Analog Simulation Meets Digital Verification – A Formal Assertion Approach for Mixed-Signal Verification

Alexander Jesser, Lars Hedrich
Department of Computer Science
University of Frankfurt
D-60325 Frankfurt a.M., Germany
{jesser,hedrich}
@em.cs.uni-frankfurt.de

Stefan Laemmermann, Roland Weiss, Juergen Ruf, Thomas Kropf, Wolfgang Rosenstiel
Department of Computer Engineering
University of Tuebingen
D-72076 Tuebingen, Germany
{laemmerm,weissr,ruf,kropf,rosenstiel}
@informatik.uni-tuebingen.de

Alexander Pacholik, Wolfgang Fengler
Computer Architecture Group
Ilmenau Technical University
D-98693 Ilmenau, Germany
{alexander.pacholik,wolfgang.fengler}
@tu-ilmenau.de

Abstract— Functional and formal verification are important methodologies for complex mixed-signal designs. But there exist a verification gap between the analog and digital blocks of a mixed-signal system. Our approach improves the verification process by creating mixed-signal assertions which is described by a junction of digital assertions and analog properties. The proposed method is a new assertion-based verification flow for designing mixed-signal circuits. The effectiveness of the approach is demonstrated on a Σ/Δ-converter.

I. Introduction

The increase of integration density and functionality in todays microelectronic systems force designers for new elaborated system architectures. For this reason mixed-signal circuits have attracted considerable interest recently. A mixed-signal system is a dynamically interacting system with both analog and digital circuit parts integrated on one chip. One characteristic of mixed-signal systems is that each subsystem interacts simultaneously with each other by internal connections and reacts to inputs coming from extern. Another characteristic is different changes behavior in both domains. Digital systems behavior usually exhibits discrete changes in time and value, whereas analog circuits usually exhibit continuous changes. These different behaviors challenge mixed-signal circuit designers in modeling and validation.

Today mixed-signal circuit validation is still done by several *analog/mixed-signal* (AMS) simulators that only give an input pattern dependent, incomplete correctness of the circuit. To overcome todays design complexity, formal prove techniques that guarantee total correctness, must be applied, whenever possible. Beyond, the verification process time consumption has to be below a practical limit. One common verification technique is called *assertion-based*. Assertion-based verification joins formal properties and simulation-based verification to provide more powerful system verification. In other words, assertion-based verification methods prove desired design properties (almost formal) during simulation. The functional verification process has the advantage of coverage analysis and randomized input generation.

In this paper, we introduce a method to describe assertions for assertion-based verification of mixed-signal systems. To validate our method we used the MLDesigner [1] tool. MLDesigner supports the composition of models by using model fragments with different models of computation. This is particularly useful for modeling mixed-signal systems. Additionally we were able to use an assertion monitoring library which can be connected to the tool. Our approach is capable to describe analog and digital behavior simultaneously in closed formal property assignments for assertion-based verification. These properties can also be used for subsequent analog/mixed-signal formal verification tasks.

The following sections discuss the flow we used for assertion-based mixed-signal verification. In section II an introduction to mixed-signal verification and assertion-based verification is given. Section III gives a short survey of actual works related to analog/mixed-signal verification and formal specification. Section IV describes our definition of mixed-signal assertions and their application to functional verification (simulation) and formal verification. In section V we demonstrate our verification method by considering a Σ/Δ - converter. Finally, we conclude and mention further work in section VI.

II. Preliminary

Functional verification and assertion-based verification (ABV) [2] are well-established methods in system on chip design. A property specification is the formalization of design aspects whose format and clearly defined semantics make it readable for humans and machines. Assertions (properties) can be expressed in a formal language as the *property specification language* (PSL) [3] or with temporal logic as *linear temporal logic* (LTL), *finite linear time temporal logic* (FLTL), and *computation tree logic* (CTL) [4]. Assertions which are manually generated from the specification in parallel to the design are included in the *design under verification* (DUV) through monitors. Monitors

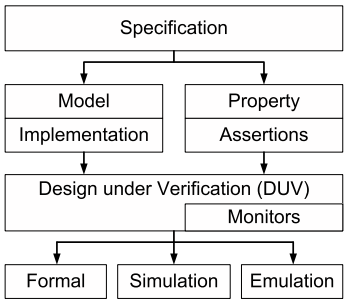

Fig. 1. Assertion-based verification (ABV) flow.

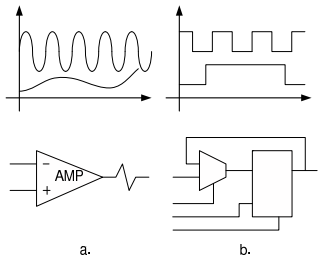

Fig. 3. (a) Analog and (b) digital circuit simulation.

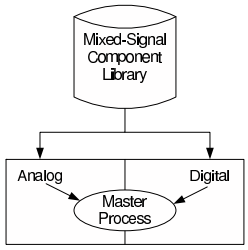

Fig. 4. Mixed-signal simulation elements.

compare the temporal behavior of the assertions against the design during the simulation. Assertions are used in the validation environments of *transaction level modeling* (TLM), *register transfer level* (RTL) and gate level. This flow is shown in Figure 1. The reasons to apply assertions are the following:

- Assertions detect design errors at their source and increase observability.
- Assertions actively monitor a design to ensure correct functional behavior.
- Assertions can be used for functional and formal verification.
- Assertions have two standard languages: Property Specification Language (PSL) [3] and SystemVerilog Assertions (SVA) [5].

