Verification of Analog and Mixed Signal Designs using Online Monitoring

Zhiwei Wang[1], Naeem Abbasi[1], Rajeev Narayanan[1],
Mohamed H. Zaki[2], Ghiath Al Sammane[1], and Sofiène Tahar[1]
[1]Dept. ECE, Concordia University, Montreal, Quebec, Canada
Email: {zhiw_wan, n_ab, r_naraya, sammane, tahar}@ece.concordia.ca
[2]Dept. CS, University of British Columbia, Vancouver, British Columbia, Canada
Email: mzaki@cs.ubc.ca

***Abstract*—Analog and mixed signal (AMS) circuits play an important role in today's System on Chip design. They pose, however, many challenges in the verification of the overall system due to their complex behavior. Among many developed verification techniques, runtime verification has been shown to be effective by experimenting finite executions instead of going through the whole state space. In this paper, we present a methodology for the specification and verification of AMS designs using online monitoring at runtime based on the notion of System of Recurrence Equations (SREs). We implement the proposed methodology in a C language based tool, called C-SRE, and utilize it to verify several properties of a PLL design. We compare our proposed online monitoring techniques with the offline approach. Finally, we apply the proposed methodology to monitor the jitter noise associated with a voltage controlled oscillator.**

I. Introduction

With the prominent evolution of semiconductor industry, more and more System on Chip (SoC) architectures are utilized in embedded systems. The design of Analog and Mixed Signal (AMS) components, which connect the physical world to digital space, plays an important role in SoC design.

Traditionally, the verification of AMS designs is carried out using simulation to validate basic specifications. However, with this approach it is not possible to experiment all the combinations of inputs. For this purpose, formal methods, such as model checking, have recently been employed for the verification of AMS designs [17]. Model checking is capable of validating the system by proving that the behavior of the circuit satisfies a set of properties. Unfortunately, model checking of AMS circuits is computationally expensive in terms of memory and CPU time and suffers from the state space explosion problem as well as long processing time which make exhaustive verification very hard [17].

In order to overcome the above mentioned limitations, runtime verification, which integrates program execution and formal methods, has been advocated. Runtime verification is a technique for checking whether an execution of the design model violates the design specifications (properties). Instead of exhaustively walking through the state space, runtime verification considers finite executions. Consequently, the computational resources and simulation time are saved dramatically. There are two types of monitoring techniques [11]: (1) *Online monitoring*, where the properties are observed while the simulation is running, and hence the violation or satisfaction of a property can be detected as soon as it occurs; (2) *Offline monitoring*, where the properties are checked based on the results stored at the end of the simulation. Online monitoring costs less than offline by saving the space required for storage of simulation results. Moreover simulation time is also saved instead of running the whole simulation cycle.

In this paper, we propose an online monitoring approach for the verification of AMS designs based on the notion of Sequence of Recurrence Equations (SRE) [16]. SREs are a form of difference equation used to model both the AMS design under verification as well as the properties to be monitored. We have implemented an SRE based simulator to monitor specifications in PSL (Property Specification Language) standard [2], which we translate into SREs. To illustrate the feasibility of the proposed approach, we have experimented with checking of several properties on a PLL (Phase-Locked Loop) design as case study.

Related Work. The verification of AMS designs using monitoring has attracted a lot of attentions in recent years. The most prominent work is presented in [10], where the authors presented an *offline* methodology for monitoring the simulation of continuous signals. The monitoring technique was based on an extension of PSL to support analog signals. The PSL extension was synthesized into timed automata to check for property violation of simulation traces using Matlab. To demonstrate their approach, a tool AMT [14] was built to support monitoring in both *offline* and incremental fashion. This incremental procedure checks the simulation traces only upon changes in the input signals, rather than in an online fashion. This approach synthesizes the property in terms of lower abstraction such as timed automata and FSM. In this paper, we propose a higher level of abstraction and a unified framework for both modeling and monitoring of AMS designs.

In [6], an online monitoring technique is proposed, where the authors take advantage of linear hybrid automata as monitor to analyze the reachability of time domain features; signal amplitude and jitter of an oscillator circuit were studied.

978-1-4244-4617-9/09/$25.00 ©2009 IEEE

The work in [6] assumes the existence of templates to build the monitor. It does not provide a generic way to obtain the monitors from the specification. In contrast to [6], the approach we advance is capable of modeling and monitoring the AMS design and supports PSL as specification language. In a related work, an FPGA implementation of assertion based monitor is presented in [13]. The authors utilize PSL to generate an asynchronous monitor which is robust to process, temperature and voltage variations. Moreover, the online monitoring is suitable for ASIC designs. Nevertheless, the work in [13] was unable to support both analog and mixed signal circuits nor SERE expressions in PSL.

