Towards Assertion Based Verification of Analog and Mixed Signal Designs Using PSL

Ghiath Al Sammane, Mohamed H. Zaki, Zhi Jie Dong and Sofiène Tahar
Department of Electrical and Computer Engineering
Concordia University, Montreal, Québec, Canada
Email: {sammane, mzaki, zh_do, tahar}@ece.concordia.ca

***Abstract*— Analog and Mixed Signal (AMS) designs are important integrated systems that link digital circuits to the analog world. Following the success of PSL verification methodologies, recent research suggested extending PSL to support AMS systems. However, PSL has been defined initially to formalize properties about logical signals using models such as automata. In fact, the presence of continuous signals in AMS systems and the continuous notion of time are the main obstacles in adopting PSL. In this paper, we propose an approach to verify PSL properties for a class of AMS systems. Our approach is based on modeling the AMS design in terms of a System of Recurrence Equations (SRE). Then, we define an Assertion Based Verification method using the symbolic trace of SRE.**

I. Introduction

With the latest advancement of semiconductors technology, circuit integration and performance paved the way to the design of system on chip (SoC) for embedded systems. A cornerstone in embedded systems are analog and mixed signal (AMS) designs which are integrated circuits that are usually needed at the interface between these system and the real world [2], [3]; by processing analog signals and converting between analog and digital data representation.

Classically, the verification of AMS designs is carried out using simulation to validate basic properties or assertions. For example, property monitoring of AMS designs is performed primarily using assertions and tests. The monitoring can be described in general as follows: The AMS design under verification is simulated by attaching it to a testbench which provides the inputs while monitoring its output. Assertions have the property that they are always checked, regardless of what tests are running. The monitor could be as simple as observing a current or voltage. The main challenge in monitoring AMS designs is the development of adequate monitors able to express the properties. With AMS systems growing more and more complex, current methodologies are not strong enough to express interesting properties such as temporal requirements.

The Accellera Property Specification Language (PSL) [1] enjoys a concise syntax supported with a well defined formal semantics, permitting unambiguous documentation and automatic verification for complex designs. In fact, PSL has changed completely how designers specify functional requirements and properties of digital systems. Today, most EDA companies are supporting PSL in their tools through formal property checking or through Assertion Based Verification (ABV). However, PSL has been defined initially to formalize properties about logical signals using models such as automata, without any support of analog signals and the continuous notion of time.

In this paper, we propose an approach to verify PSL properties for a class of AMS systems. Our approach is based on modeling the AMS design in terms of a System of Recurrence Equations (SRE) for both the analog and the digital parts of the system. Using these equations, symbolic traces of both the design and the assertions are computed. Then, we define how these traces are used to numerically simulate the design and verify PSL assertions in an offline fashion.

The rest of the paper is organized as follows: Related work is discussed in Section II. In Section 1 we give an overview of our verification methodology. A background about PSL and the SRE model is given in Section IV. In Section V, we show how to verify PSL properties using the notion of trace with an illustration on a third order $\Delta\Sigma$ modulator. In Section VI, we use our methodology to verify a PLL circuits before concluding in Section VII.

II. Related Work

According to [7], run-time verification can be classified in two different fashions : *Offline monitoring*, where the property verification starts after the whole simulation sequence is given. *Online monitoring* is interleaved with the process of reading the sequence and is similar to the way the sequence is read by an automaton. Online monitors can detect violation or satisfaction as soon as they happen. Several works have been recently proposed for run-time verification of AMS designs. The most prominent are discussed below.

In [8], an Online monitoring technique was proposed, where the authors used linear hybrid automata as template monitors for time domain features of oscillatory behavior, such as bounds on signal amplitude and jitter. For the automata with an error state, the reachability computation can be stopped as soon as this state is reachable. The monitors are used within the PHAver tool where nonlinear circuit equations are modeled with piecewise affine differential inclusions. The authors used the methodology to verify the oscillation property of tunnel diode oscillators.

