Using Haloes in Mixed-Signal Assertion Based Verification

Dogan Ulus
Department of Electrical and Electronics Engineering
Bogazici University
Istanbul, Turkey
Email: dogan.ulus@boun.edu.tr

Alper Sen
Department of Computer Engineering
Bogazici University
Istanbul, Turkey
Email: alper.sen@boun.edu.tr

***Abstract*—We develop an assertion based verification solution for analog mixed-signal designs. We introduce the halo concept for analog signals to express them with their tolerance and variation values in assertions. The halo of a signal provides a relaxation over the signal and it defines an effective region for that signal which can be used in assertion based verification. Using haloes for analog signals allow us to define a new set of comparison relations between two analog signals including their equivalence. In our intended design flow, these new analog signal operators are placed into the *Analog* layer of mixed-signal assertion languages as an extension. We present experimental results on a programmable switch and a VCO.**

I. Introduction

As electronic market drivers push for more functionality in more compact devices, level of system complexity has sky-rocketed and becomes unmanageable without a significant support from electronic design automation (EDA) industry. Despite many ingenious solutions from EDA industry, the ever-increasing pressure for larger and complex designs makes electronic design process more vulnerable to faults. Therefore, different verification methodologies has been studied extensively in the past several years. Formal approaches like model and equivalence checking have been developed to verify designs to get a full verification coverage. However, since these methods suffer from the state-explosion problem, less-formal or semi-formal approaches have been gaining popularity in the industry. Property-based or assertion-based verification is an important part of the verification flow.

Verification methodologies are primarily developed for designs in digital domain. However, many chips designed today do not only consist of digital but also big analog and radio-frequency (RF) modules. The portion of such modules in total design has been increasing steadily over the years. Traditional methods, which are mostly manual and ad-hoc, to verify mixed-signal designs are too slow and error-prone to satisfy today's highly ambitious time-to-market and first-pass-silicon requirements. For this reason, mixed-signal verification needs a more structured and formal methodology to catch up with the efficiency of digital verification.

Assertion-based verification (ABV) methodology for analog mixed-signal designs is as an alternative to traditional methods and it tries to reduce the manual effort and increase reusability of the verification process by formalizing the specification of the design and the evaluation of simulation results. Such an automation is performed by specifying system properties in a formal language or an assertion language. Then, simulation results are checked according to this formal specification and automatically it is decided whether the property is satisfied or not.

Mixed-signal assertion languages have been developed in recent years [1], [2], [3]. These languages are specified as a set of specification layers where the top layer is the *Temporal* layer, then comes the *Boolean* layer, and *Analog* layer. Although the *Temporal* and *Boolean* layers are well understood as they are similar to their digital counterparts, the analog layer is still lacking expressiveness that the designers require. Our contributions in this paper are on *Analog* layer in order to express the analog signals more naturally. As analog signals usually associated with some tolerance and variation values in real world, operations over these signals should consider these facts. We introduce the *Halo* concept to express tolerance and variation values of analog signals in assertions. Essentially a halo for an analog signal is a pair of boundary signals. The region between these boundaries corresponds acceptable values for the value of the analog signal. We show that the concept of haloes is useful to express analog signals in a more natural way in assertions.

The paper is organized as follows. We briefly explain general characteristics and the structure of assertion languages for mixed signal designs in Section II. In Section III, we introduce the *Halo* concept. In Section IV, methods to calculate a halo are explained. Booleanization of analog signals for real-valued time are given in Section V and comparison operators for haloes are defined in Section VI. We present two case studies, one programmable switch and one VCO model, to show our intended verfication flow in Section VII. Finally, we give a related work in Section VIII, and conclude with a discussion and future work in Section IX.

II. Mixed Signal Assertion Languages

In existing assertion-based verification (ABV) flows, verification begins with system requirements. The verification engineer translates these requirements into a formal assertion language to check them automatically on simulation traces. Proposed assertion languages in the literature such as [1], [2],

978-1-4673-2899-9/12/$31.00 ©2012 IEEE

[3] are usually based on *Linear Temporal Logic* (LTL) [4] and they extend its expressiveness towards continuous-time domain.

These mixed-signal assertion languages consist of several abstraction layers and are influenced by *Property Specification Language* (PSL) [5]. Table I shows a grammar for such mixed-signal assertion language. Layers are vertically placed and the information is passed upwards through the hierarchy. In addition to PSL layers such as *Boolean* and *Temporal* layers, mixed-signal assertion languages introduce a new layer to capture analog properties. This *Analog* layer is responsible for extracting information from analog signals and converting them into boolean signals. For example, one may ask if the voltage value of an analog signal is greater than 5V, and then the *Analog* layer checks for instants that the condition is satisfied and returns a boolean signal according to this extracted information. After analog signals are converted into boolean signals, one can apply Boolean operations on them and check whether they satisfy some temporal properties or not. The upper-level *Boolean* and *Temporal* layers are meant to perform these operations in the verification flow.

