

Abstract Modeling and Simulation Aided Verification of Analog/Mixed-Signal Circuits^{*}

Scott Little and Chris Myers

University of Utah, Salt Lake City, UT 84112, USA
{little,myers}@vlsigroup.ece.utah.edu

Abstract. Analog/Mixed-signal (AMS) circuit verification is a growing problem as process variation increases and AMS circuits become more functionally complex. To improve analog verification flows, AMS circuit models are needed at different levels of abstraction. This paper discusses recent work and future directions for abstract model generation and simulation aided verification of AMS circuits. In particular, a CMOS ring oscillator with feedforward inverters is used as a motivating example for the work. This example highlights progress and future directions in AMS modeling and verification.

1 Introduction

System-level design and verification challenges are growing as *analog/mixed-signal* (AMS) circuit designs are fabricated on increasingly variable processes and become more functionally complex [1]. The increased variation and complexity are leading to an increase in the number of design errors, particularly functional errors, requiring costly respins for AMS circuits. Many of these bugs could be found prior to fabrication using improved AMS verification tools and methodologies. This paper focuses on a methodology to automatically generate abstract models of AMS circuits that can be used in system-level verification.

Abstract modeling is a common practice in system design for digital, analog, and mixed-signal circuits. High-level languages such as C/C++, SystemC, and Simulink are often used by architects to model the system at a high level. These models are useful when making architectural level decisions as well as functioning as a golden reference during verification. Transistor-level models are used to accurately represent the precise details of the circuit. Transistor-level models work well for block-level verification but are too detailed for efficient system-level simulation and verification. There is a need for intermediate level models that are abstracted versions of the transistor-level models. In digital design, these intermediate models are referred to as *register transfer level* (RTL) models and are commonly used in system-level functional verification. These models are created with languages such as Verilog and VHDL. Extensions to Verilog and VHDL for AMS designs in the form of Verilog-AMS [2] and VHDL-AMS [3] have

^{*} This research is supported by SRC contract 2002-TJ-1357 and an SRC Graduate Fellowship.

been standardized and are supported by many simulation environments. There has not, however, been widespread adoption of these languages by AMS design engineers due in large part to the time and expertise required to create high quality AMS HDL models. Maintaining these models also requires time. Most AMS designers work at the transistor-level. This results in designers having to propagate changes at the transistor-level up to the AMS HDL level. Digital designers avoid this maintenance challenge by working at the RTL level and synthesizing transistor-level circuits. Analog synthesis has been attempted with varying levels of success but is not currently in wide use. Because AMS designers continue to work at the transistor-level, we propose generating AMS HDL models from information obtained at the transistor-level. This methodology enables the use of AMS HDL models while alleviating the model creation and maintenance challenges.

Nearly all AMS verification is done at the transistor-level using a SPICE simulator. During the design process, the transistor-level design is simulated heavily to ensure it meets the specification in the nominal case. It is also well characterized over process corners and other potential sources of variation. This simulation data contains a wealth of information about the design. Our work leverages the information in the simulation data via a *simulation aided verification* (SAV) methodology. This methodology is supported by LEMA, an AMS modeling and verification tool. Fig. 1 illustrates the verification flow supported by LEMA. There are two primary input formats accepted by LEMA, VHDL-AMS and simulation data. Our previous work [4–7] describes how a subset of VHDL-AMS can be compiled into a *labeled hybrid Petri net* (LHPN) and analyzed using one of our model checkers. Each model checker uses a different data structure to represent that state space including: *difference bound matrices* (DBMs) [4], *binary decision diagrams* (BDDs) [5], and *satisfiability modulo theories* (SMT) formulas [7]. The other option is to use the simulation traces already produced by the designer as well as safety properties and thresholds on the signal levels of the design variables to automatically generate AMS HDL and LHPN models of the system [6, 8, 9]. The AMS HDL models are intended for use in system-level simulations. These models are not as accurate as the transistor-level models, but they simulate much faster and support standard simulation environments and workflows. The LHPN model generated by the model generator can be used to formally verify safety properties of the system using one of LEMA’s model checkers.

Our SAV methodology for AMS circuits has been successfully applied to a switched capacitor integrator circuit [8] and a PLL phase detector [9]. Recently, we have applied the previously developed methodology to a CMOS ring oscillator with feedforward inverters. Applying our SAV methodology to this example has produced refinements to the previous methodology as well as a number of future directions for research. This paper highlights the progress and potential future work in AMS modeling and verification via the ring oscillator example.

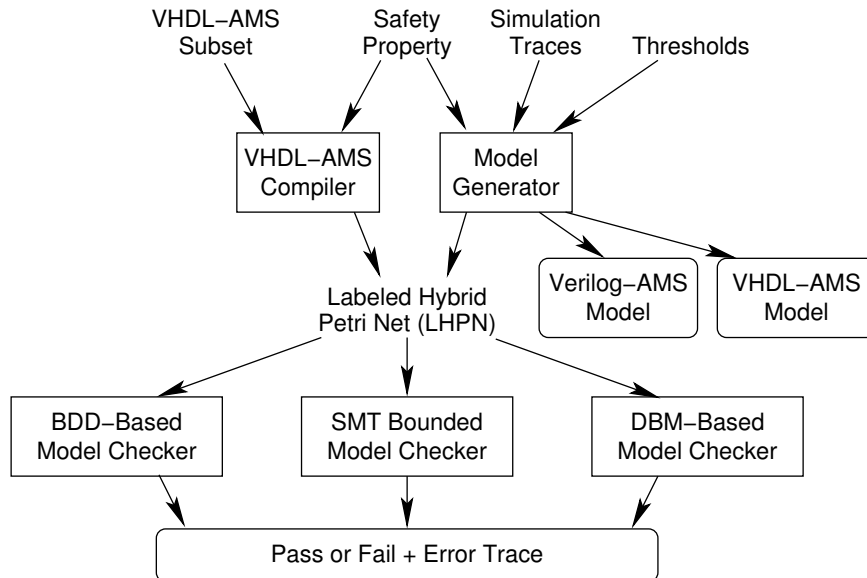


Fig. 1. LEMA workflow diagram.

