Checking Properties of PLL Designs using Run-time Verification

Zhi Jie Dong, Mohamed H. Zaki, Ghiath Al Sammane, Sofiène Tahar and Guy Bois⋆
ECE Dept., Concordia University
Email: {zh_do, mzaki, sammane, tahar}@ece.concordia.ca
⋆Génie Informatique, École Polytechnique de Montreal
Montreal, Québec, Canada
Email: guy.bois@polymtl.ca

Abstract— **Due to challenges associated with its verification process, analog and mixed signal designs like PLLs require a considerable portion of the total design time. In this paper, we propose a run-time verification approach for PLL designs. The essence of this approach is to monitor properties of interest by timed automata integrated within an automatic stimulus generation framework. The objective is to guide simulation by an appropriate simulation trace in order to quickly detect errors by the property monitor.**

I. Introduction

Analog and Mixed Signal (AMS) designs are integrated parts of embedded systems, dedicated for realizing control and signal processing functions. The verification of AMS designs is one of the major tasks in computer aided design (CAD). However, steadily increasing design sizes and functional complexity make the verification a bottleneck in the design flow.

Usually, simulation is used in the verification of large designs. However, such method is not enough to validate interesting properties of the expected behavior, involving for instance, temporal requirements. By contrast, formal verification techniques, like model checking [4], provide full verification coverage. Unfortunately, model checking of AMS designs is computationally expensive and suffers from the problem of state-space explosion that makes complete verification of real designs impossible [4]. To tackle the drawbacks of the above-mentioned verification techniques, the integration of both simulation and formal methods was advocated. Among the proposed approaches is checking whether the simulation results are conforming to the formal specification. This approach is known as *run-time verification* [8].

In run-time verification, the AMS design is simulated with an associated testbench which models the environment. In fact, the testbench provides inputs stimulus necessary to drive the design into a desired behavior. Meanwhile, the output and internal signals are monitored to detect possible errors, by means of logical models that describe the property of interest. Classically, stimulus can be manually developed (directed) or randomly generated. While, the monitor could be as simple as observing the current or voltage at different nodes of the design, or more complex like checking temporal behavior. In [5], a run-time verification approach for VHDL-AMS designs is proposed, in which the properties of interest are monitored by timed automata. In this paper, we integrate this methodology within an automatic stimulus generation framework interfaced with the VHDL-AMS simulator. The objective is to guide simulation by an appropriate simulation trace in order to quickly detect the errors by the timed automata monitor. We explain our method using a PLL design as a verification case study.

Related Work. In [10], the authors classified run-time verification in two different groups: offline and online monitoring. In *offline monitoring*, the property is only verified after the whole simulation trace is provided, while in *online monitoring*, the monitoring is interleaved with the simulation process, hence, providing the ability to detect violation as soon as they happen. Recently, several run-time approaches have been proposed for the verification of AMS designs. An online monitoring approach was proposed in [7], where template monitors described by linear hybrid automata are used to model the time domain features of oscillatory behavior. The monitors are implemented within the model checking tool *PHAver*. In [12], the authors used interval arithmetics to describe the reachable states of the design behavior. However, instead of creating template automata to use as monitor, the authors synthesize timed automata monitor from a subset of the timed computational temporal logic (T-CTL) properties. The work in [7] and [12] are only limited to analog behaviors. In addition, the techniques used are computationally expensive, due to the requirement to build an overapproximated solution flow for the system equations, making them only adequate for small designs.

An offline approach was used in [9], for monitoring Matlab simulation traces of continuous-time analog signals to check for property violation. The authors extended the PSL [1] logic to support analog signals. Synthesized timed automata are then generated from the PSL properties to monitor the traces. The main disadvantage of this method is that property violation can be checked only at the end of the simulation process, which can consume time and memory resources.

The methodology we present in this paper can also be related to the work done in [11], where the authors proposed an online monitoring verification technique of hybrid systems using timed and linear hybrid automata. With respect to this work, we distinguish ourselves in the fact that we build the stimulus generator and the property monitor as timed automata within the VHDL-AMS environment. In addition, we provide a mechanism to guide simulation by an appropriate simulation trace in order to quickly detect the errors by the timed automata monitor.