In order to discuss properties in detail, it is beneficial to take a layered view. Properties are composed of three layers:

1. The *Boolean layer* consists of propositions and Boolean connectives.
2. The *Temporal layer* adds operators for temporal reasoning to the Boolean layer.
3. The *Verification layer* provides indicators for the verification tools how to apply the property.

The third layer is used to control the high-level behavior of the verification tools. The first two layers define the actual property that relates to parts of the system under verification, thus describing desired or error states. An example of a PSL property construction is shown in Figure 2. *Assert* is for the verification operation. The next symbol

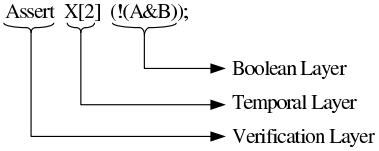

Fig. 2. PSL property

X[2] signifies the next operator and includes a time interval of two time steps. The last part includes the *Boolean* operation of the property.

Todays mixed-signal simulation allows to incorporate the AMS blocks along with digital blocks. The purpose of analog simulation is to verify the required functionality according to voltage, current and timing specifications. The view on digital blocks is to verify the required functionality according to predetermined input vectors, assertions and timing specifications. Figure 3 shows an analog and a digital simulation example and their signal traces.

The mixed-signal simulation environment needs the information of the digital and analog block to simulate the interaction process between the two parts. Currently, most tools couple an existing digital simulator to an analog simulator. The simple block diagram of such a simulator is shown in Figure 4. The analog and digital simulators run as separate processes, which will be combined by a master process. The interfaces between the analog and digital parts are simple D/A- and A/D - interconnections which can handle the different signal characteristics for simulation.

Simulation is the state of the art in validation procedure for complex analog and mixed-signal circuits. Mixed-signal verification is often done by observing the waveforms of the inputs, outputs, and the interface signals between the analog and digital blocks. Because of the complex structure of mixed-signal circuits conventional simulation for validation is not sufficient. Due to that, formal verification is inevitable during the design process of complex systems. This is the reason for applying formal methods to analog systems design. However, analog behavior challenge designer in writing formal properties.

III. Related Work

In the last decade several formal verification techniques for digital circuits are proposed, some of them are established in todays commercial design tools. However, AMS designs suffer from the absence of formal verification methods. But simulation is still an essential validation approach in AMS design flows. Nevertheless, in recent years several academic approaches for verification of analog and mixed-signal systems were proposed. Most of them are based on the finite state discrete abstraction by partitioning the con-

tinuous state space, whose dynamics is described by differential algebraic equations (DAE), into hypercubes using a fixed grid. In [6] a model checking tool, called *amcheck*, for non-linear analog circuits is proposed. The most popular tool for hybrid systems is called *HyTech* and was introduced in [7]. *HyTech* is a symbolic model checker for linear hybrid automata that can analyze hybrid systems automatically using polyhedral state sets. In [8] the tool *d/dt* is proposed which computes the reachable states for hybrid systems by discrete time integration. Sets of states are represented by orthogonal polyhedra, which guarantees the reachable state enclosures. In [9] the DAE representation of the analog circuit part is approximated by iterative assignments and then implemented using a digital hardware description language like VHDL. The commercial bounded model checker *GateProp* is then used to verify crucial properties for this model. In [10] a new model to describe hybrid systems is proposed. The model is based on *continuous Petri nets* called *timed hybrid Petri nets* (THPN). With this model reachable states can be found in the system.

Typically, desired properties in the analog circuit domain differ from the digital domain. Analog properties are usually characterized using continuous time in physical signal behavior and are often considered in the frequency domain, where frequency transformations can be performed. Formal specification languages used in formal verification flows are still insufficiently resolved in analog design validation. Yet, all analog formal specification languages are principally based on LTL or CTL, that actually cannot fully cover desired analog behavior. However, some incomplete academic analog specification languages are introduced with the aim to a designer-oriented characterization. In [11] an extension of CTL, called CTL-AT, is introduced, that can describe dynamic behavior with time constrained temporal operators. The most known *property specification language* PSL is extended for analog circuits in [12], whereas it is restricted to signal level specifications.

In contrary to digital circuits, where formal specifications enter todays design flows, specification languages in the analog domain suffer from expressiveness. However, a few rudimentary approaches exist treating analog behavior. Yet, formal AMS properties for assertion-based verification without exclusion of subsequent formal verification procedures do not exist at the moment. With this work we are going to close the gap between simulation based verification and formal verification with formal mixed-signal assertions.

IV. Definition of Mixed-Signal Assertions

Mixed-signal systems consist of an interacting collection of digital and analog components integrated in few or ideally one single chip. The main difficulty in designing mixed-signal systems consists in the different time and value characteristics. Even combing the different modeling techniques for each subcircuit is a challenge in todays design process. Analog circuits are usually described by some differential equations in the following form:

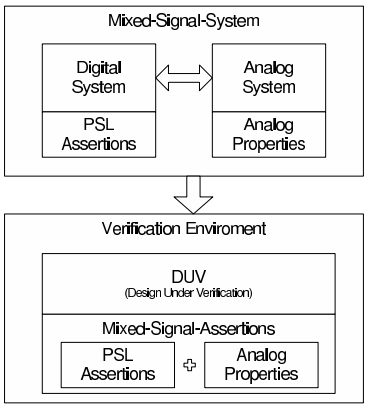

Fig. 5. Combination of analog and digital assertions.

$$0 = \vec{f}(\dot{\vec{x}}, \vec{x}(t), \vec{u}(t), t) \quad (1)$$

$$\vec{y}(t) = \vec{g}(\vec{x}(t), \vec{u}(t), t) \quad (2)$$

$$\vec{x}(0) = \vec{x}_0 \quad (3)$$

where $\vec{x} \in \mathbb{R}^n$ is the state vector, $\vec{u} \in \mathbb{R}^m$ the input vector and $\vec{y} \in \mathbb{R}^r$ the output vector. $\vec{x}_0$ denotes the initial state. More generally, equation (1) can be replaced by a set of differential and algebraic equations, which then is called a differential-algebraic system (DAE system). Digital circuits instead are usually described by discrete models where transitions are related to a given global clock.