2 CMOS Ring Oscillator with Feedforward Inverters

Engineers at Rambus recently suggested investigating the modeling and verification of the ring oscillator in Fig. 2 [10]. Traditional CMOS ring oscillators are created using an odd number of inverters in the ring. These oscillators have an oscillation period of $2N\tau_d$, where τ_d is an inverter delay. The period of oscillation can be reduced to $N\tau_d$ by using *feedforward inverters* to send the signal ahead in the ring [11]. An advantage of using feedforward inverters is that the inverter ring can be designed to oscillate using an even number of inverters. This allows for *quadrature outputs* (i.e., four outputs where each output is 90° degrees out of phase with the previous output). This is not possible with a ring oscillator employing an odd number of inverters. Quadrature outputs are becoming increasingly important in clock recovery and multi-phase processor clocking. For these reasons, Rambus uses this circuit in many of their VCO designs. Despite having an essentially digital specification, this circuit requires an analog analysis, since its behavior cannot be reasonably analyzed at the switch-level.

This circuit oscillates for a particular range of ratios in transistor sizes for the chain inverters to feedforward inverters. If the feedforward inverters ($F_1 - F_4$) are much larger than the chain inverters ($C_1 - C_4$), the feedforward inverters overpower the chain inverters and latch the values X and Y at opposite logical values. A simulation using a ratio of chain to feedforward inverters of 0.39 exhibits this behavior as shown in Fig. 3. If the ratio is changed to 0.40, the circuit

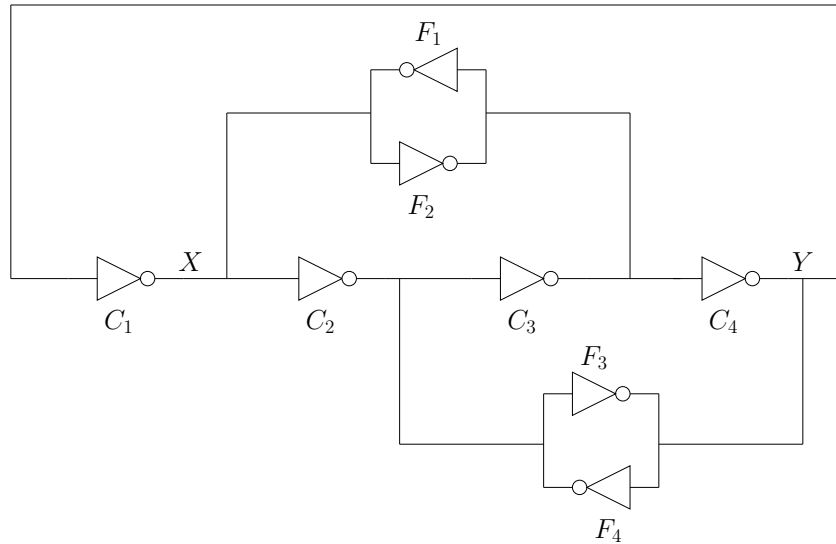


Fig. 2. A schematic diagram of a CMOS ring oscillator with feedforward inverters provided by Rambus. The chain inverters are labeled with C_i . The feedforward inverters are labeled with F_i .

begins to oscillate slowly with X leading Y as demonstrated by the simulation in Fig. 4. As the ratio increases the speed of oscillation increases. Approximately in the middle of the range of oscillating ratios, 1.60, the simulation in Fig. 5 demonstrates the stability of the oscillator. As the ratio increases to 2.275, the chain inverters begin to overpower the feedforward inverters, and the circuit struggles to begin to oscillate. It eventually starts oscillating and is stable after the initial startup period as demonstrated in Fig. 6. Finally, Fig. 7 shows the simulation results for a ratio of 2.30 where the chain inverters overpower the feedforward inverters and produce the expected result for a ring oscillator with an even number of inverters. In other words, the oscillator begins to oscillate but quickly reaches equilibrium with X and Y at opposite values and stops oscillating.

The behavior described above provides an interesting modeling and verification challenge. Using a single simulation, a designer may be convinced that the circuit oscillates. However, if the circuit is on the edge of the oscillating ratio, it is possible that process variation could cause regular failures of the circuit. Therefore, a model created from simulations near the center of the oscillating transistor ratios should upon analysis find that the circuit always oscillates. In contrast, a model generated from simulations near the edge of the acceptable region should to show the possibility of this failure. This result makes the designer aware of the potential failure for the inverter sizes being used. The designer could then do a set of simulations to characterize the oscillator and center the circuit within the acceptable operating range.