The rest of the paper is organized as follows: we describe the verification methodology in Section II, followed by the PLL verification in Section III, before concluding with Section IV.

II. Methodology

The proposed verification methodology is illustrated in Figure 1. Given the environment constraints and the property of interest for the design under verification, we construct the stimulus generator and the property monitor as deterministic timed automata. A deterministic *Timed Automaton* (TA) is a finite state machine with a set of timers constraints specified on each state location. The timer constraints are defined over a set of clock variables. On the edges between these locations, constraints guards on the state variables along with the evaluation updates of the variables are specified. Variables updates holding on a certain edge if the constraints are satisfied on that edge. In addition, a set of clock variables constraints and resets are also specified on the edges. The *monitoring automaton* is an extended timed automaton, with the acceptance condition that determines the set of allowable states; therefore rejecting bad states. The stimulus and monitoring automata are implemented as components written in VHDL-AMS [3]. Together with the VHDL-AMS design and the PLL libraries they are input to the ADM simulator [2] to perform run-time verification.

To enhance the verification coverage, we have extended the run-time verification with a communication mechanism between the stimulus generator and the monitor automata. For instance, the communication signals from the stimulus automaton to the monitor automaton are used as trigger signals to start a new monitoring process. While the feedback signals from the monitor to the stimulus are used to guide the stimulus to choose next test vector. In fact, the generator automata must produce an input test vector for the design, such that the property constraints will be satisfied.

III. PLL based frequency synthesizer design

Phase-locked loops (PLLs) are a class of AMS designs widely used for synchronization purposes, demodulation or to synthesize new frequencies. In this paper, we apply our verification methodology to a PLL design based frequency synthesizer.

Figure 2 shows the PLL based frequency synthesizer block diagram which consists of four blocks: comparator, voltage-controlled oscillator (VCO), digital phase and frequency detector (PFD), and analog loop filter with charge pump. The PFD is composed of two D Flip-Flops followed by a charge pump and an analog passive lead-lag loop filter. The input

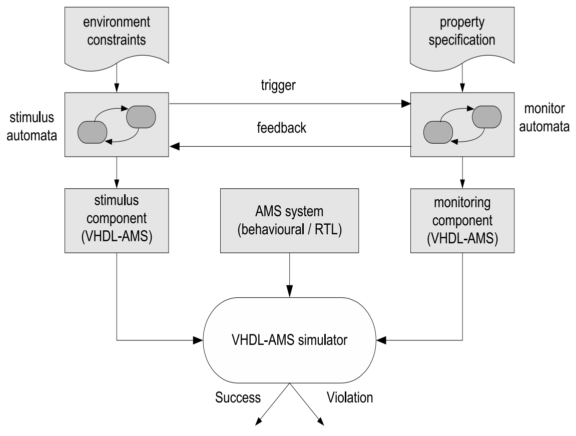


Fig. 1. Proposed Verification Methodology

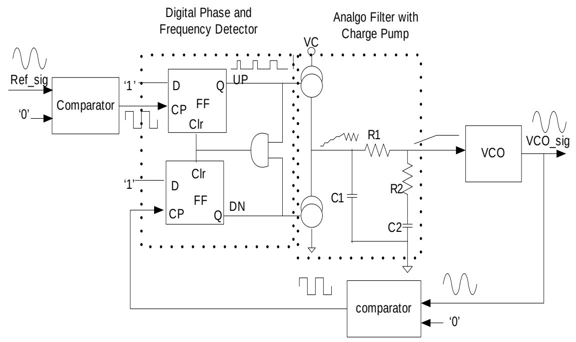
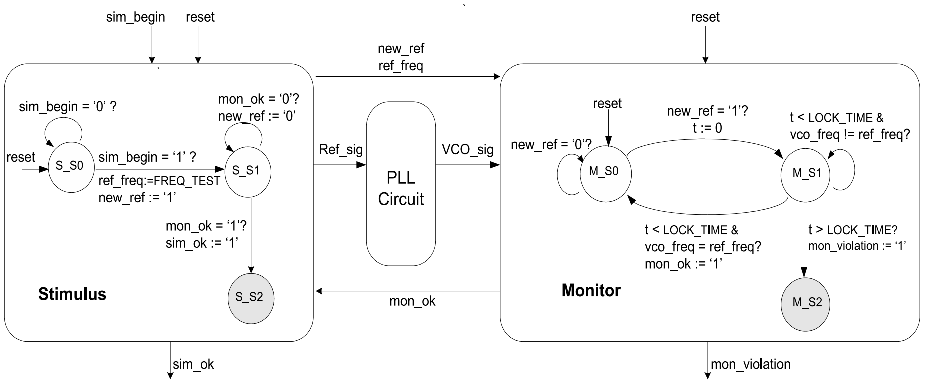