A behavior of a digital system can be viewed as a sequence of states, $s : \mathbb{N} \longrightarrow \mathbb{B}^n$. Unlike digital systems where such values are observed only at certain time instances, in analog systems we are often interested in the evolution of these quantities over the entire time axis. Such behaviors are viewed as signals of the form $s : \mathbb{R}_+ \longrightarrow \mathbb{R}^n$.

Digital sequences can be simple represented by a list $s[0], s[1], \ldots$ that specifies their values at each time instance. Analog signals cannot be completely specified in this way due to the density of real numbers. In some cases they can be specified by a closed form expression, for example $s(t) = sin(t)$, while for the purpose of simulation they are represented via discretization of both the time domain and the state-space.

Nowadays, the two parts (analog and digital) of mixed-signal systems are in most cases verified separately. There exist a lot of methods for the verification and validation of digital circuits for checking by simulation or formal verification. The usual analog methods are simulation and now formal verification is coming up. The verification of the behavior between analog and digital blocks in mixed-signal system is very complex and crucial. Furthermore, there exists no formal specification language for mixed signal properties, which can be proved by functional or formal verification. This is the reason for the verification gap when verifying mixed-signal systems. To close this gap we need a method to define *mixed-signal assertions* (MSA). These assertions are connected through an analog and digital assertion part for automatic observation during simulation. The problem is the combination of the two parts with the different views of the system. Therefore, we need a formal

specification of analog and digital assertion which allows to combine the two different assertions to an assertion for mixed-signal systems. The idea of the combination is shown in Figure 5. The upper part shows the analog and digital systems with their individual desired properties, separately. The digital blocks exhibit PSL assertions describing digital behavior, whereas the analog blocks exhibit analog properties. The part below depicts our new method combining analog and digital assertions. The aim is a mixed-signal system as a design under verification (DUV) include MSA in the verification environment.

In the verification flow with mixed-signal assertions, shown in Figure 6, a process is described to define MSA. These assertions are formal properties which can be applied for different verification methods as functional and formal verification. To reach this aim we need a new description language for MSA. PSL is an industry standard for formal digital assertions and is used in a lot of simulation and formal verification tools. This is the reason why we take PSL as the basic language for the proposed MSA. The analog block needs some new methods to describe analog assertions which must be transformed into the PSL language.

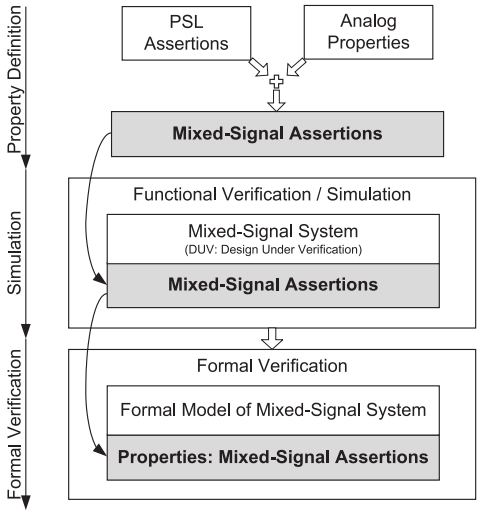

Fig. 6. Verification flow with mixed-signal assertions.

A. Analog Properties

The design focus in mixed-signal systems makes it difficult to apply assertions for both systems together. It is difficult to combine the assertions from the digital and analog system for the verification of the interfaces and connections.

It is often desired to describe the behavior of analog systems in the frequency domain. Frequency behavior and transfer functions are typical properties considered by analog designers. All these properties can actually not be described by common specification languages. The main problem consists therein that analog verification methods are usually based on the analog state space discretization derived from the differential algebraic equations (DAE) considering dynamic behavior in the time domain. Other properties, like offset, gain, *common-mode rejection ratio* (CMRR), and *power supply rejection ratio* (PSRR), refer to the time domain and are still insufficiently resolved in formal specification, especially in writing user friendly properties.

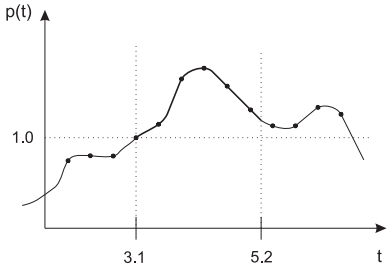

Fig. 7. Timed constraint analog assertion example.