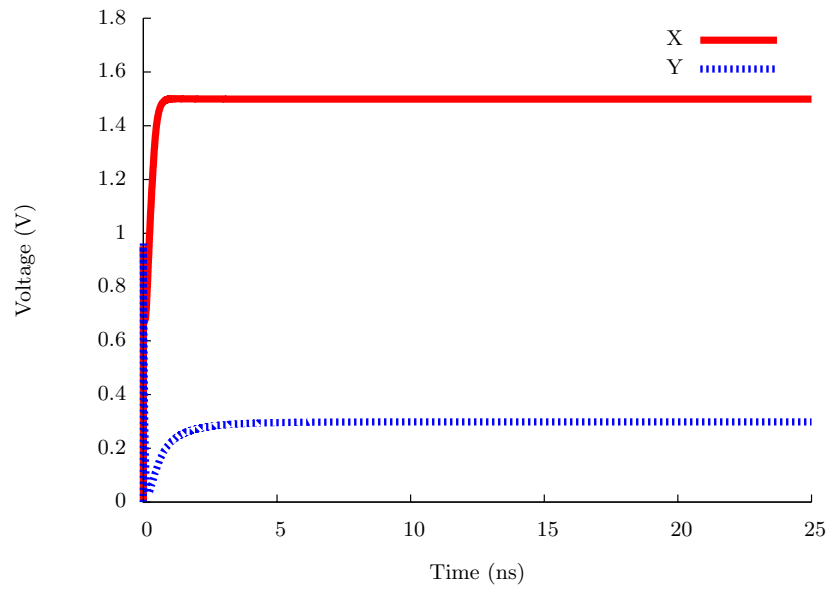


Fig. 3. CMOS ring oscillator with a ratio of chain to feedforward inverters of 0.39.

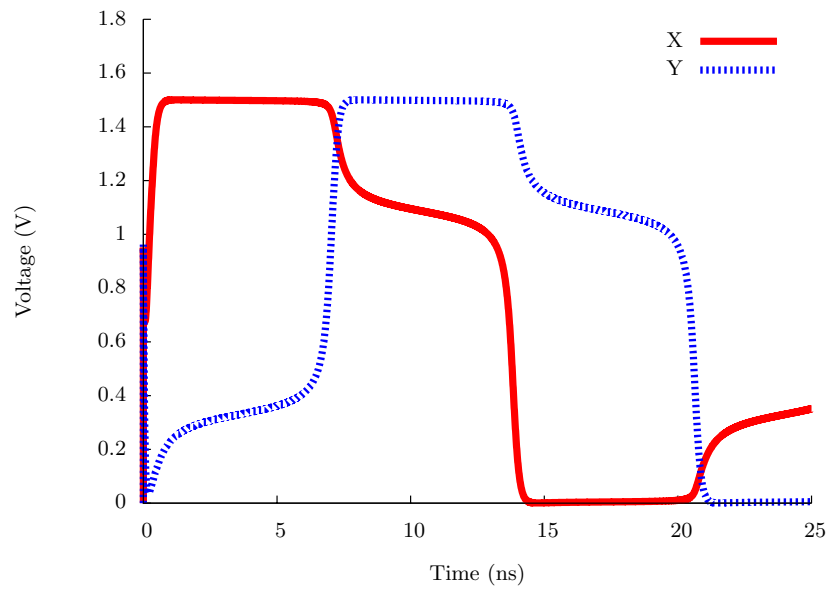


Fig. 4. CMOS ring oscillator with a ratio of chain to feedforward inverters of 0.40.

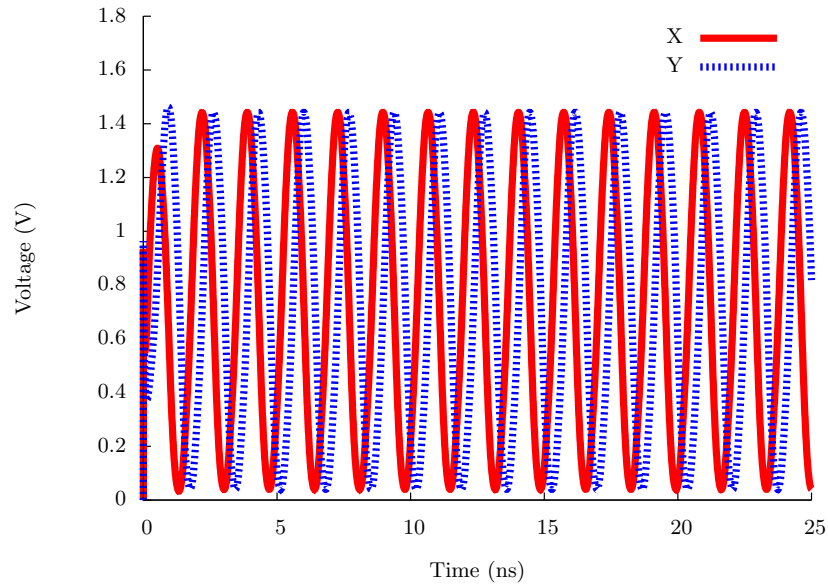


Fig. 5. CMOS ring oscillator with a ratio of chain to feedforward inverters of 1.60.

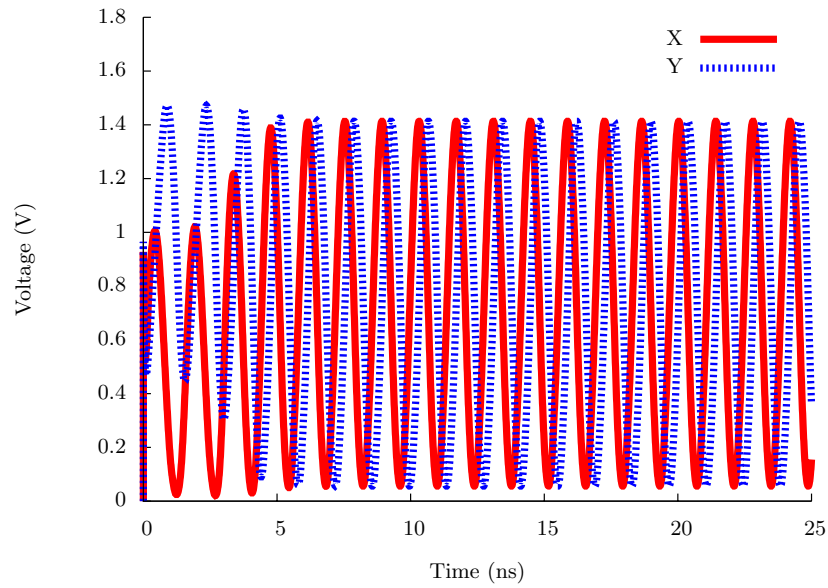


Fig. 6. CMOS ring oscillator with a ratio of chain to feedforward inverters of 2.275.