In the analog layer, extracting information from analog signals can be done in several ways. Comparing analog signals with other analog signals or constant values is the most common way and it is the simplest method to convert analog signals into boolean signals in practice. Therefore, early ABV methodologies for mixed-signal designs only support comparison operators in their *Analog* layers [1], [6], [7], [8], [9]. Also, a limited set of frequency domain properties have been checked using FFT to extract information from the signals [2].

In this paper, we extend the *Analog* layer in a new direction. We introduce haloes for analog signals to express their tolerance and variation values. Our proposed extension for the *Analog* layer of mixed-signal assertion languages are shaded in Table I, which shows our proposed grammar.

In the table, different symbols are used to represent different types of operators. These operators are:

- Temporal Operators, $\circledcirc \in \{G, F\}$
- Binary Boolean Operators, $\bullet \in \{\wedge, \vee, \Rightarrow\}$
- Boolean Negation Operator, $\neg$
- Halo Comparison Operators, $\boxdot$
- Halo Calculation Operator, $\mathcal{H}$
- Arithmetic Operators, $\odot \in \{+, -, *, /\}$

The temporal and boolean operators have their usual semantics and we define the halo operators in the next sections. An arbitrary property using our grammar is as follows:

$$G((p \wedge q) \Rightarrow F(G(\mathcal{H}(x)\ SGT\ \mathcal{H}(y + c))))$$

where p and q are Boolean signals, x and y are analog signals, c is a constant and SGT is a halo comparison operator.

III. Signal Halo Concept

In astronomy a halo is defined as a circular band of colored light visible around the sun or the moon. As a naive

TABLE I
The AMS Assertion Language Grammar

Temporal Layer	$TempExpr$	::=	$\circledcirc BoolExpr$
		|	$\circledcirc TempExpr$
		|	$TempExpr \bullet TempExpr$
		|	$\neg TempExpr$
Boolean Layer	$BoolExpr$	::=	$HaloExpr \boxdot HaloExpr$
		|	$BoolExpr \bullet BoolExpr$
		|	$\neg BoolExpr$
Analog Extension	$HaloExpr$	::=	$\mathcal{H}(SignalExpr, \text{params})$
		|	$SignalExpr$
Analog Layer	$SignalExpr$	::=	$AnalogSignal$
		|	$Const$
		|	$SignalExpr \odot SignalExpr$

$\circledcirc$ Temporal Operators — $\odot$ Arithmetic Operators
$\bullet$ Binary Boolean Operators — $\boxdot$ Halo Comparison Operators
$\neg$ Boolean Negation Operator — $\mathcal{H}$ Halo Calculation Operator

interpretation, a halo is a kind of boundary for the object at the center that determines its effective region. Translating this notion into the analog signal domain, we define a pair of boundaries (upper and lower) for each sample of the analog signal we're interested in and name this as the *Signal Halo* or just the *Halo*, in short. Further, the area covered by these boundaries in two-dimensional space of analog domain is called as the *Halo Region* in the rest of the paper. To visualize the halo concept, an arbitrary analog signal and its halo are plotted in Figure 1. The methods to calculate such a halo for an analog signal are explained in Section IV.

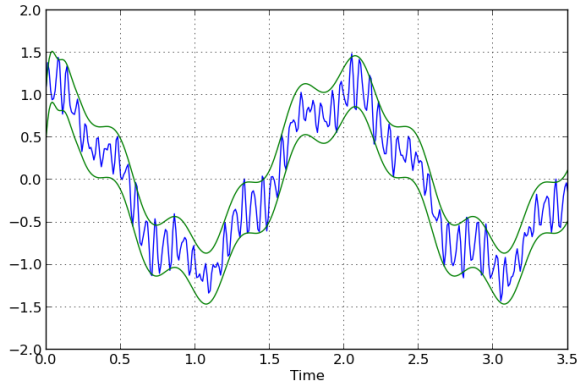

Fig. 1. An analog signal and its halo are plotted. The area between green lines is called the halo region. The noisy analog signal has been low-pass filtered and shifted up and down by a constant value to determine the halo.