In [10], the authors proposed an online monitoring methodology for analog systems. They present a run-time verification methodology based on monitoring the behavior (solution flow)

of analog circuits validated using interval analysis. Given the system description and its specification described by non-linear differential equations and timed computational temporal logic (T-CTL) formulas, respectively, the authors build a timed automata monitor which can detect bad behavior within a specified time period of the interval arithmetics simulation. They applied the methodology for the tunnel diode oscillators.

By contrast to the work we propose in this paper, the above two work consider only analog behaviors, while mixed systems are not supported. In addition, they lack any practical property specification language.

In [6], Maler *et.al* proposed an offline methodology for monitoring the simulation of continuous signals described by differential equations. The work is based on extending the PSL logic to support monitoring analog signals, by defining the syntax and semantics of metric timed linear temporal logic (MTL) and extended it with predicate over real to form the signal temporal logic (STL). MTL is then synthesized into timed automata which monitor Matlab simulation traces to check for property violation. A related idea was proposed in [9], where the authors use an extended temporal logic, AnaCTL (CTL for analog circuit verification), for monitoring the transient behavior of non-linear analog circuits. The transient response of a circuit under all possible input waveforms is represented as an FSM created by means of repeated SPICE simulations, bounding and discretizing the continuous state space of an analog circuit. Exhaustive simulation is again a drawback as the created FSM is not guaranteed to cover the total transient behavior leading to soundness problem.

The above approaches synthesize the property in terms of timed automata or in terms of FSM resulting in a detailed lower abstraction level model. We propose to use a higher level of abstraction based on recurrence equations and hence avoiding state space explosion. Besides, unlike above related work, our approach supports Sequential Extended Regular Expressions (SERE) of PSL.

III. Overview of The Methodology

An AMS design is usually described using languages like VHDL-AMS [13]. The semantics of these languages describes separately the analog behavior and the digital behavior of the design. For example, the VHDL-AMS standard specifies that in order to simulate a mixed system, a set of solvability conditions should be satisfied. Among these conditions, the analog part should be described using a set of differential-algebraic equations (DAEs) that can be converted into an equivalent set of difference equations (recurrence equations). The digital part, on the other hand, deals with discrete values or states that change only at specified times or events.

In fact, the mixed-signal simulator should consider the interactions between models represented as digital (with events) and those represented as analog (with algebraic or differential equations). The verification of properties relies on the communication model between the two behaviors of the AMS system. In our methodology, we support AMS systems that can be described using discrete recurrence equations. The model gives the possibility to handle continuous behaviors like that of current and voltages, but in discrete time intervals which covers a non trivial class of mixed behaviors. This mathematical model is more familiar to analog designers (as most discrete circuits are described using difference equations) rather than automata.

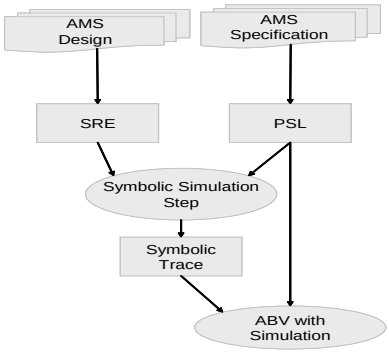

Fig. 1. Overview of the Methodology

The proposed methodology is shown in Figure 1. We start with the AMS description and we extract a System of Recurrence Equations that represent the system. In our case, we consider the properties provided in terms of an extended PSL that we introduce in the next section. Both the SRE and PSL properties are input to a special symbolic simulator that performs one simulation step. The output of this operation is the symbolic trace of the combined system (AMS model + PSL properties). This trace gives the possibility of indexing the design signals in terms of simulation time. The Assertion Based Verification (ABV) is performed using these time indexed values. Details of this approach will be explained in the subsequent sections. We believe this method to enable bridging the gap between AMS modeling and PSL verification.