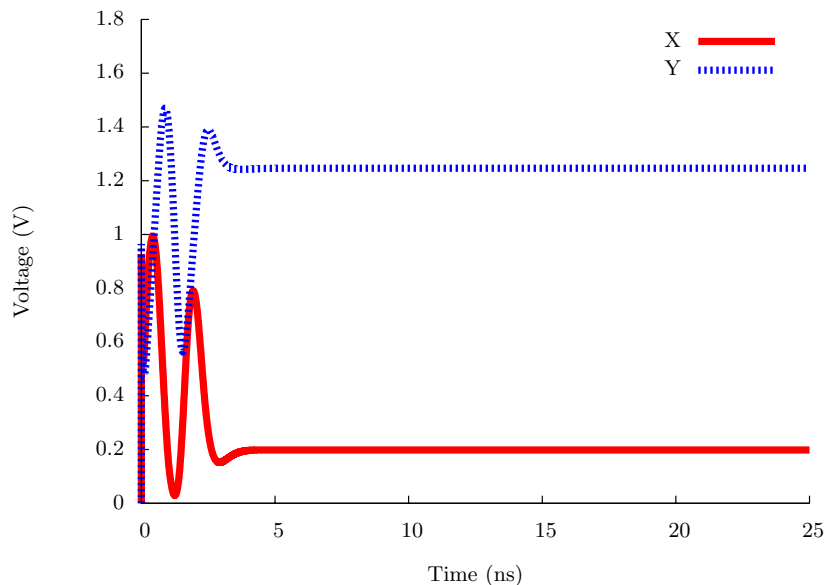


Fig. 7. CMOS ring oscillator with a ratio of chain to feedforward inverters of 2.30.

3 Abstract Modeling of AMS Circuits

Based on the success of formal methods for digital circuits, there has been an increasing body of work in formal methods for AMS circuits. Several tools and methods have been developed to explore the continuous state space of these systems [12–14, 4, 5, 7]. These methods work well on small examples and have shown some promise to work on larger circuits. One challenge for these methods is the significant effort required to create an appropriate abstract formal model for each circuit of interest. These methods also suffer from high computation costs for the analysis of the model. The more accurately the method explores the state space of the system, the more computationally intensive it is.

In response to these challenges, there has been recent work in verifying formal properties within the framework of simulation [15]. There are currently two main approaches for using simulation as a verification aid. The first approach attempts to find a finite number of simulation traces that are sufficient to represent all trajectories of the system and therefore prove correctness of the circuit [16–19]. The second approach uses simulation traces to generate a formal model which is then analyzed using a state space exploration engine [20]. The method used in LEMA [8, 9] is similar to the method by Dastidar, et al. [20] that generate a *finite state machine* (FSM) from a systematic set of simulation traces. Their FSM includes currents, voltages, and time as state variables to generate an acyclic FSM. The state space of the system is divided into symmetric state divisions. After each delta time step, the current state of the simulator is determined

and rounded to the center of the appropriate state division. The simulator is then started from this point and run for the next delta time step. This process continues until the global time reaches a user specified maximum. Conversely, our approach uses *Labeled Hybrid Petri Nets* (LHPNs) [4, 5] as the model. The state space is divided as specified by user provided thresholds on signal values. A global timer is not a part of the state space, so the graphs produced may include cycles. Simulation traces are run from start to finish without stopping allowing our model to preserve the original simulation trace.

Abstract modeling of AMS circuits is not a new idea. In fact, abstract or reduced order modeling of linear systems is a well understood discipline with a strong foundation [21–24]. Unfortunately, many systems of interest are nonlinear and abstract modeling of nonlinear systems is far from a solved problem. While there are promising solutions for parts of the space [25–28], investigations into the general problem have encountered accuracy and scalability problems [29–31]. As a result, automated abstract circuit models have not become common in AMS workflows. Our modeling work differs from this previous work in several ways. The most pronounced difference is the accuracy of the abstract model. Previous methods attempt to abstract the model while maintaining transistor-level accuracy. The abstract models produced by our model generator do not attempt to maintain this level of accuracy. They are intended to be used in system-level simulations to verify properties such as reasonable interaction between the digital and analog circuits. As a result, these models are less general, but the generation process and simulations using these models are much more efficient. The generated circuit models are conservative and model all the provided simulation traces plus additional behavior. By using simulations already produced by the designer, no additional simulation time is required. However, the quality of the model is directly related to the simulations used to create it. If the designer has inadequately simulated the design, the model may not exhibit the full behavior of the system. In this case, there is a potential that the actual circuit may have a failing behavior that is not included in the generated model.

Model generation in **LEMA** uses simulation traces that characterize the system, thresholds on the signal levels of selected design variables, and properties of the system to generate Verilog-AMS, VHDL-AMS, and LHPN models. The model generation process begins by dividing simulation data into regions of operation based upon user provided thresholds. Rates of change for each design variable are calculated for each region of operation. Rates are calculated using a sliding window technique that acts as a low pass filter to smooth out transitory pulses and effects created by threshold edges. The model generator then detects variables that have several stable values (e.g. digital signals) in order to model them as a set of constant values and not continuously varying values. Each of the three model types can be generated using information derived in these steps.