A halo defines a tolerated region around the original analog signal; therefore, provides a space to relax analog operations and allows to assert more realistic properties by considering the analog nature of signals. The most natural example would be the equivalence of two analog signals. Normally, for an analog designer, it is easy to compare two analog signals – where one signal corresponds to the expected theoretical signal and another one corresponds to the observed signal from simulation– and say that they are both equivalent to each other by visually inspecting two waveforms. On the

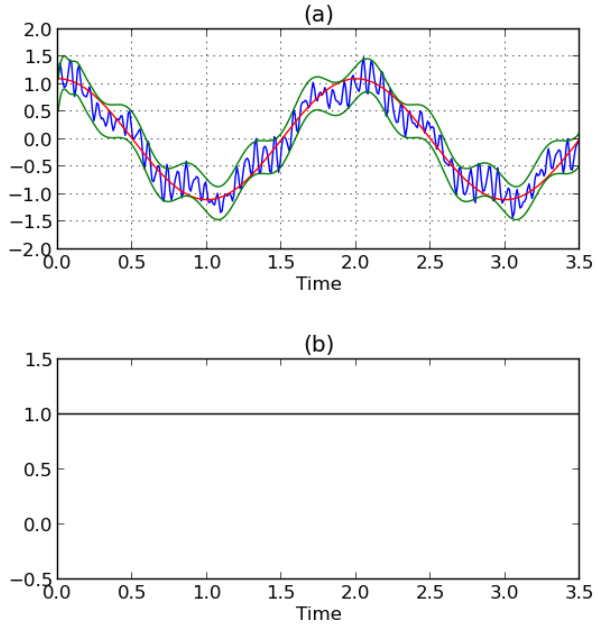

Fig. 2. A sinusoidal analog signal with noise is compared with a reference sinusoidal signal with the same frequency but amplified by a factor of 1.1. The halo region covers the reference signal and the equivalence operator outputs a boolean signal that shows that the signals are equivalent.

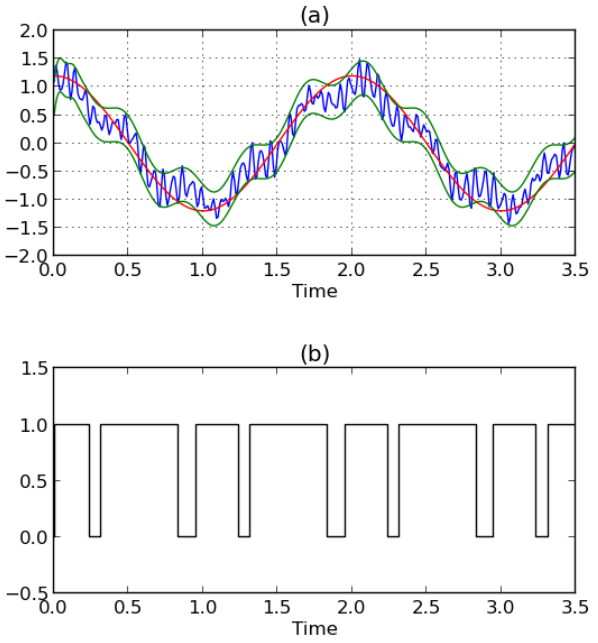

Fig. 3. A sinusoidal analog signal with noise is compared with a reference sinusoidal signal with the same frequency but amplified by a factor of 1.2. The halo region does not cover the reference signal for some points and the equivalence operator outputs a boolean signal that shows that the signals are not always equivalent.

other hand, for a computer, concluding the same result is not that straightforward. To do this, the algorithm should ignore small differences between signals while ensuring the general pattern. In other words, expressing analog signals together with tolerances has an important role in the success of automated analog verification and preventing false negatives that lead to long debugging efforts. Defining a halo provides a way to realize such a relaxation over analog signals.

Two more examples are shown in Figure 2 and Figure 3 to demonstrate the usefulness of the halo concept. In part a) of these examples, an analog signal, shown in blue, is composed of a main frequency component and several side components. Then, we want to compare the analog signal with a reference signal, shown in red, that is a sinusoidal at the frequency that equals to the analog signal's main frequency. The only difference between the first example and the second is the amplitude of reference signals plotted. Practically, a computer is too precise to conclude that the blue signal and the red signals are equivalent. However, we define haloes around the signals, shown in green, and compare using the haloes. In Figure 2, the reference signal always remains inside the halo region of the analog signal; so the output Boolean equivalence signal in part b) has a true value for each time instant and we can conclude that the signals are equivalent for that period, as the designer would conclude. However, in Figure 3, the reference signal leaves the halo region for some time instants so we cannot say that the signals are equivalent.