IV. Technical Background

A. The Property Specification Language PSL

PSL contains four layers: Boolean, temporal, verification and modeling layer. The modeling layer is needed to define the verification environment specially for formal verification tools. The verification layer is more related to the description of verification tools where notions like assume and guarantee are present.

We limit our discussions only to discuss the Boolean layer and the temporal layer. The Boolean layer provides the core to build expressions. The temporal layer is the essence of PSL where complex temporal relations between signals can be expressed. The Foundation Language (FL) in the temporal layer express properties that can be as simple as a Boolean expression, an LTL property or SEREs. We present here only the syntax of PSL (the semantics is presented along with our verification approach later).

Definition 1: ***Syntax of Sequential Extended Regular Expressions (SEREs)***:

- if b is Boolean expression, then b is a SERE

- if r is a SERE, then $r[*]$ is a SERE (finite consecutive repetitions)
- if r_1 and r_2 are SEREs, then the following are SEREs:
  - the consecutive concatenation of two sequences, $r1;r2$
  - one-state overlapping concatenation $r1:r2$
  - disjunction of sequences $r1|r2$
  - overlapping sequences $r1\&r2$
  - length-matching sequences $r1\&\&r2$

$FL\_Property ::= Boolean|(FL\_Property)$

The LTL operators, provide Linear Temporal Logic syntax to PSL. These LTL operators form an alternative syntax for the parts of equivalent PSL operators, for example:

$$\begin{aligned} next &\Leftrightarrow X \\ eventually! &\Leftrightarrow F \\ until! &\Leftrightarrow U \\ always &\Leftrightarrow G \end{aligned}$$

B. The System of Recurrence Equations: SRE

A recurrence equation or a difference equation is the discrete version of an analog differential equation. In conventional system analysis, recurrence equations are used in the definition of relations between consecutive elements of a sequence.

In [5], the notion of recurrence equation is extended to describe digital circuits using the normal form: generalized `If-formula`.

Definition 2: **Generalized `If-formula`**
In the context of symbolic expressions, the generalized `If-formula` is a class of expressions that extend recurrence equations to describe digital systems. Let i and n be natural numbers. Let $\mathbb{K}$ be a numerical domain in ($\mathbb{N},\mathbb{Z},\mathbb{Q}$, $\mathbb{R}$ or $\mathbb{B}$), a generalized `If-formula` is one of the following:

- A variable $X_i(n)$ or a constant C that take values in $\mathbb{K}$
- Any arithmetic operation $\diamond \in \{+,-,\div,\times\}$ between variables $X_i(n)$ that take values in $\mathbb{K}$
- A logical formula: any expression constructed using a set of variables $X_i(n) \in \mathbb{B}$ and logical operators: $not, and, or, xor, nor, \ldots$ etc.
- A comparison formula: any expression constructed using a set of $X_i(n) \in \mathbb{K}$ and comparison operator $\alpha \in \{=, \neq, <, \leq, >, \geq\}$.
- An expression $IF(X,Y,Z)$, where X is a logical formula or a comparison formula and Y,Z are any generalized `If-formula`. Here, $IF(x,y,z) : B \times \mathbb{K} \times \mathbb{K} \longrightarrow \mathbb{K}$ satisfies the axioms:
  (1) $IF(True,X,Y) = X$
  (2) $IF(False,X,Y) = Y$

We define a System of Recurrence Equations as follows:

Definition 3: ***A System of Recurrence Equations (SRE)***
Consider a set of variables $X_i(n) \in \mathbb{K}$, $i \in V = \{1,\ldots,k\}$, $n \in \mathbb{Z}$, an SRE is a system of the form:

$$X_i(n) = f_i(X_j(n-\gamma)), (j,\gamma) \in \varepsilon_i, \forall n \in \mathbb{Z}$$

