh, O. Maler, and O. Stursberg, "On Systematic simulation of open continuous systems," in *Proceedings of the International Workshop Hybrid Systems: Computation and Control (HSCC '03)*, vol. 2623 of *Lecture Notes in Computer Science*, pp. 283–297, Springer, 2003.

[11] A. Girard and G. Pappas, "Verification using simulation," in *Proceedings of the International Workshop Hybrid Systems: Computation and Control (HSCC '06)*, vol. 3927 of *Lecture Notes in Computer Science*, pp. 272–286, Springer, 2006.

[12] A. Chutinan and B. H. Krogh, "Computing polyhedral approximations to flow pipes for dynamic systems," in *Proceedings of the 37th IEEE Conference on Decision and Control (CDC '98)*, pp. 2089–2094, December 1998.

[13] A. B. Kurzhanski and P. Varaiya, "Ellipsoidal techniques for reachibility analysis," in *Proceedings of the International Workshop Hybrid Systems: Computation and Control (HSCC '00)*, vol. 1790 of *Lecture Notes in Computer Science*, pp. 202–214, Springer, 2000.

[14] S. Prajna and A. Jadbabaie, "Safety verification of hybrid systems using barrier certificates," in *Proceedings of the International Workshop Hybrid Systems: Computation and Control (HSCC '04)*, vol. 2993 of *Lecture Notes in Computer Science*, pp. 477–492, Springer, 2004.

[15] E. M. Clarke, O. Grumberg, and D. A. Peled, *Model Checking*, The MIT Press, 1999.

[16] J. R. Burch, E. M. Clarke, K. L. McMillan, D. L. Dill, and L. J. Hwang, "Symbolic model checking: 1020 states and beyond," *Information and Computation*, vol. 98, no. 2, pp. 142–170, 1992.

[17] E. Clarke, O. Grumberg, and D. Long, "Verification tools for finite-state concurrent systems," in *Proceedings of the Decade of Concurrency, Reflections and Perspectives*, vol. 803 of *Lecture Notes in Computer Science*, pp. 124–175, Springer, 1994.

[18] O. Maler and G. Batt, "Approximating continuous systems by timed automata," in *Proceedings of the 1st international workshop on Formal Methods in Systems Biology (FMSB '08)*, vol. 5054 of *Lecture Notes in Computer Science*, Springer, 2008.

[19] O. Stursberg, S. Kowalewski, and S. Engell, "On the generation of timed discrete approximations for continuous systems," *Mathematical and Computer Modelling of Dynamical Systems*, vol. 6, no. 1, pp. 51–70, 2000.

[20] O. Stursberg, S. Kowalewski, and S. Engell, "Timed approximations of hybrid processes for controller verification," in *Proceedings of the 1st IFAC Conference on Analysis and Design of Hybrid Systems*, pp. 289–295, 2003.

[21] R. Alur, T. A. Henzinger, G. Lafferriere, and G. J. Pappas, "Discrete abstractions of hybrid systems," *Proceedings of the IEEE*, vol. 88, no. 7, pp. 971–984, 2000.

[22] T. Henzinger and H. Wong-Toi, "Linear phase-portrait approximations for nonlinear hybrid systems," in *Proceedings of the Hybrid Systems III: Verification and Control*, vol. 1066 of *Lecture Notes in Computer Science*, pp. 377–388, Springer, 1996.

[23] W. Hartong, L. Hedrich, and E. Barke, "Model checking algorithms for analog verification," in *Proceedings of the 39th Annual Design Automation Conference (DAC '02)*, pp. 542–547, June 2002.

[24] D. Grabowski, D. Platte, L. Hedrich, and E. Barke, "Time constrained verification of analog circuits using model-checking algorithms," *Electronic Notes in Theoretical Computer Science*, vol. 153, no. 3, pp. 37–52, 2006.

[25] W. Hartong, L. Hedrich, and E. Barke, "On discrete modelling and model checking of nonlinear analog systems," in *Proceedings of the 14th International Conference on Computer Aided Verification (CAV '02)*, pp. 401–413, 2002.

[26] S. Lämmermann, A. Jesser, M. Rathgeber et al., "Checking heterogeneous signal characteristics applying assertion-based verification," in *Proceedings of the Frontiers in Analog Circuit Verification (FAC '09)*, Grenoble, France, 2009.

[27] A. Pnueli and O. Maler, "Extending PSL for analog circuits," Tech. Rep., 2005, PROSYD Deliverable D 1.3/1.

[28] W. Heupke, C. Grimm, and K. Waldschmidt, "Semi-symbolic simulation of nonlinear systems," in *Proceedings of the Forum on Specification and Design Languages (FDL '05)*, ECSI, Lausanne, Switzerland, September 2005.

[29] C. Grimm, W. Heupke, and K. Waldschmidt, "Refinement of mixed-signal systems with affine arithmetic," in *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE '04)*, pp. 372–377, IEEE Press, February 2004.

[30] C. Grimm, W. Heupke, and K. Waldschmidt, "Analysis of mixed-signal systems with affine arithmetic," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, no. 1, pp. 118–123, 2005.

[31] D. Grabowski, M. Olbrich, C. Grimm, and E. Barke, "Range arithmetics to speed up reachability analysis of analog systems," in *Proceedings of the Forum on Specification, Verification and Design Languages (FDL '07)*, Barcelona, Spain, 2007.

[32] N. Femia and G. Spagnuolo, "True worst-case circuit tolerance analysis using genetic algorithms and affine arithmetic," *IEEE Transactions on Circuits and Systems*, vol. 47, no. 9, pp. 1285–1296, 2000.

[33] A. Lemke, L. Hedrich, and E. Barke, "Analog circuit sizing based on formal methods using affine arithmetic," in *Proceedings of the IEEE/ACM International Conference on Computer Aided Design (ICCAD '02)*, pp. 486–489, November 2002.

[34] C. F. Fang, R. A. Rutenbar, M. Püschel, and T. Chen, "Toward efficient static analysis of finite-precision effects in DSP applications via affine arithmetic modeling," in *Proceedings of the 40th Design Automation Conference (DAC '03)*, pp. 496–501, June 2003.

[35] C. F. Fang, T. Chen, and R. A. Rutenbar, "Floating-point error analysis based on affine arithmetic," in *Proceedings of IEEE International Conference on Accoustics, Speech, and Signal Processing (ICASSP '03)*, vol. 2, pp. 561–564, April 2003.

[36] R. E. Moore, *Interval Analysis*, Prentice-Hall, Englewood Cliffs, NJ, USA, 1966.

[37] D. Grabowski, M. Olbrich, and E. Barke, "Analog circuit simulation using range arithmetics," in *Proceedings of the Asia and South Pacific Design Automation Conference (ASP-DAC '08)*, pp. 762–767, IEEE Computer Society Press, Seoul, Korea, March 2008.

[38] K. J. Åström and T. Hägglund, *PID Controllers: Theory, Design and Tuning*, Instrument Society of America, 2nd edition, 1995.

[39] D. Ryder-Cook, "Thermal modelling of buildings," Tech. Rep., University of Cambridge, 2009.