A more recent work in [7] introduces a methodology to define mixed signal assertions (MSA) for verification by combining PSL and STL (Signal Temporal Logic) [14], where the specifications for digital and analog parts are translated into PSL and STL, respectively, as either precondition or postcondition. An MSA then is constructed by combining the precondition and postcondition with an implication. Assertion based verification or formal verification could be carried out due to the formalized properties. The authors applied the MSA to a first order Σ/Δ converter and several properties were checked. The work was validated within the MLDesigner [12] tool with an enhanced assertion monitoring library. Instead, in this paper, one single formalism, namely SRE, is employed to express PSL properties both for analog and digital part. Additionally, we offer a whole package including simulator and monitor. In [16], an *offline* assertion based verification is introduced, where SREs are used to model the AMS design. Our work is different from [16] in two aspects. First, we use *online* monitoring to achieve verification. Secondly, we present a tool, named C-SRE, which simulates AMS designs modeled with SREs, reads PSL properties and realizes the online monitoring.

The rest of the paper is organized as follows: in Section II, we briefly review the fundamental notions of SRE and PSL and describe how PSL properties can be expressed using SRE. In Section III, we introduce our online monitoring methodology for AMS design verification along with the SRE based simulator. The Section IV is dedicated to the modeling of PLL using SRE. The experimental results on verification of a PLL design are illustrated in Section V. Finally, Section VI concludes the paper.

II. Preliminaries

A. System of Recurrence Equations (SRE)

A recurrence equation or a difference equation is the discrete version of an analog differential equation. A recurrence equation defines a relation between consecutive elements of a sequence. In [16], the notion of recurrence equation is extended to describe digital circuits as follows: Consider a set of variables X_i

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

where $f_i(X_j(n-\gamma)$ is a generalized `If-formula` which is either a variable, an arithmetic operation, a logical formula, a comparison formula or an If-Then-Else expression [16]. The set ϵ_i is a finite non empty subset of $1,\ldots,k \times \mathbb{N}$. The integer γ is called the delay.

A given AMS design consists of blocks both in analog and digital. For digital blocks, the SRE can be expressed directly from their logic functions. For continuous time components, we have two possible ways to generate recurrence equations. First, we can write the recurrence equation based on time domain differential algebraic equations through discretization. The second method is using Impulse-Invariant z transformation to find a discrete time approximation of the continuous time transfer function. Then, we apply the inversion of z transform to generate difference equation and convert it into recurrence equation [1].

B. PSL in SREs

As an assertion language, PSL contains four layers [4]: Boolean, temporal, verification and modeling layer. The verification layer provides the communication and interaction between the properties and the verification tool. The modeling layer is used to defined the verification environment for the tool. The Boolean layer constructs the basic expressions for the property. The temporal layer, where the temporal relations between the signals are expressed, is the heart of PSL. PSL exhibits the property with either linear logic or branching logic. The Foundation Language (FL) is used to express the sequences of the states of the property and the Optional Branching Extension (OBE) is used to interpret the tree of states [2]. However, these languages deal with digital design verification. Due to the presence of analog signals in AMS designs, it is difficult to monitor analog properties in PSL. In order to overcome this obstacle, we propose to rewrite the PSL property into SRE notation. The consistency between the design and the properties allows us to observe the design specifications while running the simulation. The Boolean layer specifies propositions, from the design and signals, which evaluate to the Boolean value true or false in a simulation cycle. The signals of AMS systems may feature continuous time natures. In PSL the analog description to a Boolean variable is an inequation which is built using signals and registers of the AMS design [16]. This expression is defined as the *Basic Property* [16]:

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 a signal (or a register) in the design or an arithmetic function built using the design signals.

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 $n \in \mathbb{Z}$. 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* of *p*. The nature of *n* depends on the SERE or temporal layer operator preceding the basic property. It can be either a numerical constant value, or a symbolic constant value, or a time variable [16].

III. Methodology

In this section, we first present the online monitoring methodology for AMS design verification, followed by details of the SRE based simulator we use.

A. Online Monitoring Methodology

The online monitoring based runtime verification methodology we propose is shown in Figure 1. The AMS design is modeled using SRE based on the circuit description and implemented into our simulator (C-SRE). Design properties are expressed in PSL. The PSL expression is converted to SRE notation. Finally, the input stimulus and output traces are delivered to the monitor. The monitor evaluates the inputs and outputs of the simulator and checks whether the behavior satisfies the design specification. The monitoring is performed in an *online* fashion which means if the property is satisfied, the monitor reports the satisfaction; otherwise, the monitor terminates the simulation at the cycle when the violation occurs. The input stimulus includes the input signal of the system and environment such as control signals. The output trace from the simulator can be either the system output or the output from any component within the system. The AMS specifications we focus on in this paper are written in temporal logic. Hence, we use PSL to express the property initially and then convert it to SRE notation. The evaluation on the relation between input and output with the design specification is carried out within the monitor.