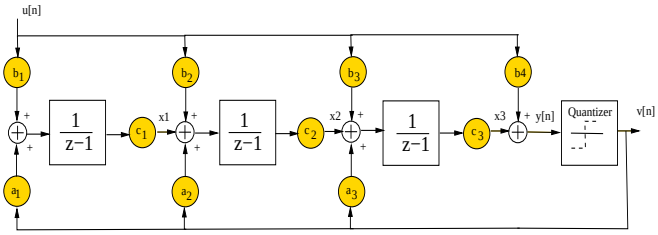

Fig. 2. Third Order ΔΣ Modulator

where $f_i(X_j(n-\gamma))$ is a generalized `If-formula`. The set ε_i is a finite non empty subset of $1,\ldots,k \times \mathbb{N}$. The integer γ is called the delay.

C. SRE Based Symbolic Simulation

Transforming Ordinary Differential Equations (ODEs) into difference equation is well known in the domain of analog circuits simulation. For the digital part, [5] has defined a method and implemented tool to extract a set of recurrence equations (difference equations) from a digital synchronized VHDL descriptions. We use the same framework to extract the recurrence equation of the digital part. The symbolic simulation algorithm is based on rewriting by substitution over the SRE. The computation aims to obtain the trace of the system as defined in [11].

Example: $3^{rd} Order \Delta\Sigma$ *modulator*

ΔΣ modulators with single-bit quantizers have made possible the construction of robust high-resolution analog-to-digital and digital-to-analog converters. The design of the ΔΣ modulator in Figure 2 is given using vector recurrence equations $X(k+1) = C\ X(k) + B\ u(k) + A\ v(k)$, where A, B and C are matrices providing the parameters of the circuit, $u(k)$ is the input and $v(k)$ is the digital part of the system.

The condition of the threshold of the quantizer is computed to be equal to $c3x3(k) + u(k)$. The digital description of the quantizer is transformed into a recurrence equation: $v(k) = IF(c3x3(k) + u(k) \geq 0, -a, a)$. The equivalent SRE of the system is then:

$$\begin{aligned} x_1(k+1) &= if(c_3x_3(k)+u(k) >= 0, x_1(k)+b_1u(k)-a_1a, \\ &\quad x_1(k)+b_1u(k)+a_1a) \\ x_2(k+1) &= if(c_3x_3(k)+u(k) >= 0, c_1x_1(k)+x_2(k)+b_2u(k) \\ &\quad -a_2a, c_1x_1(k)+x_2(k)+b_2u(k)+a_2a) \\ x_3(k+1) &= if(c_3x_3(k)+u(k) >= 0, c_2x_2(k)+x_3(k)+b_3u(k) \\ &\quad -a_3a, c_2x_2(k)+x_3(k)+b_3u(k)+a_3a) \end{aligned}$$

A ΔΣ modulator is said to be stable if the integrator output remains bounded under a bounded input signal.

$$Always\ -1 < x_3 < 1$$

V. Verification of PSL Properties

In the seek of compactness, we present the more intuitive of the checker algorithmic description for a PSL operator or its definition using SRE trace. Also, we limit our discussion for one constructor to each operator as the idea can be generalized to the variant of this constructor.

A. Writing PSL for AMS Systems

The Boolean layer in PSL is built using Boolean expressions imported from logical signals (and registers) of the digital design. However, in the case of AMS systems signals can have continuous nature where values are represented as Reals. Then, the analog equivalent to a Boolean variable in PSL is an inequations (or a predicate) that is constructed using signals and registers of the AMS system. We call this expression the *Basic Property*.

Definition 4: ***Basic Property***
Let x be the name of an AMS signal (or register), a basic property p is a logical formula defined as follows:
$p = x \diamond y$, where $\diamond \in \{<, \leq, >, \geq, =, \neq\}$ and y is a value, a name of signal (or a register) in the design or an arithmetic function built using the design signals.