We relaxed the signal comparison up to some point and introduced a tolerated region for signal equivalence via a halo. This relaxed behavior of equivalence due to haloes is closer to the understanding of an analog designer, when he checks for equivalence of signals with certain tolerances in mind rather than a stricter comparison that leads to false negatives.

IV. Calculating The Halo

A halo is basically a pair of boundary signals (upper and lower) for analog signals. Since analog signals are represented by their sampled versions for verification in practice, we calculate a pair of boundary values for each sample point of the analog signal to calculate a halo.

Computing boundary values for an analog signal may be done in various ways. We propose four methods for computing boundary values.

- Absolute Method. Adding and subtracting a constant value from the original signal value is a very intuitive and simple way to calculate the boundaries. In this case, the shape of the boundaries is exactly the same as the signal except that they are shifted up or down by the constant value.
- Relative Method. For some signal types and applications, absolute tolerances may not be desired. Instead, we may want high precision around zero and less precision for values far from zero. To provide such a behavior, boundary value is calculated with a relative value given as a percentage.

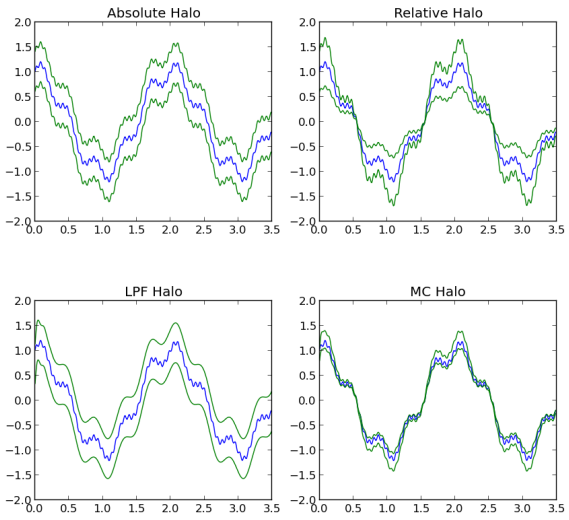

Fig. 4. An analog signal (blue) and its haloes (green) calculated by different methods

- Low-Pass Filter (LPF) Method. Absolute and Relative methods calculate the boundaries for each sample individually but we may want that the boundary values only represent the general pattern of the signal rather than individual sharp changes in the signal that is usually caused by noise. To provide such a behavior, we use a low-pass filtered version of the original signal as the base for other methods.
- Monte-Carlo Method. A number of analog signals obtained by Monte-Carlo simulations are used to calculate a halo. Simply maximum and minimum values of these signals for each sample point are considered as upper and lower boundaries, respectively.

In Figure 4, we show an arbitrary signal and its haloes calculated using four different halo calculation methods.

V. Booleanization of Analog Signals

We define analog (real-valued or continuous) signals as functions from the time-domain $\mathbb{T} = \mathbb{R}^+$ to a state-space $\mathbb{X} \subseteq \mathbb{R}^n$, where n is the number of predicates, as in [1]. The *Booleanization* of an analog signal corresponds to a transformation of the state-space $\mathbb{X}$ of the signal into Boolean values using a predicate such that $\mu : \mathbb{X} \to \mathbb{B}$. After using such predicates over analog signals, the analog signal $a : \mathbb{T} \to \mathbb{X}$ is transformed into a Boolean signal $w : \mathbb{T} \to \mathbb{B}$.

Rise and *Fall* events in Boolean signals correspond some qualitative changes in analog signals and a Boolean signal can be fully identified by its *Rise* and *Fall* events. Therefore, a Boolean signal is nothing but a sequence of such events. In our implementation, each Boolean event (*Rise* and *Fall*) in a Boolean signal is associated with a time-stamp. We prefer to use continuous-time notion to express time. This means that *Rise* and *Fall* events can happen at any time and it is independent from any type of clocking mechanism. The benefits of such an approach is presented in [10].

Although we want to express Boolean signals in continuous-time, analog signals are represented by their sampled versions in practice. This means that a predicate that we use to booleanize an analog signal should return a continuous-time boolean signal from a sampled analog signal. For such a duty, we implement a predicate which determines if an analog signal is *Greater-than-zero* or not and returns a continuous-time Boolean signal. This is a base method for us, and all other comparison operators can be derived from this method. This method sequentially checks each sample value of the analog signal, and searches for a change in the output value. Whenever a change happens, it places the corresponding event (either *Rise* or *Fall*) at that time instant. For example, if the current signal sample value is positive, and the previous sample value is negative, this means that the output value of *Greater-than-zero* operator is changed at some time between the current and the previous samples. We use linear interpolation to calculate the exact zero-crossing time, denoted by $t_{ZeroCross}$, and we place a *Rise* event at the zero-cross instant. Similarly, we place a *Fall* event to the zero-cross instant if the previous sample value is positive, and the current sample value is negative. If there is no change at the output, no event is placed. All possible cases for successive analog samples a_n and a_{n+1}, where n denotes sample index. The corresponding time-stamp and event are displayed in Table II.