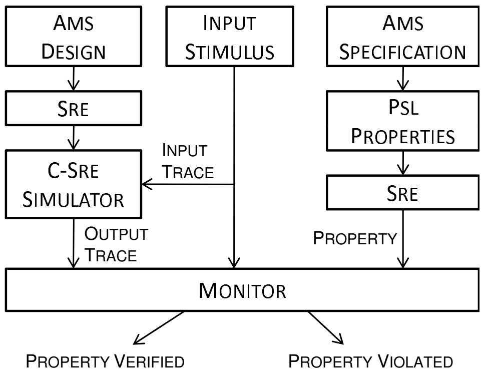

Fig. 1. Proposed Verification Methodology

The monitor is used to check whether the current simulation behavior satisfies a given correctness property. Based on the definitions of *Basic Property* and *Trace of Basic Property* [16], the properties in PSL can be observed by translating them into recurrence sequence notations. In our methodology, the input and output traces are provided to the monitor at each simulation time instant. Incorporated with the property checker described above, at each time instant of the simulation, the property satisfaction is checked. The process is carried out as long as no violation is detected within enough simulation length. Moreover, by taking advantage of the C-SRE simulator (described in Section III-B) which records all the transient data of all circuit blocks at runtime, the monitor is capable of observing the property of individual block.

B. C-SRE Simulator

The proposed modeling technique and *online* monitoring are implemented in a tool named C-SRE[1]. Figure 2 shows the C-SRE simulator framework. The C-SRE tool basically solves a system of recurrence equations describing the behavior of an analog and mixed signal system. There are four main inputs to the tool: (1) The AMS design behavior described using continuous-time (CT), discrete-time (DT) and discrete-events (DE) SRE notations; (2) PSL properties, in SRE notations, expressed using C language; (3) Various inputs and initial conditions to the design; and (4) Simulation parameters such as minimum and maximum time step sizes, and simulation duration etc. The tool output contains the result of executing the monitor in an online fashion, along with various supporting signal traces for easy visualization of the results.

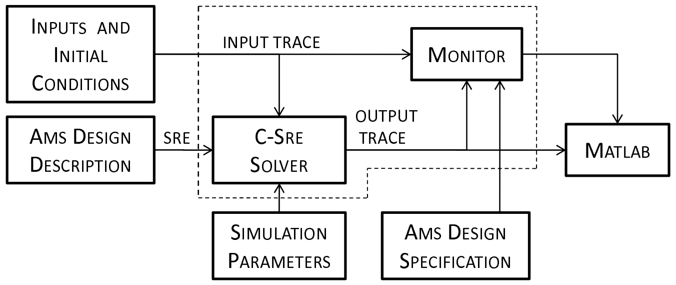

Fig. 2. C-SRE Simulator Framework

The C-SRE solver is the core of the simulator. It guarantees that the CT, DT or DE SREs are executed at an appropriate instant of time to simulate the correct transient behavior of the circuit. The scheduling algorithm is explained below: Let T_{CT}, T_{DT} and T_{DE} be the continuous time, discrete time, and discrete event time steps, respectively. If we assume that T_{CT} is always the smallest time step taken during the simulation, we can achieve both a desired time resolution and accuracy. T_{DT} is uniformly spaced in time and is known in advance. The remaining two time step sizes are determined dynamically during the simulation. In an AMS design, those three processes may interact with each other. The discrete-time part of the design only interacts at intervals of T_{DT} with the other parts. The simulation time advances by following four rules given below:

- If $T_{CT} = T_{DT}$ and $T_{CT} = T_{DE}$ then update the DE and DT SREs
- If $T_{CT} = T_{DT}$ and $T_{CT} < T_{DE}$ then update the DT SREs
- If $T_{CT} < T_{DT}$ and $T_{CT} = T_{DE}$ then update the DE SREs
- If $T_{CT} < T_{DT}$ and $T_{CT} < T_{DE}$ then update the CT SREs

where $t_{CT} = t_{CT} + T_{CT}, t_{DT} = t_{DT} + T_{DT}, t_{DE} = t_{DE} + T_{DE}$ and $T_{current} = MIN(t_{DT}, t_{CT}, t_{DE})$. Figure 3 illustrates example time instants at which continuous-time (circle),

[1]The simulator is named C-SRE because all the components of the simulator are coded in C/C++.