For instance, a basic property of the $\Delta\Sigma$ modulator described in Section IV-C is $-1 < x_3 < 1$. The Boolean aspect of this predicate makes it possible to define the design behavior using PSL. However, the mixed nature imposes some difficulties in the verification of these properties. Our modeling in terms of recurrence equations aims to overcome these limitations by giving an explicit time representation for each property. In fact, as the design is described using a System of Recurrence Equations (SRE), we can represent each signal or register in the basic property by an equivalent time instance from the SRE. This time instance is not an abstraction of the property in terms of Boolean variables, but it is the symbolic execution trace of the system represented as recurrence relations.

Definition 5: ***Trace of the Basic Property***
Let p be a basic property. For each signal (or register) x, y in the basic property we associate a time instance of the form $x(n)$, where $n \in \mathbb{N}$. This instance corresponds to the evaluation of the recurrence equation representing this value at time n. We call $p(n)$ the trace of the basic property p. The nature of n depends on the SERE or temporal layer operator proceeding the basic property. The nature of n can be one of the following:

- Numerical constant value.
- Symbolic constant value.
- Time variable

The Temporal layer is used to specify temporal chains of events of Boolean primitives. In our approach, these Boolean primitives are replaced with basic properties.

Definition 6: ***Evaluation of the basic property***
We say that, the trace of a basic property p holds at a simulation cycle n, we write $TrueQ(p,n)$, if its trace is evaluated to True. Otherwise, the property is said to not hold at that cycle.

B. Verification Checkers of SERE

The SERE concatenation operator $(;)$ constructs a SERE that is the concatenation of two other SEREs. For two SEREs A and B: $A;B$ holds on a path iff there is a future cycle n, such that A holds tightly on the path up to and including the n^{th} cycle and B holds tightly on the path starting at the $n+1^{th}$ cycle. Using our trace notations, we write:

$$A;B := TrueQ(A,n) \wedge TrueQ(B,n+1)$$

The SERE consecutive repetition operator $([*])$ constructs repeated consecutive concatenation of the same SERE. $A[*n]$ holds tightly on a path iff the path can be partitioned into n parts, where A holds tightly on each part. We write:

$$A[*n] := \bigwedge_{i=1}^{n} (TrueQ(A,i))$$

The SERE non-consecutive repetition operator $([=])$, constructs repeated (possibly non-consecutive) concatenation of a Boolean expression. $A[=n]$ holds tightly on a path iff A occurs exactly n times along the path. The observer is defined as follows:

$J = 0$
for $i = 1$ to N_{max} **do**
  if $TrueQ(A,i)$ **then**
    $J{+}{+}$
  end if
  if $J == n$ **then**
    return $True$
  end if
end for

C. Verification Checkers of FL Properties

FL properties describe single- or multi-cycle behaviors built from Boolean or sequential expressions. The most basic FL property is a Boolean expression which is a basic property in the case of AMS systems. More complex FL properties are built from Boolean expressions, sequential expressions, and subordinate properties using various temporal operators. Here, we discuss the verification of Linear Temporal Logic (LTL) properties. However, the bounded nature of ABV makes the verification of this property limited in time.

Implication operators, specify that an FL property or sequence holds if some pre-requisite sequence holds:

$$A \Rightarrow B := \neg TrueQ(A,n) \vee (TrueQ(A,n) \wedge TrueA(B,n))$$

Next operators, specify that an FL property holds at some next cycle:

$$\mathbf{X}(p) := TrueQ(p,n+1)$$

Always operators, specify that a property holds in the current cycle of a given path iff the FL property or sequence that is the operand holds at the current cycle and all subsequent cycles. The operand of the always operator is an FL Property or Sequence:

$$\mathbf{G}(p) := \bigwedge_{k=0}^{\infty} TrueQ(p,k)$$

Eventually! operator, specifies that an FL property or a sequence holds at the current cycle or at some future cycle:

$$\mathbf{F}(p) := \bigvee_{k=0}^{\infty} TrueQ(p,k)$$

Until operators, specify that one FL property holds until a second FL property holds. An until! property holds in the current cycle of a given path iff:
1) the FL property that is the right operand holds at the current cycle or at some future cycle; and
2) the FL property that is the left operand holds at all cycles up to, but not necessarily including, the earliest cycle at which the FL property that is the right operand holds. We write:

$$p\mathbf{U}p' := \bigvee_{k=0}^{\infty} (TrueQ(p',k) \wedge \bigwedge_{j=0}^{k-1} TrueQ(p,j))$$