The formal specification languages used in common digital verification tools are not suitable for analyzing analog circuits. Analog formal specification languages should consist of constructs describing timing behavior and analog subspaces. For this reason formal analog specification languages are extended by analog operators (>,<,=) that offer the possibility to inductively define sets of states. Furthermore, timing constraints in form of intervals $[t_{low} : t_{high}]$, with $t_{low}, t_{high} \in \mathbb{R}$ have to be included. These extensions are enhanced to the Temporal and Boolean layer within standard PSL syntax (see Figure 2). As an example Figure 7 depicts the meaning of analog timed constrained assertion. Considering the assertion

$$next\_a[3.1 : 5.2](p(t) > 1.0) \tag{4}$$

the bold section of the waveform represents the regarded time interval [3.1:5.2], because the waveform within the boundaries is continuously above the threshold of 1.0. For further computation the continuous time is discretized by e.g. equidistant time points. The result of the assertion is then a collection of all discretized points within the desired time interval located on the bold waveform section. This means, that properties consisting of continuous time and state representations are mapped to related discrete time points and set of states. With this approach usually digital behavior like reachable analysis can also be done for analog/mixed-signal cases.

Considering mixed-signal systems, the concurrent interacting of the analog and digital subcircuits is of extreme importance. Therefore the interface behaviors between the analog and digital parts have to be primarily analyzed. Hence, an assertion, declared for mixed-signal behavior, can be used to specify characteristics desired during simulation time.

B. Construction of Mixed-Signal-Assertions

The construction of mixed-signal assertions is dependent on the interaction between the analog and the digital blocks. The single operations to construct the mixed-signal assertion begin with the definition of analog and digital properties of their own blocks. After this step we ana-

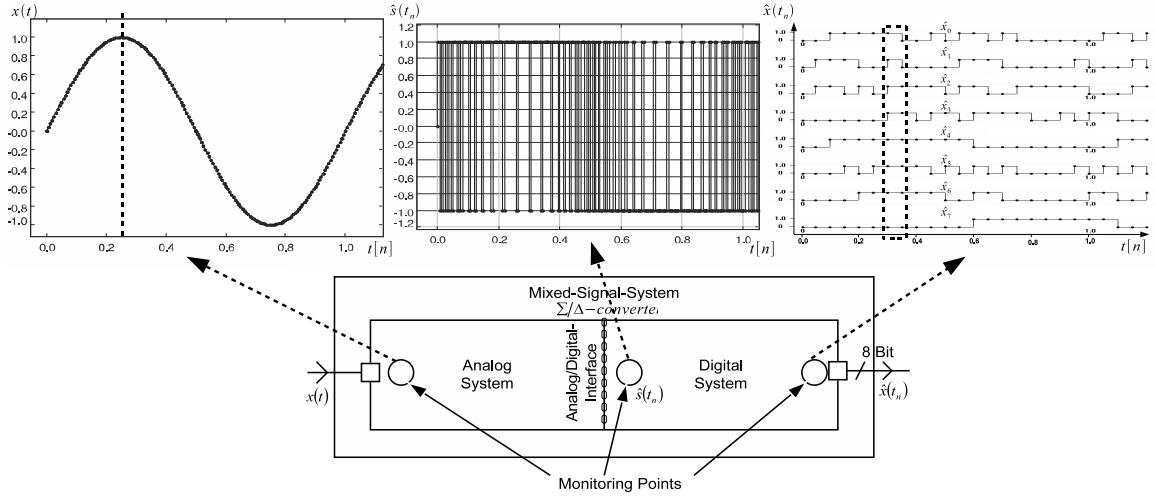

Fig. 8. An example of construction mixed-signal assertions.

lyze the interaction and dependency of the analog and digital properties. The block properties which drive the other properties are the preconditions and the others are the postcondition. This sequence of operations is shown in Figure 9.

1. Definition of analog properties	1
2. Definition of digital properties	2
3. Interaction between analog and digital properties	3
(Which properties drive the others?)	4
4. The properties which drive are precondition	5
5. Other properties are postcondition	6
6. Combine the two conditions	7

Fig. 9. Steps for mixed-signal assertion construction.

All mixed-signal assertions have the same basic structure with a precondition and a postcondition which are explained in Formula (5) and (6). The property becomes true if at first the precondition is true and then the postcondition. In Formula (5) the analog block drive the digital block. Therefore the analog property is the precondition and the digital block is the postcondition for the assertion. Formula (6) represents the opposite of Formula (5), and vice versa.

$$\{AnalogPrecondition\} |\!\rightarrow \{DigitalPostcondition\} \quad (5)$$

$$\{DigitalPrecondition\} |\!\rightarrow \{AnalogPostcondition\} \quad (6)$$

This standard construction of properties allows checking a block condition, the interaction between two blocks and the other block condition. Finally we verify the dependent behavior of the two different blocks. For a better understanding we consider the mixed-signal example of a Σ/Δ-converter shown in Figure 8 with the three monitor points in the analog and digital blocks. The Σ/Δ-converter is the same that also will be used in section V for case study.

The left top of Figure 8 depicts an analog input signal named $x(t)$. In the middle of the top of Figure 8 the modified signal $\hat{s}(t_n)$ in the digital block is shown. At the right top of Figure 8 the related discretized digital output signal $\hat{\mathbf{x}}(t_n)$ is shown. The digital output signal is converted into a bit vector specified by 8 bits $\hat{x}_0 \dots \hat{x}_7$ shown in the dashed box. The waveforms show the delay time from input to the output signal. For proving these conditions we need the proposed formal properties for a better overview during the simulation and a possible following formal verification. An example condition (property) is that a special analog state drives the value of the bit vector to reach the bit vector $\hat{x}_0 \& \hat{x}_1 \& \hat{x}_2 \& \hat{x}_3 \& \hat{x}_4 \& \hat{x}_5 \& \hat{x}_6 \& !\hat{x}_7$. At first we need the precondition of the analog block which is $x(t)$ reaches 1. The second condition of the digital block is the bit vector. These two condition are marked in the waveforms of Figure 8 with the bold line. The result mixed-signal assertion is shown in Figure 10. The analog assertion is connected with an *implication* ($\rightarrow$) which means at first the analog precondition must get true thereafter the digital postcondition. The digital assertion is included with the *globally* operator G and the *next* operator X with n time steps (delay time) which is dependent to the mixed-signal system.