VI. Halo Comparison

After we obtain haloes for analog signals, we need to compare these haloes for the booleanization of the analog signals. However, this is different from an ordinary comparison because a halo has two values to be compared rather than a single value. Furthermore, a halo divides the space into three regions (upper, inner and lower) unlike an analog signal that divides it into two regions (upper and lower). This difference makes comparison of haloes more complex, and we obtain six different possibilities of comparison.

Let a_1 and a_2 be two sampled analog signals, S_1 and S_2 be the haloes for a_1 and a_2, respectively. Haloes S_1 and S_2 are represented by tuples (u_n, l_n) for each sample point, n. Real numbers, u_n and l_n correspond to the upper and lower boundary values of the halo, S_n, where $u_n \geq l_n$. Then, we can define six comparison relations between two haloes, S_1

TABLE II
Computation a Boolean signal from sample values

a_n	a_{n+1}	Timestamp	Event
–	+	$t_{ZeroCross}$	Rise
–	0	N/A	No Event
–	–	N/A	No Event
0	+	t_n	Rise
0	0	N/A	No Event
0	–	N/A	No Event
+	+	N/A	No Event
+	0	t_{n+1}	Fall
+	–	$t_{ZeroCross}$	Fall

and S_2. We named these six relations as *strictly-greater-than* (sgt), *nearly-greater-than* (ngt), *covers* (cvr), *is-covered* (cvd), *nearly-less-than* (nlt), and *strictly-less-than* (slt).

Each relation is calculated by comparing the upper and the lower boundaries of the first halo with the upper and the lower boundaries of the second halo. Although a single comparison may be enough to define some halo comparison relations (e.g. sgt), we show all comparisons for the sake of completeness.

$$sgt = (u_1 \geq u_2) \wedge (u_1 \geq l_2) \wedge (l_1 > u_2) \wedge (l_1 > l_2)$$

$$ngt = (u_1 \geq u_2) \wedge (u_1 \geq l_2) \wedge (l_1 \leq u_2) \wedge (l_1 > l_2)$$

$$cvr = (u_1 \geq u_2) \wedge (u_1 \geq l_2) \wedge (l_1 \leq u_2) \wedge (l_1 \leq l_2)$$

$$cvd = (u_1 \leq u_2) \wedge (u_1 \geq l_2) \wedge (l_1 \leq u_2) \wedge (l_1 \geq l_2)$$

$$nlt = (u_1 < u_2) \wedge (u_1 \geq l_2) \wedge (l_1 \leq u_2) \wedge (l_1 \leq l_2)$$

$$slt = (u_1 < u_2) \wedge (u_1 < l_2) \wedge (l_1 \leq u_2) \wedge (l_1 \leq l_2)$$

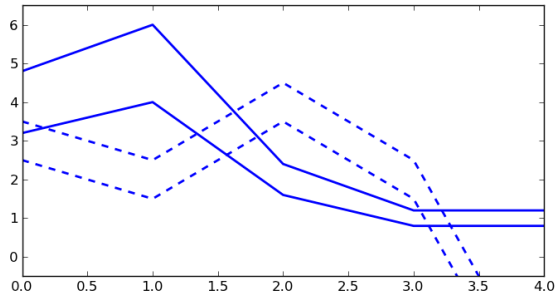

Fig. 5. Two haloes with five sample-points

In Figure 5, we display two haloes –straight lines and dashed lines– each with only five sample-points for analog signals. We apply halo comparison operators to these haloes and the results are plotted in Figure 6.

VII. Case Studies

We show the viability of our approach for AMS verification on two case studies. As the first case study, we use a programmable switch circuit with local memory. The property that we want to check is that the output of the circuit should follow the input if the memory value is logic high.

Simulation waveforms, $a : in$, $a : out$ and $a : on$ correspond to the input, output and the memory value, respectively. We show these waveforms in the first three plot of Figure 7. We convert the analog waveform, $a : on$, to a boolean signal, $b : on$, via a threshold which is $V_{dd}/2$. This booleanization is essentially a comparison-with-zero operation, and it can be formally written as $b : on = (a : on - V_{dd}/2 > 0)$. For our case, $V_{dd} = 2.5V$.