To model the ring oscillator, we have added an additional feature to **LEMA**'s model generator, the ability to limit the value of continuous variables in the LHPN model. Many continuous quantities are bounded by physical limits. For instance, the voltage in a circuit is often bounded by the voltage rails. The

model generator can now generate models that restrict the continuous values to predefined limits. As an example, consider a two variable system with a single threshold on each variable. In this case, there are four potential regions of operation labeled a-d in Fig. 8. Rates for each variable in each region as well as the direction of transitions found between the regions are calculated during the model generation process. To add the limiting behavior to the model, our method adds *pseudo-regions* e-p shown in Fig. 8 using dotted lines. These pseudo-regions are not reachable via simulation as the simulator imposes the appropriate physical limits on the circuit. The state space exploration engine for LHPNs, however, does not impose these same physical limitations and may end up exploring these pseudo-regions. In pseudo-regions, the rate of change for the limited variable is modified to prevent the variable from moving farther into a region of physical impossibility. For instance, in region a, if y increases above its upper threshold then moves to pseudo-region f it is prevented from increasing further by modifying $\dot{y} = [-5, 5]$ to $\dot{y} = [-5, 0]$. In region a, it is impossible for x to decrease below its limit as the rate of change for x is always positive. In this case, the rate is copied directly into pseudo-region i. The rates for all pseudo-regions are calculated similarly. Fig. 8 represents possible paths through regions of operation by annotating the boxes with direction arrows. These arrows indicate potential transitions between regions of operation. The solid arrows represent transitions observed in the simulation data. The dashed arrows represent transitions between observed regions of operation and pseudo-regions. These transitions have not been observed in the simulation data but are theoretically possible.

This box-like representation translates into a structurally similar LHPN as shown in Fig. 9. Each region translates to a place in the LHPN while each arrow translates to a transition. The enabling conditions for each transition are derived from the thresholds and limits. For instance, when moving from region a to region b the enabling condition on the transition is $\{x \geq 0\}$ as x must cross 0 to change regions. The rate assignments are set to the rates calculated for the place where the transition leads. In the transition from region a to b, the rate is set to $\langle \dot{x} := [-5, -2], \dot{y} := [2, 7] \rangle$, the rate of change for place p_b .

The original model generator only generated regions and transitions observed in the simulation data. By generating a model from several simulation traces, the model describes behavior not found in any simulation trace but that is inferred from the combination of traces. We believe that a model of this form accurately captures behavior presented in the simulation traces. However, the model may not contain behaviors of the system not exhibited by the simulation traces used to build the model. The question then arises if unobserved transitions or regions should be added to the model. Adding missing transitions between existing regions is straightforward and enhances the model by allowing potentially new transitions without dramatically increasing model complexity or analysis, so LEMA has been modified to add these transitions. Adding unobserved regions to the model is more difficult. The rate of change for the continuous variables is unknown for these regions. A methodology is needed to approximate a rate of change for these new regions. The pseudo-regions described previously are a

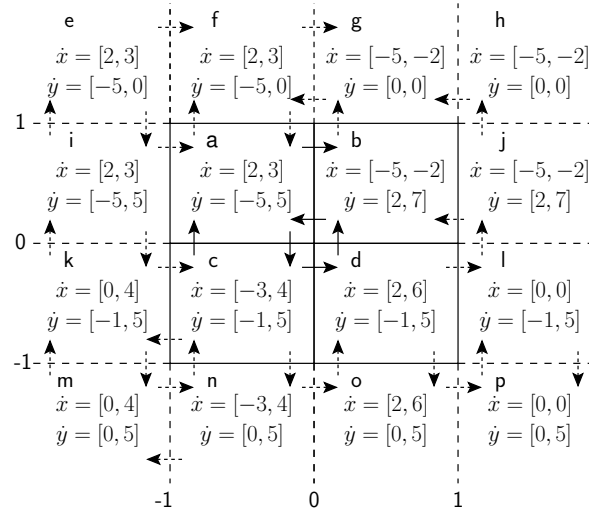


Fig. 8. An example of a two variable system where each variable has a single threshold and an upper and lower value limit.

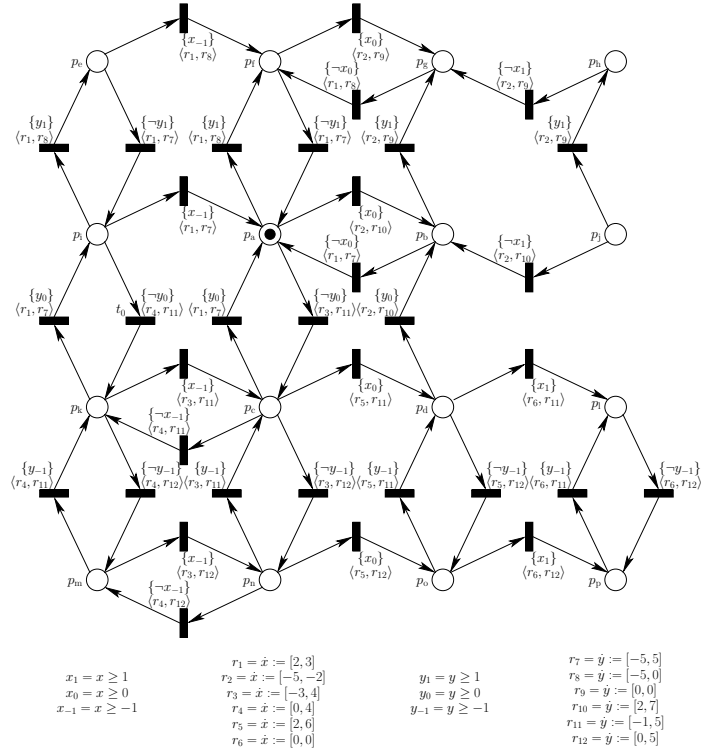


Fig. 9. An LHPN representing the regions in Fig. 8

special case of additional regions. The rate of change is calculated based on the knowledge that the value of the continuous variable should not exceed a specific limit. As a result, the rates are modified to prevent an excessive increase or decrease of the continuous variable. Since adding new regions increases model complexity substantially, LEMA currently only adds pseudo-regions.