The time between the analog and the digital block is a problem for creating MSA. The application of MSA is different for both verification methods. Some simulation tools can handle two different time conditions. In the digital block we have discrete time steps connected to a global clock signal. The other block has a real valued continuous time signal, which can be transformed into discrete time explained in section IV.A. After this the assertions can be used in all simulation tools.

$GeneralFormula$	1
$\{AnalogPrecondition\} \mid\!\rightarrow$	2
$\{DigitalPostcondition\}$	3
$ExampleAssertion$	4
$\{x(t) = 1\} \mid\!\rightarrow$	5
$\{X[n](\hat{x}_0 \& \hat{x}_1 \& \hat{x}_2 \& \hat{x}_3 \& \hat{x}_4 \& \hat{x}_5 \& \hat{x}_6 \& !\hat{x}_7)\}$	6

Fig. 10. Example of a mixed-signal assertion.

V. Case Study

The application of assertions for mixed signal systems is demonstrated using a first order sigma-delta (Σ/Δ) converter. A Σ/Δ-converter is a well known mixed-signal system converting analog signals to digital ones represented by discrete bit vectors. The converter consists of complex analog and digital parts and a defined interface between them. The mixed-signal assertion verification method helps the designer to get a better overview of the whole system during the verification. This is shown with some MSA in this example. We used the MLDesigner tool to simulate the Σ/Δ-converter and to apply assertion-based verification.

A. Σ/Δ-Converter

In Figure 11 the first order Σ/Δ-converter is shown which we used to demonstrate our approach. The Σ/Δ-converter transfers a time and value continuous signal $x(t)$ into a time and value discrete fixed bit length signal $\hat{\mathbf{x}}(t_n)$ dependent to equidistant time points t_n related to the sampling frequency f_s. The considered Σ/Δ-converter consists of a subtracter, an integrator, an amplifier, a 1 bit quantizer, a digital low-pass filter, a 1 bit D/A-converter, a decimation filter, and a sample and hold device. The interconnection between the analog and the digital part is the 1 bit quantizer which transforms the analog signal to a Boolean value represented by -1 and 1. The interconnection device between the digital and analog parts is a simple 1 bit D/A-converter, which transforms the digital signal $\hat{s}(t_n)$ to a continuous signal conditioned for analog subtraction (see signal $g(t)$).

The sample device uses an oversampling frequency of $f_{os} = 8 \cdot f_s = 160$ Hz. The final sampling frequency of $f_s = 20$ Hz is achieved by using the decimation filter. For the gain of the amplifier we choose $k = 1.4$. For the input signal we choose a overlayed sinusoid signal $x(t) = A_1 \cdot sin(2\pi \cdot f_1) + A_2 \cdot sin(2\pi \cdot f_2)$ with $A_1 = 1, A_2 = \frac{1}{4}, f_1 = 1$ Hz, and $f_2 = 3$ Hz shown in Figure 12. The digital output signal $\hat{\mathbf{x}}(t_n)$ has a resolution of 8 bits declared with $\hat{x}_7(t_n), \hat{x}_6(t_n), \ldots, \hat{x}_0(t_n)$, whereas $\hat{x}_7(t_n)$ is the most significant bit. We placed four meaningful monitor points to assure overall and intern behavior to apply assertion-based verification. Two monitor points are placed at the input $x(t)$ and the output $\hat{\mathbf{x}}(t_n)$, respectively. The third monitor point is placed after the 1 bit quantizer to display the bit stream $\hat{s}(t_n)$ we get after the integrator. The last monitor $g(t)$ point is placed after the amplifier. The signals related to the first three monitor points $x(t), \hat{\mathbf{x}}(t_n)$, and $\hat{s}(t_n)$ are depict in Figure 12.

B. Simulation Environment

We implemented the design in MLDesigner to simulate analog as well as digital behavior. The terminology for a model of computation is modeling domain for MLDesigner. We used the modeling domains CTDE (continuous time discrete event) for analog/mixed-signal parts and DE (discrete event) for digital parts. Thanks to the multi domain approach, mixing different models of computation is easy.

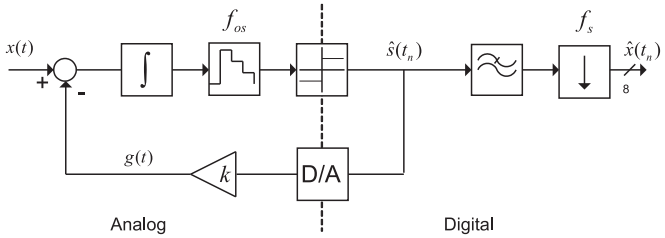

Fig. 11. Σ/Δ - converter.

The only thing to consider is the hierarchy of modeling domains regarding simulation time. To match the simulation needs, the domains used in the model structure need to match the following order: continuous time, discrete time, untimed. Thus, for mixed-signal systems, the top level domain is CTDE.