Digital Phase and Frequency Detector
Analgo Filter with Charge Pump
VC
Ref_sig
'0'
Comparator
'1'
D
Q
CP
FF
Clr
UP
DN
R1
R2
C1
C2
VCO
VCO_sig
comparator

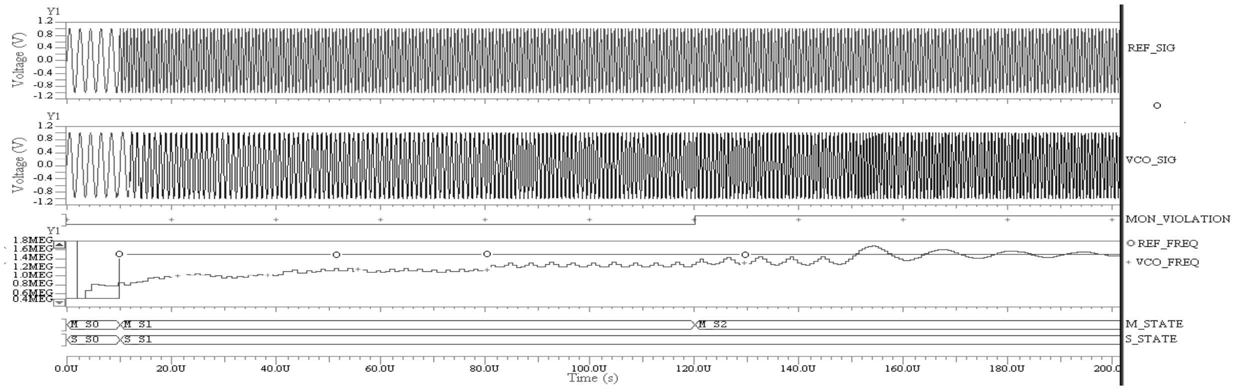
Fig. 2. Frequency Synthesizer PLL

reference signal *Ref_sig* is a sine wave signal with frequency *ref_freq*. The VCO output *VCO_sig* is a cosine wave signal with frequency equal to *ref_freq* if the PLL is locked.

The first property that we have considered is the *lock time property*. It is the time necessary for the PLL to switch from one frequency to another within a given range. 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. The property to be verified is described as follows: *if the input reference signal changes to a new frequency, the VCO output signal should be able to keep up with the new frequency within the lock time*. The corresponding stimulus and monitor automata are shown in Figure 3(a). The stimulus automata generates the stimulus signal *ref_sig* to the PLL. *New_ref*, *ref_freq* and *mon_ok* are the communication signals between stimulus and monitoring timed automata. Setting *new_ref* signal to '1' triggers the monitor to go to the *LOCK_TIME* checking state M_S_1. If within the *LOCK_TIME*, the VCO output frequency *vco_freq* equal to *ref_freq*, the monitor will set the signal *mon_ok* to '1' triggering the stimulus to reach the end state S_S_2. However, if *vco_freq* is not equal to *ref_freq* within *LOCK_TIME*, the monitor will reach the failure state M_S_2 and stay there for ever. Consequently, the signal *mon_violation* will be set to '1' indicating the lock time property is violated.