The methodology just described can produce a model from the simulation data for the ring oscillator. The simulation data for the ratio of 1.60 produces the model shown in Fig. 10. The simulation data only includes the variables X and Y because other nodes in the inverter chain are correlated to X and Y . Correlated variables do not add any useful information to the generated model. These correlation relationships are determined manually through designer knowledge and experimentation. Following the solid arrows in this model sequentially from p0 to p15 shows that X and Y oscillate with X leading Y . Starting from p0 and moving to p4, X is decreasing as Y remains largely stable. Because Y lags X , in the progression from p4 to p8 Y decreases while X stabilizes. In the path from p8 to p12, X increases while Y is stable. To finish the cycle, Y increases while X remains stable. This is the only path through the region space found during the model generation. This single path indicates stable oscillation. The dashed arrows have been added as described above to allow additional behaviors that may be encountered during formal analysis of this model.

The simulation data for the ratio of 2.275 produces the model shown in Fig. 11. While this model still includes an oscillating path similar to the one in the previous model, it also includes numerous other paths that represent potentially non-oscillating or truncated oscillations. Since the regions are numbered in the order they are encountered, the initial path through the regions from p0 to p11 shows truncated oscillations. A path with non-truncated oscillations follows the following sequence: p0, p14, p2, p3, p4, p5, p19, p15, p6, p12, p18, p16, p17, p9, p13, p11, and back to p0. The presence of truncated and non-truncated oscillations should indicate to the designer that this set of transistor ratios is potentially on the margin of correct operation, and that the ratios likely need to be modified.

The major verification concern with the ring oscillator circuit is stability of oscillation. From the models produced by LEMA's model generator, it is possible to provide the designer with a measure of stability for the circuit. Based on a stability assurance measure provided by the verification tool for a given simulation, the designer would either be satisfied or use additional simulations to characterize the design space. The model built from the simulation with a ratio of 1.60 has a very high stability score as there is only a single oscillatory path through the state space. The model built from the simulation with a ratio of 2.275 would have a lower stability score as there are multiple oscillatory paths through the state space. The stability score for the second model is still reasonably high as all paths do oscillate. A model generated from a ratio of 2.30 would include paths through the state space that terminate without full, so it would have the lowest score. This type of stability analysis is area of future research interest.