We used a custom assertion checking library, which supports the check of a subset of PSL in continuous time during simulation. The assertion checker fully supports the DE timing model. This means, that timing constraints of PSL operators can be defined in continuous time and are evaluated on occurring events. The assertion checking library is connected to the model using a special DE model element.

To support the analog parts of the properties comparators are integrated within the model. This is the same as using macros within other simulation environments. The additional elements influence the simulator in finding trigger points more precisely. A comparator triggers an event, if the comparator state changes. The discrete and temporal logic parts of the assertions are mapped onto PSL statements, which are then evaluated by the model library.

C. Definition of Assertions

Several formal assertions representing exhaustive verification of the Σ/Δ-converter were proved. First, we checked non-reachable states of the output signal $\hat{\mathbf{x}}(t_n)$ for the given input signal $x(t)$. The corresponding PSL assertion we used predicates that if an input signal value $x(t)$ is in the range [-0.7, 0.7], the output signal $\hat{\mathbf{x}}(t_n)$ never achieves the maximum value within the clock delay $T_s = 0.05$ seconds related to sampling frequency f_s. Assuming a 2's complement representation for the bit vector, the resulting assertion is given by:

$$
\begin{array}{ll}
1: \{x(t) < 0.7 \& x(t) > -0.7\} \mapsto & 1 \\
\quad \{next\_e[1:2]!((\hat{x}_0(t_n) \& \hat{x}_1(t_n) \& \hat{x}_2(t_n) \& \hat{x}_3(t_n) \& & 2 \\
\quad \hat{x}_4(t_n) \& \hat{x}_5(t_n) \& \hat{x}_6(t_n) \& !\hat{x}_7(t_n)) || & 3 \\
\quad (!\hat{x}_0(t_n) \& !\hat{x}_1(t_n) \& !\hat{x}_2(t_n) \& !\hat{x}_3(t_n) \& !\hat{x}_4(t_n) \& & 4 \\
\quad !\hat{x}_5(t_n) \& !\hat{x}_6(t_n) \& \hat{x}_7(t_n)))\} & 5
\end{array}
$$

Next, we proved the response when the given input signal reaches the maximum value $x(t) = 0.891$. The discretized output signal $\hat{\mathbf{x}}(t_n)$ has to reach the maximum value represented by a 1 for all non sign bit allocations and vice versa for the minimum input value. These two properties are given by the assertions 2 and 3.

2: $\{next\_a[0.0:0.05](x(t) \geq 0.8)\} \mapsto$
$\{next\_e[2](\hat{\mathbf{x}}(t_n) = \hat{x}_0(t_n) \& \hat{x}_1(t_n) \& \hat{x}_2(t_n) \&$
$\hat{x}_3(t_n) \& \hat{x}_4(t_n) \& \hat{x}_5(t_n) \& \hat{x}_6(t_n) \& !\hat{x}_7(t_n)\}$
3: $\{next\_a[0.0:0.05](x(t) \leq 0.8)\} \mapsto$
$\{next\_e[2](\hat{\mathbf{x}}(t_n) = !\hat{x}_0(t_n) \& !\hat{x}_1(t_n) \& !\hat{x}_2(t_n) \&$
$!\hat{x}_3(t_n) \& !\hat{x}_4(t_n) \& !\hat{x}_5(t_n) \& !\hat{x}_6(t_n) \& \hat{x}_7(t_n)\}$

The especialness of these assertions is the continuous time given in the analog precondition by time intervals. This assertions can be transformed into digital conform assertions by replacing the real value time interval into a clock based expression shown in assertion 2a and 3a.

2a: $\{next\_a[1](x(t) \geq 0.8)\} \mapsto$
$\{next\_e[2](\hat{\mathbf{x}}(t_n) = \hat{x}_0(t_n) \& \hat{x}_1(t_n) \& \hat{x}_2(t_n) \&$
$\hat{x}_3(t_n) \& \hat{x}_4(t_n) \& \hat{x}_5(t_n) \& \hat{x}_6(t_n) \& !\hat{x}_7(t_n)\}$
3a: $\{next\_a[1](x(t) \leq 0.8)\} \mapsto$
$\{next\_e[2](\hat{\mathbf{x}}(t_n) = !\hat{x}_0(t_n) \& !\hat{x}_1(t_n) \& !\hat{x}_2(t_n) \&$
$!\hat{x}_3(t_n) \& !\hat{x}_4(t_n) \& !\hat{x}_5(t_n) \& !\hat{x}_6(t_n) \& \hat{x}_7(t_n)\}$

Further, we proved three properties at the monitor point $\hat{s}(t_n)$. Considering that the input signal $x(t)$ achieved the maximum value $x(t) = 0.891$, the bit stream $\hat{s}(t_n)$ should be somewhere during the next 10 clock cycles for 0.025 seconds (equivalent to 4 clock cycles) at a high level and change than for 6.25×10^{-3} seconds (equivalent to 1 clock cycle T_{os}) to the lower level related to the sampling frequency f_{os}. For an input signal of $x(t) = -0.891$ the signal $\hat{\mathbf{x}}(t_n)$ behaves inverse. Another special case is: when the input signal achieve $x(t) = 0$, as a consequence the monitor point $\hat{s}(t_n)$ stays for 6.25×10^{-3} seconds (equivalent to 1 clock cycle) at high level and changes than for 6.25×10^{-3} seconds to a lower level somewhere during the next two clock cycles. The assertion we got is as follows:

4: $\{x(t) = 0.891\} \mapsto$
$\{next\_e[10](next\_a[4](\hat{s}(t_n)) \&$
$next\_e[4:5](!\hat{s}(t_n))\}$
5: $\{x(t) = -0.891\} \mapsto$
$\{next\_e[10](next\_a[4](!\hat{s}(t_n)) \&$
$next\_e[4:5](\hat{s}(t_n))\}$
6: $\{x(t) = 0\} \mapsto$
$\{next\_e[1:2](\hat{s}(t_n)) \& next\_e[1:2](!\hat{s}(t_n))\}$

Finally, the signal $g(t)$ is proved by the assertion 7. Signal $g(t)$ is the continuous representation of the digital signal $\hat{s}(t_n)$ amplified by the gain factor k. That means, that when signal $\hat{s}(t_n)$ is one the continuous signal $g(t)$ is $k = 1.4$ and when the signal $\hat{s}(t_n)$ is zero then signal $g(t)$ is $k = -1.4$ respectively. In assertion 7 we consider only the first case.

7: $\{\hat{s}(t_n)\} \mapsto \{g(t) = 1.4\}$

This last four assertions gives more detailed information of the Σ/Δ-converter at an intermediate signal in case of failing properties.

D. Results

Table I summarizes the results we got while proving the assertions 1-7. At the first column all desired assertions are enumerated. The associated results are listed at the other columns, showing at which clock cycle the property has been accepted or rejected. Whereas the inscription **A** stands for acceptance and **R** for rejection of the property. t_{clk} represents the digital clock cycle which can be $T_s = \frac{1}{f_s}$ or $T_{os} = \frac{1}{f_{os}}$ dependent to the assertion that is applied. The number beside **A** and **R** stands for the clock cycle in which the assertion is accepting or rejecting respectively. We demonstrate only the first 5 results we got from assertion-based verification achieved during one simulation run reacting to the given precondition. As an example the entry A,18 for assertion 3 at results column 2 means that the second result for assertion 3 is valid at the clock cycle 18 corresponding to the clock frequency f_s. The Assertion 1 to 6 are valid throughout the simulation time. The starting point of the simulation, 0.0, is not recognized, because the intersection point cannot be calculated in time prior the simulation start. Assertion 7 is invalid for the first five verification results. The reason for the invalid assertion results is the gain of the amplifier which is $k = 1.3$ inconsistent to the predefined specification. This is a counterexample which shows an internal error in the Σ/Δ-converter at signal location $g(t)$ that could not be seen considering the first 6 assertions.

TABLE I
ASSERTION VERIFICATION RESULTS

MSA	Property **A**ccept/**R**eject in t_{clk}				
	1	**2**	**3**	**4**	**5**
1 (f_s)	A,1	A,2	A,10	A,11	A,12
2 (f_s)	A,4	A,8	A,24	A,28	A,44
3 (f_s)	A,14	A,18	A,34	A,38	A,54
4 (f_{os})	A,26	A,31	A,69	A,186	A,191
5 (f_{os})	A,107	A,267	A,306	A,427	A,466
6 (f_{os})	A,81	A,161	A,241	A,321	A,401
7 (f_{os})	R,0	R,2	R,4	R,6	R,7

The waveforms corresponding to the monitoring points $x(t), \hat{s}(t_n)$, and $\hat{\mathbf{x}}(t_n)$ are depicted in Figure 12 which is divided into three parts. The part at the top shows the predefined input signal $x(t)$. The part below depicts the unfolded output signal $\hat{\mathbf{x}}(t_n)$ by the 8 bits $\hat{x}_0(t_n), \ldots, \hat{x}_7(t_n)$ The discrete bold points in the waveform can be identified as the digital clock related to the sampling frequency f_s. The bottommost part depicts the intermediate signal $\hat{s}(t_n)$, which is a one bit signal toggling between the values -1 and 1 characterizing the Boolean values *low* and *high*. The related clock corresponding to the sampling frequency f_{os} can be identified as discrete points in the waveform. Figure 12 shows exemplarily the results for assertion 1 we applied. The precondition (*AnalogPrecondition*) defined an input area of the input signal $x(t)$ between -0.7 and 0.7, which is marked with the dashed box. Because of the periodicity of the input signal the area of the precondition is enhanced periodically (for a better viewing not shown). The postcondition (*DigitalPostcondition*) is defined with a time delay of one clock cycle ($T_s = \frac{1}{f_s}$) corresponding to the signal $\hat{\mathbf{x}}(t_n)$. The time delay is viewed by the time-displaced dashed box within the middle picture part. For the analysis of assertion 1 the time-displaced dashed box needs to be considered. It is shown that the postcondition