(a) Timed Automata



(b) Simulation Results

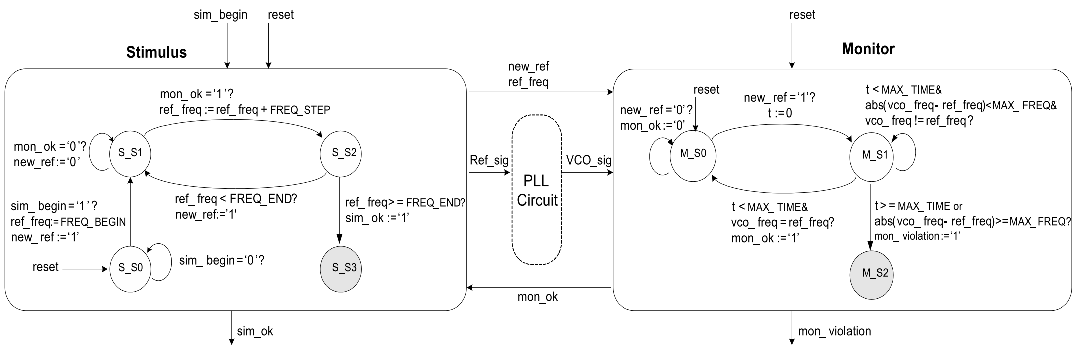
Fig. 3. Lock Time Verification

The lock time simulation result is shown in Figure 3(b). We assume that the requirement of the lock time is 110 μs for a frequency of 1.5 MHz. By inspecting the violation simulation trace *mon_violation*, we identify that a violation is detected at time 120 μs. This is due to the fact that the frequency of VCO output signal is not equal to 1.5 MHz. In such case the monitor automaton reaches to the failure state M_S_2. In fact, from the simulation result we can see that *vco_freq* will be equal to *ref_freq* until 200 μs.

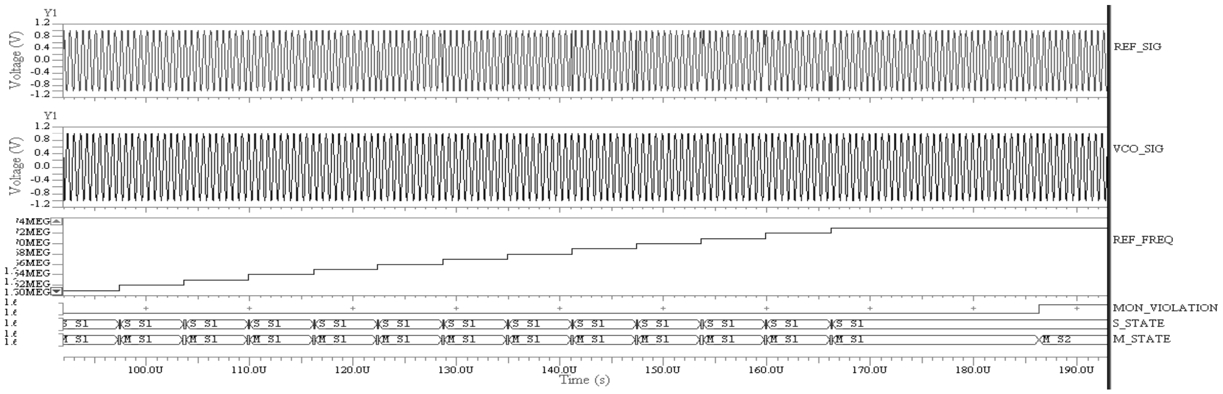
The second property that we have verified is the *lock range property*. The lock range is defined as the range of frequencies over which the loop will remain in lock. It is also called the tracking range or hold-in range. This can be measured by slowly increasing or decreasing the frequency of the input until the loop looses lock. The stimulus generator and the monitor automata along with the communication signals are shown in Figure 4(a). The monitor automaton is nearly the same as the one in the lock time monitor except that the *LOCK_TIME* is replaced by *MAX_TIME* and a frequency out-of-range constraint guard is added. The *MAX_TIME* is user defined and is larger than the *LOCK_TIME*.