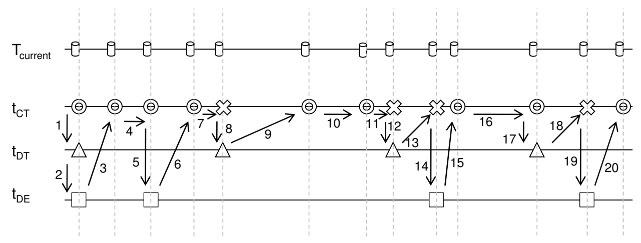

Fig. 3. Timing Diagram

discrete-time (triangle), and discrete-event (square) SREs have to be executed so as to simulate the correct behavior of the system. The numbers in the figure show the sequence of operations. The discrete time steps (triangle) are equally spaced where as the continuous time (circle) and discrete event (square) time steps are determined dynamically during the simulation. The simulation starts with initialization and then proceeds guided by the scheduling algorithm. It terminates when the current simulation time ($T_{current}$) either exceeds or becomes equal to the maximum simulation time. The algorithm described above guarantees that SREs execute in proper sequence in order to simulate the correct behavior of the circuit. For detailed description of the C-SRE simulator, please refer to [1].

In the following sections, we present, respectively, the SRE modeling and runtime verification of a PLL frequency synthesizer as a case study.

IV. PLL Modeling using SRE

A Phase Locked Loop (PLL) is one of the basic AMS building blocks in modern communication systems. A PLL based frequency synthesizer model, as shown in Figure 4, is composed of a comparator (COMP), a phase and frequency detector (PFD), a charge pump (CP), an analog filter (AF), a voltage controlled oscillator (VCO) and a divider (DIV) [3]. If the frequency control signal *Freq_sel* is set to 0, the frequency of the reference input and VCO output will be the same. If *Freq_sel* is set to 1, the VCO output frequency is two times as the reference signal.

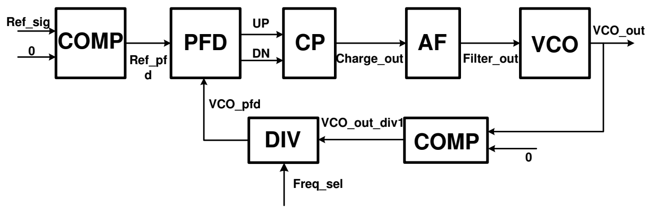

Fig. 4. PLL Frequency Synthesizer

In the following, we briefly describe the SRE models of the important blocks. Please refer to [1] for a full description of PLL in SRE.

Comparator

The comparator extracts the positive value of the input and generates binary sequence with the same frequency as the input. The comparator is a mixed signal component as the input is analog and the output is digital. The SRE model of the comparator is given by

```
Ref_pfd(n+1) = IF(Ref_sig(n) ≥ 0, 1, 0)
Ref_sig(n) = sin(ω₀nT)
```

where Ref_sig is one of the inputs to the comparator.

Phase and Frequency Detector

The phase and frequency detector (PFD) shown in Figure 5 is a pure digital block of the PLL system. Two input signals

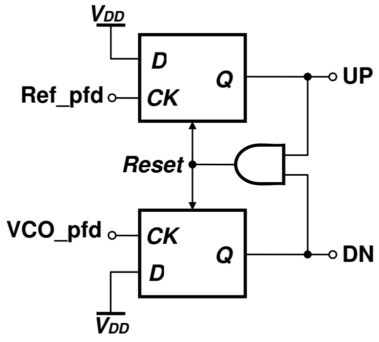

Fig. 5. Phase and Frequency Detector

Ref_pfd and VCO_pfd, which are the reference signal and the feedback signal from VCO output and the divider, respectively, act as the clocks of two flipflops. Each input triggers at its rising edge and propagates the supply voltage from data port D to output port Q. The outputs of interest, UP and DN, reflect the difference in both frequency and phase between the two input signals. When UP and DN are simultaneously high, the AND gate resets both flipflops. The rising edge trigger behavior can be express in SRE as:

```
UP(n+1) = IF{[UP(n)=1 ∧ DN(n)=1],
0,IF[Ref_pfd(n)=1∧Ref_pfd(n-1)=0,1,UP(n)]}

DN(n+1) = IF{[UP(n)=1 ∧ DN(n)=1],
0,IF[VCO_pfd(n)=1∧VCO_pfd(n-1)=0,1,DN(n)]}
```

Two `If-formula` SRE expressions are nested to model the PFD. The outer SRE identifies the reset condition. The inner one generates the difference of frequency and phase according to the rising edge of the input signal.

Charge Pump

The charge pump is usually interposed between the PFD and the analog filter to provide voltage or current for the capacitance in the successive filter. In this case study, we adopt the voltage V_C as source supply. The resulting source supply is proportional to the difference of the output signal UP and DN from PFD. The SRE model of the charge pump is given by

```
Charge_out(n) = V_C × [UP(n-1) - DN(n-1)]
```

The SRE of the charge pump is the difference equation of the functionality.