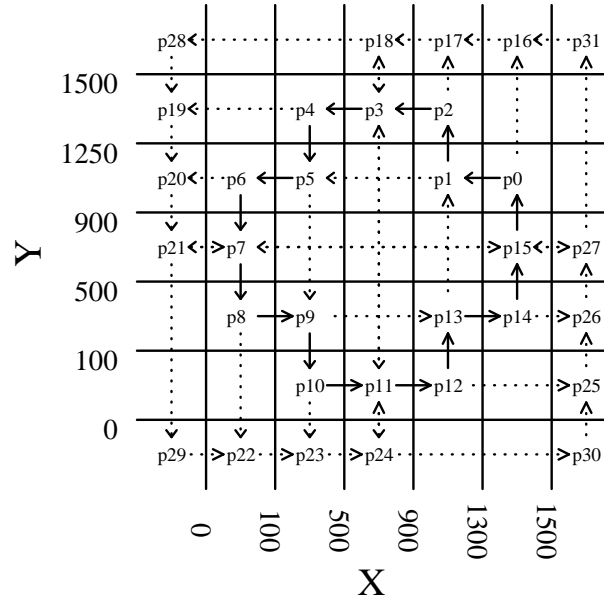


Fig. 10. An LHPN model of the ring oscillator for the simulation in Fig. 5

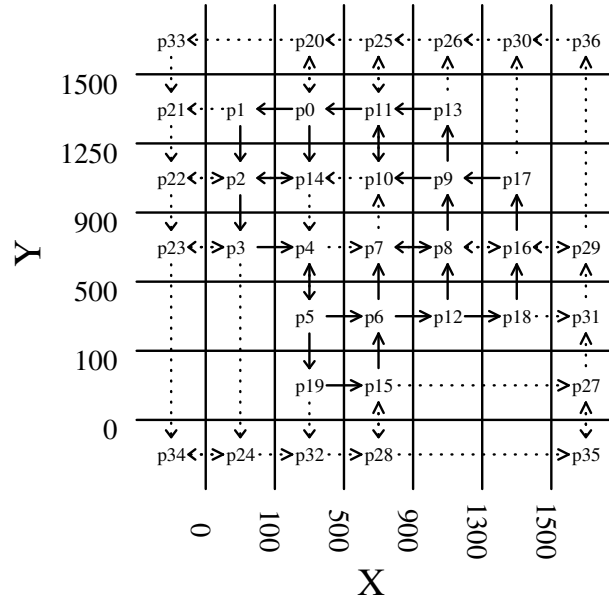


Fig. 11. An LHPN model of the ring oscillator for the simulation in Fig. 6

While our initial study of the ring oscillator is promising, it has also highlighted several areas of future work for the model generator. While the model generation process itself is automated, there are several user decisions required to configure the model generation process. The user must select a good set of design variables, thresholds for these design variables, and a window size for rate generation. Ideally, the user would select the minimal set of uncorrelated variables required to model the system. Modeling correlated variables increases the complexity of the LHPN model without adding any additional information to the model. Support can be added to the model generator to detect correlated variables and notify the user of this condition. Finding suitable thresholds for each selected design variable can also be difficult. We have begun work to automatically detect suitable thresholds given a user desired number of thresholds for each selected design variable. This method uses an optimization algorithm to search for the optimal set of thresholds in which the ranges on the rates are minimized. Further research is needed to find improved optimization algorithms and cost function for effective threshold generation. Finally, setting the window size for rate generation affects both the rates and modeled regions. A window size should be large enough to smooth out transitory pulses and other waveform artifacts, but it should be small enough to include regions that are traversed quickly. This requirement may be difficult to meet and modifying the model generation algorithm to employ a variable window size may prove useful to address this problem.

4 Coverage Metrics and Verification

Coverage information can be extracted from the set of simulations and is key to any SAV methodology. Coverage metrics serve to quantify verification completeness as well as identify portions of the design space that have not been adequately explored using the current set of simulations. Understanding the design coverage of the set of simulations used to generate the model is particularly important for our SAV methodology. When the verifier returns a verification result for a given model, it is important to understand what that result actually means. Namely, the completeness of the result depends on the completeness of the simulations used to build the abstract model. Ideally, the model generator would compute a set of coverage metrics that describe the model quality.

Our initial investigation in this area includes an implementation of a coverage metric where each simulation trace is given a score. A higher score is achieved by a simulation trace that exhibits behavior not previously seen. From the perspective of the LHPN model, new behavior is entering a previously unvisited region, taking a previously untaken region-to-region transition, or altering the range of a recorded value (rate, constant value, or delay). A metric of this type gives a qualitative measure of the utility of an additional simulation trace. This type of metric could be used to determine the benefit of doing further simulations.

The proposed coverage metric gives guidance as to whether a recently added simulation provides additional information to the model generator, but it does

not suggest if further simulations should be done or which simulations should be done. More sophisticated coverage metrics are needed to serve this purpose. Looking to digital coverage metrics for inspiration has not proved to be a profitable approach. Most notions of coverage for digital systems assume that each input combination may produce a unique and interesting output combination or are related to the RTL code [32]. This digital assumption does not hold for analog circuits whose output is often linearly related to the input. AMS designers also spend significant effort characterizing their designs for potential sources of variation. Coverage metrics for analog circuits need to account for these characteristics of analog design and then quantify the value of a set of simulations to explore the operating range of the circuit as well as the variation space. These metrics could then be used to suggest or even automatically run simulations to improve the model. For example, if a region is not visited during the current set of simulations, a new simulation can be run that has initial values to place it within the previously unexplored region.

Another way to help improve the quality of the model generation by leveraging information from the verification engine would be to use a counter-example guided abstraction-refinement methodology. When the verification engine returns an invalid error trace, the trace could be analyzed to determine the continuous variable and its value that creates the error. New thresholds could then be added to this variable around this value refining the model. More sophisticated approaches would likely also consider adding more continuous variables to the model or adjusting the window size used for rate generation. This abstraction-refinement loop could then be continued until a valid error trace is found or an adequately abstract model is verified.

5 Conclusion

AMS circuit verification is a growing concern as process variation and design complexity continue to grow. Talented AMS circuit designers have been able to design systems successfully, but automated modeling and verification tools are needed to take the next step in complexity and productivity for AMS designs. AMS designers work at the transistor-level, but abstract circuit models are needed for improved verification. We have developed a method to generate abstract AMS models for use in system-level simulations using simulation traces. These models can then be used for system-level simulation and verification. The results using our current SAV methodology are encouraging, but several hurdles remain in making the methodology usable by the typical AMS designer.

6 Acknowledgements

We would like to acknowledge Kevin Jones, Jaeha Kim, and Victor Konrad of Rambus for providing the ring oscillator example. We would also like to thank Alper Sen for helpful discussions on this work.