For the next step, we calculate a halo for the input signal, $a : in$ because we want to check if the output signal, $a : out$, stays in the boundaries of the input halo. By this way, we can say if the output follows the input or not. In our case, we calculate

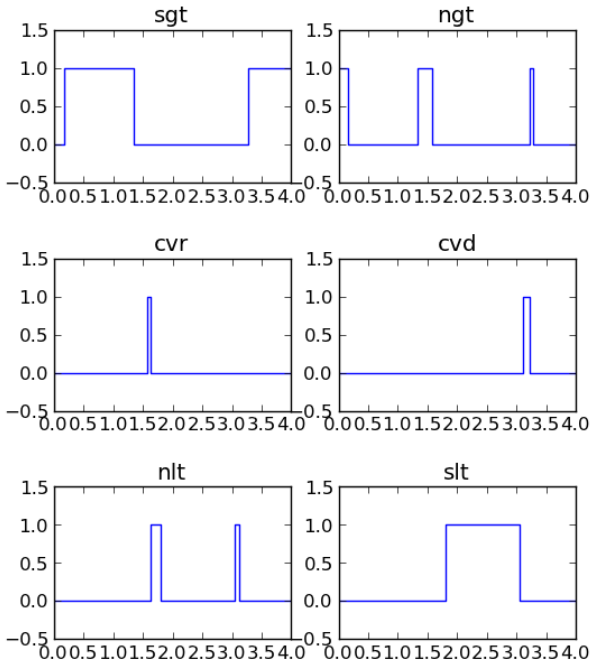

Fig. 6. Comparison of haloes in Figure 5 using halo comparison operators

the input halo using Absolute method with 0.01V margin. It means we allow a constant tolerance of ± 0.01 around the input signal. Note that the output signal is exactly the same with input signal in the ideal world. However, it is never the case in the real life; therefore, we should allow small differences, and we are doing such a relaxation by using haloes.

Ultimately, we write the desired property in our assertion language as follows:

$$G(b : on \Rightarrow (a : out\ CVD\ \mathcal{H}(a : in)))$$

where G is *Always* operator of temporal logic, CVD is our *covered-by* operator and $\mathcal{H}$ is our halo calculation procedure. In English, we can write this property such: For all time instants, $a : out$ is *covered by* the halo of $a : in$ when $b : on$ is True.

In Figure 7, we show evaluation steps for the property above and the property is satisfied in this case. First three plots are the waveforms coming from the simulator. In fourth, we show the booleanization of $a : on$ signal via a threshold. In fifth, we show $a : in$ (blue), $\mathcal{H}(a : in)$ (green) and $a : out$ (red) in a combined and zoomed view. In this plot, we can see the output signal, $a : out$, is inside the halo region of the input signal, $a : in$. Following three plots show the evaluation of sub-formulas in the property.

In the second case, we use a voltage-controlled oscillator (VCO) model with linear, quadratic, and cubic gain parameters. The higher order quadratic and cubic effects are meant to model the non-linearity in real designs; however, non-linearity