Analog Filter
An analog filter is usually constructed by combining resistances and capacitances. In our frequency synthesizer, a simple first order lowpass filter is employed. The implementation of the analog filter is shown in Figure 6.

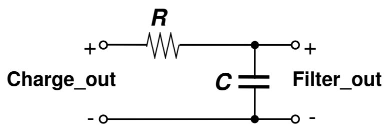

Fig. 6. First Order Lowpass Filter

When modeling an analog filter, we are given the transfer function in frequency domain as:

$$H(s) = \frac{1}{1 + \dfrac{s}{\omega_c}} \tag{1}$$

where $\omega_c = \frac{1}{RC}$. ω_c is the cutoff frequency of the lowpass filter. After applying Impulse-Invariant z transform [3], we obtain the z domain transfer function:

$$H(z) = \frac{\omega_c T}{1 - z^{-1} exp(-\omega_c T)} \tag{2}$$

where T is the sampling time. Taking the inverse z transformation of Equation 2, we achieve the time domain difference equation of the lowpass filter as

$$Filter\_out(n) = \frac{T}{RC} \times Charge\_out(n) + Filter\_out(n-1) \times e^{-\frac{T}{RC}} \tag{3}$$

The corresponding SRE is expressed as:

Filter_out(n) = IF$\{$true,
$(\frac{T}{RC})\times$Charge_out(n)+Filter_out(n-1)$\times e^{(-\frac{T}{RC})}$,0$\}$

The SRE is the difference equation of the lowpass filter.

Voltage Controlled Oscillator
The voltage controlled oscillator (VCO) is the key component of the PLL system. It is used to generate a periodic signal, which has the frequency proportional to the control input voltage. The mathematical model of VCO can be expressed as [3]:

$$VCO_{out}(t) = A_c \cos(\omega_r t + K_{VCO} \int_0^t u(\tau)d\tau + \phi_0) \tag{4}$$

where ω_r is the VCO operating frequency, K_{VCO} is the VCO gain factor, $u(t)$ is the control voltage coming from the lowpass filter, and ϕ_0 is the initial phase. In order to convert voltage input to output frequency, the integral takes place within the total phase. The SRE model of VCO is given by

VCO_gain(n)=$T\times$Filter_out(n)+VCO_gain(n-1),
VCO_out(n)=cos$\{\omega_0$nT+$K_{VCO}\times$VCO_gain(n)+ϕ(n)$\}$,
VCO(n)=IF$\{$VCO_out(n)$\geq$0, 1, 0)$\}$

where *Filter_out* is the output of the analog filter, *VCO_out* is sinusoidal and *VCO_out_div1* is square in shape. The first SRE achieves the integral part of Equation 4. The second one generates the continuous output of the VCO. The last one acts as a comparator to generate the square signal with the same frequency as VCO_out.

Divider
The divider block works as frequency divider. In order to achieve half frequency, the rising edge of the output occurs every two periods of the input signal. In other words, the output signal alternates at each rising edge of the input signal. When the frequency select signal $freq\_sel$ is activated, the output of the divider holds the frequency as half as the input signal. The entire functionality of the divider is modeled using SRE as:

VCO_div(n+1)=IF$\{$(VCO(n)=1)$\wedge$
(VCO(n-1)=0), $\neg$VCO_div(n), VCO_div(n)$\}$

VCO_pfd(n+1)=IF$\{$freq_sel(n)=1,
VCO_div(n), VCO(n)$\}$

The first SRE performs the frequency divider function by identifying the rising edge of the input signal. Once the rising edge occurs, the output alternates itself. Otherwise, it retains its previous value. The second SRE selects the VCO_pfd signal for the comparator.

V. PLL Runtime Verification

We simulated the PLL design shown in Figure 4 using the C-SRE simulator framework described in Section III-B. The simulation parameters are given in Table I. Several properties were verified using the proposed methodology. In following, we describe three example properties.

TABLE I
PLL Parameters

Parameters	Value	Description
T(sec)	10^{-8}	Sampling time
RC(sec)	0.0001	Filter RC parameter
α	exp(-T/RC)	Charge time parameter
Vc(V)	5	Voltage supply
ω_0(rad·Hz)	$2\pi \times 10^6$	Input signal frequency
K_0(rad·Hz·V^{-1})	$2\omega_0/Vc$	VCO gain
New_DC_Level(V)	2.5	Filter output

Property 1. Consider the PLL in Figure 4, the lock time is one of the most important properties. It determines how fast the frequency synthesizer gets stable from one frequency to another. No useful data can be transmitted during this time. A large lock time can reduce the data rate of the system. This is the key factor when designing PLL circuits. The lock time

property can be described as: after the *Freq_sel* signal changes from 0 to 1, the output of the lowpass filter will reach the new DC value within the lock time. The PSL style definition of this safety property is given below:

```
Property_1 = {Freq_sel==0;Freq_sel==1} |->
 (Filter_out≠New_DC_Level)[*1:LOCK_TIME]
```