References

1. Chang, H., Kundert, K.: Verification of complex analog and RF IC designs. *Proceedings of the IEEE* **95**(3) (March 2007) 622–639
2. Accellera: Verilog-AMS Language Reference Manual: Analog & Mixed-Signal Extensions to Verilog HDL. Version 2.2 edn. (November 2004)
3. IEEE-SA Standards Board: IEEE Standard VHDL Analog and Mixed-Signal Extensions. (March 1999)
4. Little, S., Seegmiller, N., Walter, D., Myers, C., Yoneda, T.: Verification of analog/mixed-signal circuits using labeled hybrid petri nets. In: *Proc. International Conference on Computer Aided Design (ICCAD)*, IEEE Computer Society Press (2006) 275–282
5. Walter, D., Little, S., Seegmiller, N., Myers, C.J., Yoneda, T.: Symbolic model checking of analog/mixed-signal circuits. In: *Proc. of Asia and South Pacific Design Automation Conference (ASPDAC)*. (2007) 316–323
6. Walter, D.C.: Verification of analog and mixed-signal circuits using symbolic methods. PhD thesis, University of Utah (May 2007)
7. Walter, D., Little, S., Myers, C.: Bounded model checking of analog and mixed-signal circuits using an SMT solver. In Namjoshi, K.S., Yoneda, T., Higashino, T., Okamura, Y., eds.: *Automated Technology for Verification and Analysis (ATVA)*. Volume 4762 of *Lecture Notes in Computer Science.*, Springer (2007) 66–81
8. Little, S., Walter, D., Myers, C.: Analog/mixed-signal circuit verification using models generated from simulation traces. In Namjoshi, K.S., Yoneda, T., Higashino, T., Okamura, Y., eds.: *Automated Technology for Verification and Analysis (ATVA)*. Volume 4762 of *Lecture Notes in Computer Science.*, Springer (2007) 114–128
9. Little, S., Sen, A., Myers, C.: Application of automated model generation techniques to analog/mixed-signal circuits. In: *International Workshop on Microprocessor Test and Verification*. (December 2007)
10. Jones, K.D., Kim, J., Konrad, V.: Some "real world" problems in the analog and mixed signal domains. In: *Proc. Designing Correct Circuits*. (2008)
11. Grözing, M., Phillip, B., Berroth, M.: Cmos ring oscillator with quadrature outputs and 100 mhz to 3.5 ghz tuning range. In: *Proc. European Solid-State Circuits Conference*. (2003) 679–682
12. Hartong, W., Hedrich, L., Barke, E.: On discrete modeling and model checking for nonlinear analog systems. In Brinksma, E., Larsen, K.G., eds.: *Proc. International Conference on Computer Aided Verification (CAV)*. Volume 2404 of *Lecture Notes in Computer Science.*, Springer (2002) 401–413
13. Dang, T., Donzé, A., Maler, O.: Verification of analog and mixed-signal circuits using hybrid systems techniques. In Hu, A.J., Martin, A.K., eds.: *Formal Methods for Computer Aided Design (FMCAD)*. Volume 3312 of *Lecture Notes in Computer Science.*, Springer (2004) 21–36
14. Frehse, G., Krogh, B.H., Rutenbar, R.A.: Verifying analog oscillator circuits using forward/backward refinement. In: *Proc. Design, Automation and Test in Europe (DATE)*, IEEE Computer Society Press (2006) 257–262
15. Nickovic, D., Maler, O.: Amt: A property-based monitoring tool for analog systems. In: *Formal Modelling and Analysis of Timed Systems (FORMATS)*. (2007)
16. Donzé, A., Maler, O.: Systematic simulation using sensitivity analysis. In Bemporad, A., Bicchi, A., Buttazzo, G., eds.: *Hybrid Systems: Computation and Control (HSCC)*. Volume 4416 of *Lecture Notes in Computer Science.*, Springer (2007) 174–189

17. Dang, T., Nahhal, T.: Randomized simulation of hybrid systems for circuit validation. Technical report, VERIMAG (May 2006)
18. Girard, A., Pappas, G.J.: Verification using simulation. In Hespanha, J.P., Tiwari, A., eds.: Hybrid Systems: Computation and Control (HSCC). Volume 3927 of Lecture Notes in Computer Science., Springer (2006) 272–286
19. Fainekos, G.E., Girard, A., Pappas, G.J.: Temporal logic verification using simulation. In Asarin, E., Bouyer, P., eds.: Formal Modelling and Analysis of Timed Systems (FORMATS). Volume 4202 of Lecture Notes in Computer Science., Springer (2006) 171–186
20. Dastidar, T.R., Chakrabarti, P.P.: A verification system for transient response of analog circuits using model checking. In: VLSI Design (VLSID), IEEE Computer Society Press (2005) 195–200
21. Pillage, L.T., Rohrer, R.A.: Asymptotic waveform evaluation for timing analysis. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems **9**(4) (1990) 352–366
22. Grimme, E.J.: Krylov Projection Methods for Model Reduction. PhD thesis, University of Illinois at Urbana-Champaign (1997)
23. Odabasioglu, A., Celik, M., Pileggi, L.T.: Prima: passive reduced-order interconnect macromodeling algorithm. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems **17**(8) (1998) 645–654
24. Rutenbar, R.A., Gielen, G.G.E., Roychowdhury, J.: Hierarchical modeling, optimization, and synthesis for system-level analog and rf designs. Proceedings of the IEEE **95**(3) (March 2007) 640–669
25. Li, P., Pileggi, L.T.: Norm: compact model order reduction of weakly nonlinear systems. In: Proc. Design Automation Conference (DAC). (2003) 472–477
26. Phillips, J.R.: Projection-based approaches for model reduction of weakly nonlinear, time-varying systems. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems **22**(2) (2003) 171–187
27. Lai, X., Roychowdhury, J.: Tp-ppv: Piecewise nonlinear, time-shifted oscillator macromodel extraction for fast, accurate pll simulation. In Hassoun, S., ed.: Proc. International Conference on Computer Aided Design (ICCAD), ACM Press (2006) 269–274
28. Wang, Z., Lai, X., Roychowdhury, J.: Pv-ppv: Parameter variability aware, automatically extracted, nonlinear time-shifted oscillator macromodels. In: Proc. Design Automation Conference (DAC), IEEE Press (2007) 142–147
29. Rewieński, M., White, J.: Model order reduction for nonlinear dynamical systems based on trajectory piecewise-linear approximations. Linear Algebra and its Applications **415**(2–3) (jun 2006) 426–454
30. Dong, N., Roychowdhury, J.: Piecewise polynomial nonlinear model reduction. In: Proc. Design Automation Conference (DAC), New York, NY, USA, ACM Press (2003) 484–489
31. Tiwary, S.K., Rutenbar, R.A.: Faster, parametric trajectory-based macromodels via localized linear reductions. In Hassoun, S., ed.: Proc. International Conference on Computer Aided Design (ICCAD), ACM Press (2006) 876–883
32. Tasiran, S., Keutzer, K.: Coverage metrics for functional validation of hardware designs. IEEE Design & Test of Computers **18**(4) (2001) 36–45