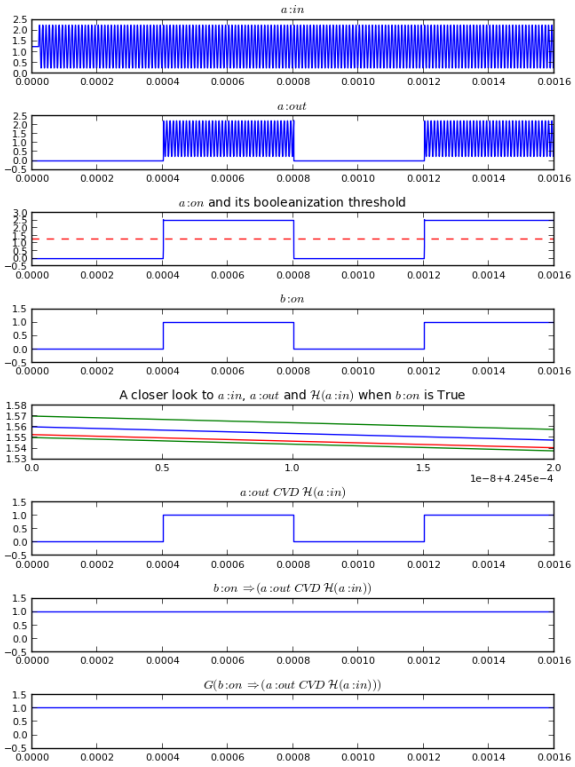

Fig. 7. Step-by-step property evaluation for programmable switch

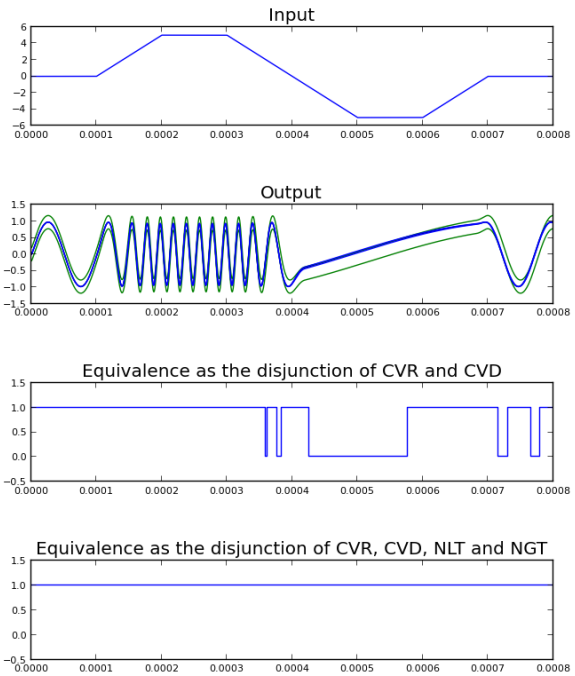

Fig. 8. VCO property checking

is undesired for VCO operation. Therefore, we simulate the model without these higher order effects to get our golden reference VCO output signal. Then, we perform Monte Carlo simulation by varying quadratic and cubic gains to generate a halo for the VCO output signal. Halo generation by Monte Carlo simulations is explained in Section VI. Generated halo then can be used in assertions to check if the desired property is satisfied or not. In the first plot of Figure 8, we show the VCO input analog signal. In the second plot, we show the halo of the VCO output signal (out) obtained through Monte Carlo method and the halo of the reference VCO output signal (ref), shown as green signal. In the third and fourth plots, we test whether $((out\ CVR\ ref) \vee (out\ CVD\ ref))$ and $((out\ CVR\ ref) \vee (out\ CVD\ ref) \vee (out\ NLT\ ref) \vee (out\ NGT\ ref))$, respectively. These properties mean different equivalence specifications between signals in time domain.

This example show that we can use haloes to express process variations over signals, and we can check if there is a case that process variation causes a violation in the specification given.

VIII. Related Work

Several approaches have been proposed in the literature for the verification of analog mixed-signal designs. Formal approaches like [11], [12], [13], [14], [15], [16] and semi-formal approaches like [1], [2], [3], [17] are two main branches in the field. Mathematical origins of these approaches have been studied over decades but practical studies are newly emerging in both academia and industry.

Maler and Nickovic [18], [19] presented *Signal Temporal Logic*, STL, to monitor temporal properties of continuous signals by extending *Metric Interval Time Logic*, MITL [10]. It checks properties in an offline fashion after the simulation is completed. They extend the checking process to a semi-online version in [20]. They also developed an Analog Monitoring Tool [21]. The tool checks properties defined in STL over simulation traces and it works seamlessly with various industry-standard waveform formats. However, although temporal layer in STL is based on real-time and it provides expressive constructs for verification, STL's analog layer is limited to basic arithmetic operations and analog signals are converted to booleans with basic comparison operators. This reduces the overall quality of analog verification.

As a part of another research effort, Lammermann et al. [2] present an online temporal checker module as SystemC-AMS library. It uses clock-based time notion rather than true real-time and adds some dedicated operators for analog layer. They use Fast Fourier Transform (FFT) to check frequency domain properties from simulation traces. This is a good direction because time domain specifications are only a part of analog verification that designers use.

Accellara has a working group to extend System Verilog Assertions (SVA) with analog extensions and to create an assertion language called ASVA [7]. EDA vendors like Ca-

dence and Synopsys present AMS assertions to increase the verification capability of their AMS design flows [8], [9].

IX. Conclusion

We introduce the halo concept for analog signals. A halo provides a way to express analog signal tolerances and variations. Using haloes in assertions instead of the signal itself provides a more relaxed property-checking, and makes assertions variation-aware. This is a natural extension of the analog designers approach in verification.

We define a new set of comparison operations for haloes and use them in assertions. These new operators extend the *Analog* layer of mixed-signal assertion languages, and can be used in existing verification flows.

We propose four methods to calculate a halo for an analog signal, namely Absolute, Relative, Low-Pass filter, and Monte Carlo methods. We performed two case studies using haloes.