D. The Offline Assertion based Verification

The definitions of PSL checkers are used along with traces of both the system and properties to verify assertions using simulation. The offline algorithm is described below (Algorithm 1). In line 1, the equations are initialized by assigning the design initial values S_{Init}. The loop in lines (2-5) describe the simulation of the design for N_{max} simulation cycles. At each cycle i we assign an input to be stored in the set of cases I. In line 4, the trace of the design is evaluated using I and S_{Init} in order to obtain values $S(i)$ for all signals (and registers) of the design at time $0 < i \leq N_{max}$. After, that we use $S(i)$ along with the trace of the properties and the checkers defined above to evaluate the assertions of the design.

Algorithm 1 ABV Algorithm

Require: SRE_{trace}
Require: P_{trace}
1: $S_{Init} = Initialize\_Equations$
2: **for** $i = 0$ to N_{max} **do**
3: $\quad I := Assign\_Inputs(i)$
4: $\quad S(i) := Evaluate\_Trace(SRE_{trace}, I, S_{Init})$
5: **end for**
6: $Run\_Verification(P_{trace}, S(i), I)_{i=0}^{N_{max}}$

Example: ΔΣ Stability Property

We illustrate our verification approach on the stability property P of the ΔΣ modulator given in Section IV-C. The basic property is $-1 < x3 < 1$. First, we run a symbolic simulation step to obtain the trace of the property using the SRE of the model. We obtain the following symbolic trace for the basic property:

$$\begin{aligned} p(n) \quad = \quad & if(c_3x_3(n) + u(n) >= 0, \\ & c_2x_2(n) + x_3(n) + b_3u(n) - a_3a < 2, \\ & -2 < c_2x_2(n) + x_3(n) + b_3u(n) + a_3a) \end{aligned}$$

Next, we bend the variable n in the trace $p(n)$ to its meaning that comes from the `Always` checker: **G** (p), and we obtain

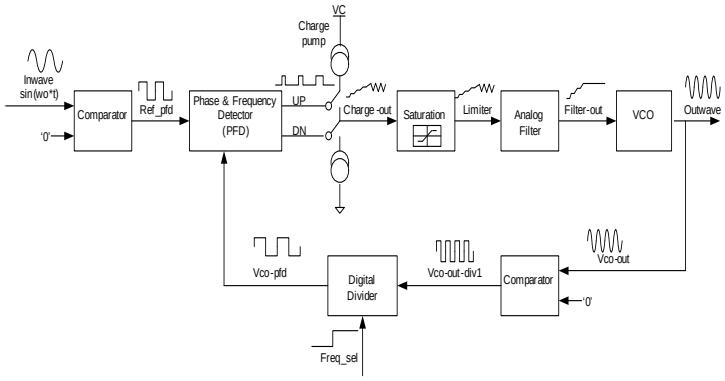

Fig. 3. PLL Frequency synthesizer diagram

$$\bigwedge_{n=0}^{\infty} TrueQ(p,n)$$

However, the verification is done using bounded time. In Table 1, we give the simulation results for this property (for chosen parameters). The results are obtained from Mathematica 5.2 [14] running on an Intel Core duo with 2GB of RAM.