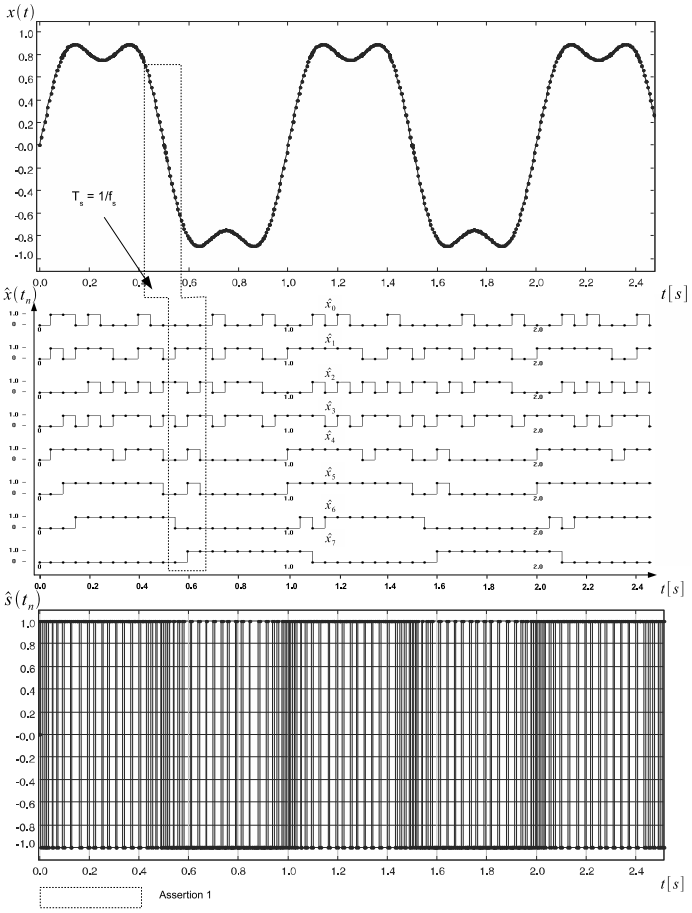

Fig. 12. Waveforms of the used monitor points within the Σ/Δ-converter and example of the result of assertion 1 (dashed box).

is valid in the whole time-displaced dashed box.

This is a general method for constructing mixed-signal assertions which can be used for all steps in the mixed-signal verification flow. The basic approach allows to define a lot of assertions of interest to verify the design by simulation and later to verify the complete range of the assertions with formal verification tools. The advantages of this method are

- MSA can be used for functional and formal verification,
- Better monitoring and controling of analog/digital interactions,
- Check formal properties during the simulation,
- Mixed-signal assertions are based on the industry standard PSL.

VI. Conclusion and Future Work

This paper proposes a new advanced method to define mixed-signal assertions for verification. This method is based on the specification of assertions with an analog and a digital part to include this in both parts of the design. These efficiently generated assertions allow to close the gap in the verification process between the interfaces of analog and digital parts of a mixed-signal system. The Σ/Δ converter example shows that the new method is applicable in the real world of the industry assertion-based verification flow. The simulation is improved by observer assertion during the simulation time. Hence, the proposed mixed-signal assertion generation can be used independently to the simulation tool.

The combination of analog and digital formal assertions makes it possible to apply the same assertions to the functional (simulation) and formal verification flow of mixed-signal systems. This method is a step towards complete functional and formal verification of mixed-signal systems. In the future we will apply this method to formal verification and to the interface verification between the analog and digital blocks of mixed-signal systems.

Acknowledgments

This work has been supported by BMBF/edacentrum project FEST under No. 01M3072C.

References

[1] *MLDesigner Documentation Version 2.5*, MLDesign Technologies, Inc., July 2004, http://www.mldesigner.com.

[2] H. D. Foster and A.C. Krolnik and D.J. Lacey, *Assertion-Based Design (2nd Edition)*. Kluwer Academic Publishers, 2004.

[3] *IEEE Standard for Property Specification Language (PSL), Version 1.1 Standard IEEE 1850*, Design Automation Standards Committee, October 2005, http://www.eda.org/vfv.

[4] Thomas Kropf, *Introduction to Formal Hardware Verification*. Springer, 1999.

[5] *IEEE Standard for SystemVerilog Unified Hardware Design, Specification, and Verification Language Standard IEEE 1800*, Design Automation Standards Committee, November 2005, http://www.eda-stds.org/sv/.

[6] W. Hartong, L. Hedrich, and E. Barke, "Model Checking Algorithms for Analog Verification," in *Proceedings of the 39th conference on Design automation (DAC'02)*, June 2002, pp. 542–547.

[7] T. A. Henzinger, P.-H. Ho, and H. Wong-Toi, "HYTECH: A Model Checker for Hybrid Systems," *International Journal on Software Tools for Technology Transfer*, vol. 1, no. 1-2, pp. 110–122, 1997.

[8] E. Asarin, T. Dang, and O. Maler, "d/dt: A Tool For Reachability Analysis Of Continuous And Hybrid Systems," in *5th IFAC Symposium Nonlinear Control Systems (NOLCOS'01)*, July 2001.

[9] A. Jesser, M. Wedler, L. Hedrich, and W. Kunz, "A case study on applying bounded model checking to analog circuit verification," in *9. GI/ITG/GMM-Workshop Methoden und Beschreibungssprachen zur Modellierung und Verifikation von Schaltungen und Systemen (MBMV'06)*, Februar 2006, pp. 106–113.

[10] S. Little, D. Walter, N. Seegmiller, C. Myers, and T. Yoneda, "Verification of Analog and Mixed-Signal Circuits Using Timed Hybrid Petri Nets," in *Automated Technology for Verification and Analysis (ATVA'04)*, November 2004, pp. 426–440.

[11] D. Grabowski, D. Platte, L. Hedrich, and E. Barke, "Time Constrained Verification of Analog Circuits using Model-Checking Algorithms," in *Proceedings of the First Workshop on Formal Verification of Analog Circuits (FAC'05)*, ser. 3, vol. 153, April 2005, pp. 37–52.

[12] O. Maler and A. Pnueli, "Extending PSL for Analog Circuits," Research Report, PROSYD Deliverable D 1.3/1, Tech. Rep., 2005.