The SERE concatenation operator (;) indicates that the two Boolean expressions it connects hold consecutively [2]. Another SERE operator [*] constructs the consecutive concatenation of the Boolean expression which should hold according to the range it specifies [2]. The property monitor implemented in C language is shown below

```
while (freq_sel[i]==1 && freq_sel[i-1]==1)
    {
    for(int n=i; n!=N_max; ++n){
     if (filter_out[n]==new_dc_level &&
            T_sample*[n-i]<=Lock_time){
            property_lock_time=1; // Satisfied
        } else{
            property_lock_time=0; // Violated
        }}
    i++;}
```

According to the parameters listed in Table I, the lock time of the system is 1.5ms. It can be seen from Figure 7 that as soon as the time reaches 1.5ms after *Freq_sel* signal changes from 0 to 1, the monitor reports a violation for the property. The simulation then is suspended at 1.5ms. There is no need to look at the simulation trace after 1.5ms in this case. Due to the prompt violation alert, the simulation time is expected to be saved. In addition, two more interesting properties are described below (please refer to [1] for more properties).

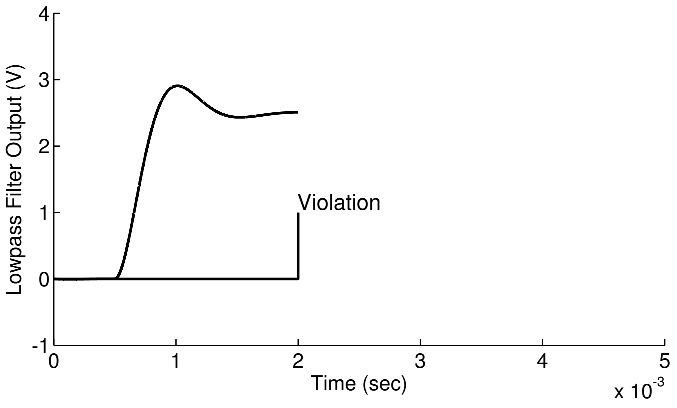

Fig. 7. Locktime Verification Results (Property 1)

Property 2. If the *Freq_sel* changes, the VCO output signal should change to a new frequency eventually. The VCO output stability can be decided by the filter output. This is a liveness property, expressed in PSL as:

```
Property_2 = {Freq_sel==0;Freq_sel==1} |->
  eventually! {Filter_out == New_DC_Level}
```

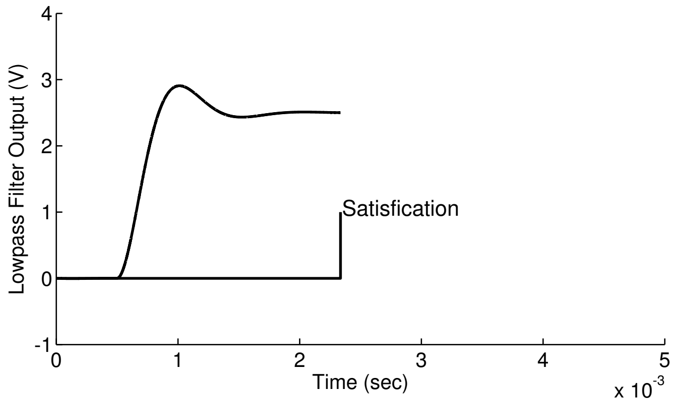

Fig. 8. Verification Results for Property 2

The "*eventually*!" is an linear time temporal logic (LTL) [5] style operator in PSL. It specifies that a property holds at the current cycle or at some future cycle. The verification result is shown in Figure 8. The monitor is sensitive to satisfaction rather than violation. The reason for that is because this is a liveness property which implies that something good eventually happens. The detection of satisfaction is more feasible than violation. The simulation is terminated when the property is verified at 2.4ms.

Property 3. After reset, the *Freq_sel* will be '0', and *Filter_out* will also be '0'. If the *Freq_sel* changes to '1', the *Filter_out* will increase until *New_DC_Level*. Hence, the *Freq_sel* will be '0' until *Filter_out* is larger than '0'. This is a safety property, expressed in PSL as:

```
Property_3=(Freq_sel==0) until!(Filter_out>0)
```

The "*until*!" is like the LTL operator *until* in strong form which requires that the termination condition eventually happens. In context, the property 3 requires signal *Filter_out* is expected to change.