TABLE I
VERIFICATION RESULTS FOR THE ΔΣ MODULATOR

Number of Cycles	100	1000	10000	10^5	10^6
CPU time (sec)	0.016	0.046	0.516	5.031	51.093

VI. APPLICATION: VERIFICATION OF A PLL DESIGN

One of the main issues in the verification of PLL designs is the lock time which is the time necessary for the PLL to switch from one frequency to another following a given frequency change. Such property is important because during the lock time, no data can be transmitted, so having a larger lock time can reduce the data rate of the system. We applied the methodology described in this paper to detect violation in the locking requirement of a frequency synthesizer based PLL design which is one of the basic building blocks in modern communication systems.

The frequency synthesizer model [12] is composed of a comparator, phase and frequency detector (PFD) module, analog loop filter, VCO and a divider modules as shown in Figure 3. The input reference signal is a *Sine wave* with frequency w_o. The output signal is a *Cosine wave* with frequency N times of w_o. For instance, if the frequency control signal *Freq_sel* is set to 0, N equals to 1 and the frequencies of input and output signals are the same. If the *Freq_sel* signal is set to 1, N equals to 2 and the output signal frequency becomes two times of the input signal.

In the simulation we run, the input Sine wave signal frequency and the VCO central frequency are 100 Hz, while the sampling time is $T = 1e-4$ seconds, RC is the electrical constant. We modeled the frequency synthesizer using the symbolic Recursive Equations. For illustration purposes and due to lack of space, we show below only sample equations

used in the modeling. For instance at the output of the PFD, we have:

$$UP(n+1) = If([ref\_pfd(n) = 1 \wedge ref\_pfd(n-1) = 0] \wedge [vco\_pfd(n) = 0 \vee vco\_pfd(n-1) = 1], 1, 0)$$

The analog filter output can be described as

$$Filter\_out(n+1) = limiter(n) - [limiter(n) - filter\_out(n)] * e^{\frac{-T}{RC}}$$

The property to be verified is described as follows; *if the Freq_sel changes, the VCO output signal should be changed to a new frequency within the LOCK_Time*. The VCO output stability can be decided by the filter output. If the $filter\_output$ signal becomes stable (fixed to a New DC level), then the VCO output will be also stable at the new frequency. The property is expressed in PSL as:
$Freq\_sel \neq X(Freq\_sel) \ |-> (Filter\_out \neq New\_DC\_level)$ $[-> 1 : LOCK\_TIME]$

The implementation of the SRE observer is as follows:

for $n = 1$: $N - LOCK\_Time$ **do**
  if $Freq\_sel(n+1) \neq Freq\_sel(n)$ **then**
    for $i = 1 : LOCK\_Time$ **do**
      if $Filter\_out(n+i) == New\_DC\_Level$ **then**
        return $Violation = False$
      end if
    end for
    return $Violation = True$
  end if
end for

We modeled the frequency synthesizer with SRE, and performed the offline runtime verification for the lock time property using Matlab [4]. The simulation result is shown in Figure 4. The initial value of $freq\_sel$ is 0 and the VCO output signal frequency is 100 Hz, equal to the reference input. The $filter\_out$ is 0. At time $T_0 = 5$ seconds ($t = 0.5 * 10^5 * T$), the $freq\_sel$ changes to 1, with the dividend changing to value 2. The goal is to verify whether the VCO output frequency can change from 100 Hz to 200 Hz within $LOCK\_TIME$. If it is true, the property is satisfied. Otherwise, a violation will be detected at time $T_1 = T_0 + LOCK\_Time$. If our requirement for the *LOCK_Time* is 9 seconds while the system actual *LOCK_Time* is around 12 seconds, then the property will be violated at time 14 seconds ($t = 1.4 * 10^5 * T$), as shown in Figure 4.