The signal *sim_begin* starts the lock time checking in the stimulus automaton, allowing a transition from S_S_0 transition to S_S_1 where the reference signal frequency *ref_freq* is set to *FREQ_BEGIN*. Meanwhile, *new_ref* will be set to '1' triggering the monitor to go to the lock range checking state M_S_1. If within the *MAX_TIME* the VCO output *vco_freq* is equal to *ref_freq*, then the monitor will go back to M_S_0, waiting for another lock range test request. This happens when the stimulus goes to S_S_2 increasing *ref_freq* by *FREQ_STEP*. This process can be repeated until *ref_freq* reaches *LOCK_END*. In such case, the stimulus will go to the state S_S_3. On the other hand, if *vco_freq* is not equal to *ref_freq* within *LOCK_TIME*, the monitor will reach the failure state M_S_2, indicating the lock range property is violated.

The simulation result is shown in Figure 4(b). Assume that the PLL central frequency is 1.5 MHz, the higher end of *LOCK_RANGE* is 1.75 MHz and the *MAX_TIME* is 20 μs. As illustration, we set the frequency increasing step *FREQ_STEP* to 0.1 MHz, the reference signal initial frequency *FREQ_BEGIN* to 1.5 MHz and the end frequency *FREQ_END* to 1.75 MHz. We need to verify that if the reference signal changes from 1.5 MHz to 1.75 MHz with a 0.1 MHz step, the PLL signal *VCO_sig* will lock to the reference input signal. The monitor automata will detect a violation at time 186 μs. This can be explained by inspecting the simulation trace. The PLL can lock the input reference frequency from 1.5 MHz to 1.72 MHz. However, at 166 μs, the reference signal frequency changes to 1.73 MHz and there is a violation detected at time 186 μs, where the frequency of the VCO output is not equal



(a) Timed Automata



(b) Simulation Results

Fig. 4. Lock Range Verification

to 1.73 MHz and *MON_MTA* reaches the failure state M_S_2 [1].

IV. Conclusion

In this paper, we have used the concept of run-time verification to check functional properties of AMS designs. The idea behind the presented approach is based on modeling and simulating the given AMS design along with the required properties described using monitoring automata in the VHDL-AMS environment. The verification is then achieved in an online fashion, hence avoiding the problem of the state space explosion. Furthermore, such approach is more reliable than the manual inspection of simulation traces which is usually complex and costs lots of time and effort. On the other hand, combining simulation and properties based verification helps achieving the benefits of both approaches, while at the same time avoids some of the drawbacks mentioned earlier. We have used the methodology to verify functional properties of a PLL based frequency synthesizer. As a future work, we plan to investigate possible methods for generating time automata from temporal properties, which could lead to more compact representation of the specification.

References

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

[2] ADVance MSTM Reference Manual, Mentor Graphics, http://www.mentor.com

[3] 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, 46: 1263-1272, 1999.

[4] E.M. Clarke, O.Grumberg, D.A. Peled, Model Checking. MIT Press, 2000.

[5] Z.J. Dong, M.H. Zaki, G. Al Sammane, S. Tahar and G. Bois: Run-time Verification Using the VHDL-AMS Simulation Environment, Proc. IEEE Northeast Workshop on Circuits and Systems, pp.1513-1516, 2007.

[6] Z.J. Dong, M.H. Zaki, G. Al Sammane, S. Tahar and G. Bois. A Run-Time Verfication Approach for AMS Designs. Technical Report, ECE Dept., Concordia University, July 2007, http://hvg.ece.concordia.ca/Publications/TECH_REP/AMS_RTV_TR07/

[7] G. Frehse, B. H. Krogh, R. A. Rutenbar, O. Maler. Time Domain Verification of Oscillator Circuit Properties. Electronic Notes in Theoretical Computer Science. 153(3): 9-22, 2006.

[8] K. Kundert and H. Chang. Verification of Complex Analog Integrated Circuits. In Proc. IEEE Custom Integrated Circuits Conference, pp.177-184, 2006.

[9] 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

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

[11] L. Tan, J. Kim, I. Lee: Testing and Monitoring Model-based Generated Program. Electronic Notes in Theoretical Computer Science. 89(2): 128-148, 2003.

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

[1]Due to lack of space, we refer to [6] for more details about the PLL verification, including checking clock jitter