We compared our methodology to the work presented in [16] and the experimental results are listed in Table II. Thereafter, we notice that our proposed online monitoring technique performs better than [16] in terms of simulation time. The memory usage of our methodology is slightly higher than that of [16] due to the computational efficiency in Matlab. Both approaches were run on the same ULTRA SPARC-IIIi server (177 MHz CPU, 1GB memory), where all properties described above are satisfied. To further demonstrate the ability of our proposed online monitoring technique, we present the modeling and verification of the jitter property within the VCO of the PLL.

Jitter is an undesired time variation of signals in most communication systems. In a PLL, jitter can be found in the phase and frequency detector (PFD), the frequency divider or the voltage controlled oscillator (VCO). The jitter in the PFD is caused by the uncertain changes in the delay between the input and output signal. The VCO output depends not only on the transitions at their inputs, but the previous output [9].

TABLE II
SIMULATION RESULTS

Property	Online Monitoring		Offline Monitoring		Verification Status
	Simulation Time(sec)	Memory Use(MB)	Simulation Time(sec)	Memory Use(MB)	
Property 1	13.4	38	38.87	32.4	Violated
Property 2	13.4	38	37.61	32.4	Satisfied
Property 3	13.3	38	43.65	32.4	Satisfied

Therefore the lock time is most affected by the jitter in the VCO. The jitter is defined as variation in the period and is modeled as a variation in the frequency of the VCO. Assume that the frequency of a periodic signal without jitter is given by

$$f = \frac{1}{T} \tag{5}$$

where T is the period of the ideal signal. The jittery frequency can be represented as

$$f_{jitter} = \frac{1}{T + \Delta T} = \frac{1}{\dfrac{1}{f} + \Delta T} = \frac{f}{1 + \Delta T \cdot f} \tag{6}$$

where $\Delta T = J\delta$. J is the jitter deviation and δ is a zero mean unit-variance Gaussian random process. Let $\phi(t)$ be the phase of the integral term in Equation 4. We have

$$\phi(t) = K_{VCO} \int_0^t u(\tau) d\tau \tag{7}$$

Suppose $\omega(t) = 2\pi f(t) = K_{VCO} \cdot u(t)$, we obtain

$$f(t) = \frac{K_{VCO} \cdot u(t)}{2\pi} \tag{8}$$

which shows the VCO output frequency is linearly proportional to the input control voltage $u(t)$ and frequency $f(t)$ by multiplication of VCO gain factor. By substituting Equation 8 into Equation 6, we achieve

$$f_{jitter}(t) = \frac{f(t)}{1 + \Delta T \cdot f(t)} = \frac{f(t)}{1 + \Delta T \cdot \dfrac{K_{VCO} \cdot u(t)}{2\pi}} \tag{9}$$

The VCO model with jitter is shown in Figure 9. By substituting Equation 9 to Equation 4, we achieve mathematical model of the VCO with jitter:

$$\begin{aligned} VCO_{out}(t) = A_c \cos(\omega_r t + \\ K_{VCO} \int_0^t \frac{u(\tau)}{1 + \dfrac{J\delta \cdot K_{VCO} \cdot u(\tau)}{2\pi}} d\tau + \phi_0) \end{aligned} \tag{10}$$

The jitter in VCO usually causes uncertain shifts which result from the phase change in the phase part of the formula above. It is modeled as a random variation in the frequency of the VCO. The lock ability of a PLL design under normal distributed jitter conditions can be identified by our monitoring technique. Figure 10 shows the output of the lowpass filter when jitter was generated using jitter deviation factor J of 10^{-6}s labeled as (2) and 7×10^{-6}s labeled as (3), respectively. The thick line labeled (1) represents the output of lowpass filter without jitter noise. The thin line (2) shows that the lowpass filter output almost settles to a new DC level but not stable at the level. The line (3) exhibits that the low pass filter output does not settle to the desired DC value and PLL is unable to lock within the lock time specification of the PLL. It is intuitive that larger jitter noise is expected to cause failures to the PLL.

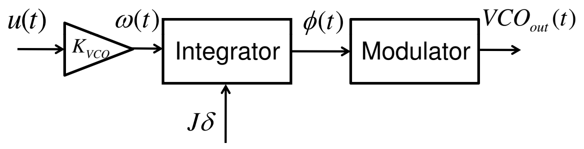

Fig. 9. Modeling of Jitter in VCO

Most real life systems contain some source of randomness. The behavior of PLL under uncertain conditions such as jitter cannot be described by PSL as there is no statistical or probabilistic layer in PSL at this time. We are currently looking into how probabilistic properties can be described and verified using monitoring techniques.