VII. Conclusion

In this paper, we have presented an Assertion Based Verification for a class of AMS designs using the Property Specification Language (PSL). Our approach is based on modeling the AMS design in terms of a System of Recurrence Equations (SRE) which support the representation of mixed behaviors. The SRE model provides a mathematical means to computes the symbolic trace for the combined system (design + assertions). We have shown how to use this trace to build checkers for PSL assertions. The verification is achieved via simulation using these checkers in an offline fashion, hence avoiding state space explosion problem. We have implemented

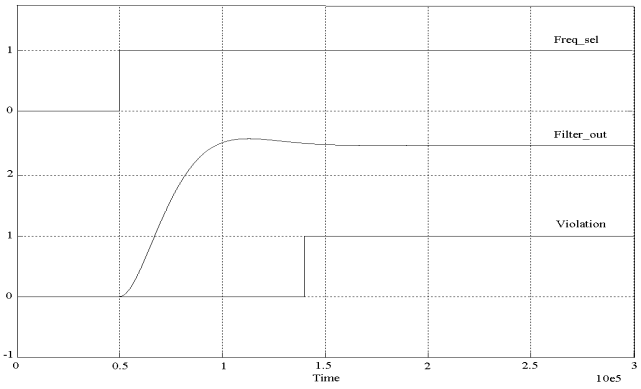

Fig. 4. PLL Frequency Synthesizer Run-Time Simulation

the approach in Mathematica and Matlab and used it to verify the stability property of a ΔΣ modulator and PLL locking time of a frequency synthesizer design.

Our approach currently is limited to discrete analog systems. Future work includes further research to support mixed behaviors for continuous time. In fact, an important effort is still needed to categorize mixed systems and to define what kind of AMS properties we can define using PSL.

References

[1] Accellera Property Specification Language Reference Manual (2004). Available: http://www.accellera.org

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

[3] K. Kundert, H. Chang, D. Jefferies, G. Lamant, E. Malavasi, F. Sendig, Design of mixed-signal systems-on-a-chip. IEEE Trans. on CAD, 19(12): 1561-1571, Dec. 2000.

[4] MATLAB Users Guide, The MathWorks Inc. Available: http://www.mathworks.com/

[5] G. Al-Sammane. Simulation Symbolique des Circuits Decrits au Niveau Algorithmique. PhD thesis, Université Joseph Fourier, Grenoble, France, July 2005.

[6] O. Maler, D. Nickovic. Monitoring Temporal Properties of Continuous Signals. In Formal Modelling and Analysis of Timed Systems, LNCS 3253, Springer, 2004, pp.152-166

[7] O. Maler, D. Nickovic, Amir Pnueli. Real Time Temporal Logic: Past, Present, Future. In Formal Modelling and Analysis of Timed Systems, LNCS 3829, Springer, 2005, pp.2-16

[8] Goran Frehse, Bruce H. Krogh, Rob A. Rutenbar, Oded Maler: Time Domain Verification of Oscillator Circuit Properties. Electronic Notes on Theoretical Computer Science, Elsevier, 153(3): 9-22, 2006.

[9] T. R. Dastidar, P. P. Chakrabarti: A Verification System for Transient Response of Analog Circuits Using Model Checking. IEEE Proc. of the International Conference on VLSI Design, pp.195-200, 2005.

[10] M. Zaki, S. Tahar, and G. Bois: A Practical Approach for Monitoring Analog Circuits; Proc. Great Lakes Symposium on VLSI, IEEE/ACM, pp. 330-335, 2006.

[11] G. Al Sammane, M. Zaki, and S. Tahar: A Symbolic Methodology for the Verification of Analog and Mixed Signal Designs; Proc. IEEE/ACM Design Automation and Test in Europe, pp. 249-254, 2007.

[12] S. Brigati et. al.. Modeling of fractional-N division frequency synthesizer with Simulink and Matlab. IEEE International Conf. on Electronics, Circuits and Systems, pp.1081-1084, 2001.

[13] E. Christen and K. Bakalar. VHDL-AMS: A Hardware Description Language for Analog and Mixed-signal Applications. In IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing., 46: 1263-1272, 1999.

[14] S.Wolfram. Mathematica: A System for Doing Mathematics by Computer. Addison Wesley Longman Publishing, USA, 1991.