In the future, we plan to use haloes during the process of booleanization and perform further experiments.

Acknowledgment

This work was supported in part by Semiconductor Research Corporation under task 2082.001, Marie Curie European Reintegration Grant within the 7th European Community Framework Programme, and the Turkish Academy of Sciences.

References

[1] D. Ničković, "Checking timed and hybrid properties: Theory and applications," Ph.D. dissertation, Joseph Fourier University, 2008.

[2] S. Lämmermann, J. Ruf, T. Kropf, W. Rosenstiel, A. Viehl, A. Jesser, and L. Hedrich, "Towards assertion-based verification of heterogeneous system designs," in *Proceedings of the Conference on Design, Automation and Test in Europe (DATE)*. European Design and Automation Association, 2010, pp. 1171–1176.

[3] R. Mukhopadhyay, S. Panda, P. Dasgupta, and J. Gough, "Instrumenting AMS assertion verification on commercial platforms," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 14, no. 2, p. 21, 2009.

[4] A. Pnueli, "The temporal logic of programs," in *Foundations of Computer Science, 1977., 18th Annual Symposium on*. IEEE, 1977, pp. 46–57.

[5] H. Foster, E. Marschner, and Y. Wolfsthal, "IEEE 1850 PSL: The next generation," in *Proceedings of Design and Verification Conference and exhibition (DVCON)*, 2005.

[6] S. Mukherjee and P. Dasgupta, "Incorporating local variables in mixed-signal assertions," in *TENCON 2009-2009 IEEE Region 10 Conference*. IEEE, 2009, pp. 1–5.

[7] H. Anand, J. Havlicek, and H. Miller, "Assertion based analog mixed signal verification," Freescale, Tech. Rep., 2008.

[8] D. J. O'Riordan and P. K. Bhattacharya, "PSL/SVA Assertions in SPICE," in *Proceedings of The Design & Verification Conference & Exhibition*, 2012.

[9] G. Nunn, F. Delguste, A. Khan, A. Verma, and B. Geden, "Using Digital Verification Techniques on Mixed-Signal SoCs with CustomSim and VCS," Synopsys, Tech. Rep., 2011.

[10] R. Alur, T. Feder, and T. Henzinger, "The benefits of relaxing punctuality," *Journal of the ACM (JACM)*, vol. 43, no. 1, pp. 116–146, 1996.

[11] S. Steinhorst, "Formal verification methodologies for nonlinear analog circuits," Ph.D. dissertation, Goethe-University of Frankfurt am Main, 2011.

[12] S. Steinhorst and L. Hedrich, "Model checking of analog systems using an analog specification language," in *Proceedings of the Conference on Design, Automation and Test in Europe (DATE)*, 2008.

[13] M. Zaki, "Techniques for the formal verification of analog and mixed-signal designs," Ph.D. dissertation, Concordia University, 2008.

[14] G. Al-Sammane, M. Zaki, and S. Tahar, "A symbolic methodology for the verification of analog and mixed signal designs," in *Proceedings of the Conference on Design, Automation and Test in Europe (DATE)*. IEEE, 2007, pp. 1–6.

[15] E. Barke, D. Grabowski, H. Graeb, L. Hedrich, S. Heinen, R. Popp, S. Steinhorst, and Y. Wang, "Formal approaches to analog circuit verification," in *Proceedings of the Conference on Design, Automation and Test in Europe (DATE)*. European Design and Automation Association, 2009, pp. 724–729.

[16] S. Little, D. Walter, K. Jones, C. Myers, and A. Sen, "Analog/mixed-signal circuit verification using models generated from simulation traces," *International Journal of Foundations of Computer Science*, vol. 21, no. 2, pp. 191–210, 2010.

[17] J. Havlicek and S. Little, "Realtime regular expressions for analog and mixed-signal assertions," in *Proceedings of the Formal Methods in Computer Aided Design (FMCAD)*. IEEE, 2011, pp. 155–162.

[18] O. Maler and D. Ničković, "Monitoring temporal properties of continuous signals," *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*, pp. 71–76, 2004.

[19] O. Maler, D. Ničković, and A. Pnueli, "Real time temporal logic: Past, present, future," *Formal Modeling and Analysis of Timed Systems*, pp. 2–16, 2005.

[20] O. Maler, D. Nickovic, and A. Pnueli, "Checking temporal properties of discrete, timed and continuous behaviors," *Pillars of computer science*, pp. 475–505, 2008.

[21] D. Ničković and O. Maler, "AMT: A property-based monitoring tool for analog systems," *Formal Modeling and Analysis of Timed Systems*, pp. 304–319, 2007.