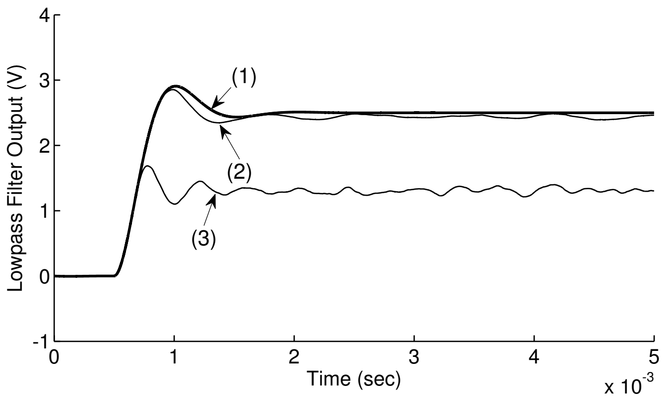

Fig. 10. Lowpass Filter Output Voltage with Jitter Noise in VCO

VI. CONCLUSION

In this paper, we presented a methodology for functional verification of AMS designs using SRE at run-time. We also described an SRE simulator developed for this purpose. Several properties of a PLL frequency synthesizer were described in PSL and then translated into SRE notations and then checked by an online monitor during simulation. The runtime verification offers a direct view of the process of transition within the analog and mixed signal system. The monitoring point can either be located at analog signals or digital signals.

The main advantages of our methodology are: (1) the SRE algorithm is likely to be understood both by analog and digital engineers; (2) since both the AMS design and PSL properties are described with SREs, continuous time behavior can be simulated and monitored together with the discrete time; and (3) online monitoring saves cost in terms of memory usage and simulation time.

There are also some limitations of the new simulator. For example in the current implementation, the PSL expression cannot be automatically translated into C language and input to the simulator, which is the subject of our future work. Furthermore, we would like to upgrade our simulator to support various constructs from newly introduced Analog PSL [11].

REFERENCES

[1] N. Abbasi, R. Narayanan, G. Al Sammane, M.H. Zaki, and S. Tahar. Enabling AMS Simulation using Recurrence Notations, Technical Report, Department of ECE, Concordia University, Montreal, QC, Canada. May 2008. http://hvg.ece.concordia.ca/Publications/TECH_REP/CSRE_TR08

[2] Accellera PSL. http://www.accellera.org/

[3] R. E. Best. Phase-Locked Loops: Design, Simulation, and Applications. McGraw-Hill, 2003.

[4] C. Eisner. PSL for Runtime Verification: Theory and Practice. Runtime Verification, LNCS 4839, pp. 1-8, Springer, 2007.

[5] E.A. Emerson. Temporal and Modal Logic. Handbook of Theoretical Computer Science, pages 995-1072. Elsevier, 1990.

[6] G. Frehse, B.H. Krogh, and R.A. Rutenbar. Time Domain Verification of Oscillator Circuit Properties. Workshop on Formal Verification of Analog Circuits, ENTCS, 153(3): 9-22, 2006.

[7] A. Jesser, S. Laemmermann, A. Pacholik, R. Weiss, J. Ruf, L. Hedrich, W. Fengler, T. Kropf, and W. Rosenstiel. Advanced Assertion Based Design for Mixed-Signal Verification. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, E91(A12):3548-3555, 2008.

[8] K. Kundert, H. Chang, D. Jefferies, G. Lamant, E. Malavasi, and F. Sendig. Design of Mixed-Signal Systems-on-a-Chip. IEEE Trans. on CAD, 19(12): 1561-1571, 2000.

[9] K. Kundert. Predicting the Phase Noise and Jitter of PLL-Based Frequency Synthesizers. http://www.designers-guide.org/Analysis/PLLnoise +jitter.pdf

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

[11] O. Maler, D. Nickovic, and A. Pnueli. Checking Temporal Properties of Discrete, Timed and Continuous Behavior. Pillars of Computer Science, LNCS 4800, pp. 475-505, Springer, 2008.

[12] MLDesign Tech., Inc. http://www.mldesigner.com

[13] K. Morin-Allory, L. Fesquet, B. Roustan, and D. Borrione. Asynchronous Online Monitoring of Logical and Temporal Assertions. Embedded Systems Specification and Design Languages, LNEE 10, pp. 243-253, Springer, 2008.

[14] D. Nickovic, and O. Maler. AMT: A Property-Based Monitoring Tool for Analog System. Formal Modelling and Analysis of Timed Systems, LNCS 4763, pp. 304-319, Springer, 2007

[15] B. Razavi. RF Microelectronics, Prentice Hall, 1997.

[16] G. Al Sammane, M.H. Zaki, Z.J. Dong, and S. Tahar. Towards Assertion Based Verification of Analog and Mixed Signal Designs Using PSL. Forum on Specification & Design Languages, pp. 293-298, 2007.

[17] M. Zaki, S. Tahar and G. Bois. Formal Verification of Analog and Mixed Signal Designs: A Survey. Microelectronics Journal, 39(12):1395-1404, Elsevier, 2008.