

Analog/Mixed-Signal Circuit Verification Using Models Generated from Simulation Traces*

Scott Little, David Walter, Kevin Jones, and Chris Myers

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

Abstract. Formal and semi-formal verification of analog/mixed-signal circuits is complicated by the difficulty of obtaining circuit models suitable for analysis. We propose a method to generate a formal model from simulation traces. The resulting model is conservative in that it includes all of the original simulation traces used to generate it plus additional behavior. Information obtained during the model generation process can also be used to refine the simulation and verification process.

1 Introduction

Increased interest in system on a chip design has resulted in a need to improve validation methods for analog/mixed-signal (AMS) circuits. Validation of digital circuits has changed dramatically in the past ten years while AMS circuit validation remains largely the same. AMS circuit validation is still largely driven by designers using many simulation traces to validate specific properties of a circuit. While this methodology has been used with success for many years, recent trends are stretching it beyond its capacity. Increase in process variations and use of mixed-signal circuits present challenges that this simulation only methodology is not well prepared to address.

Currently, most AMS designers use an informal approach to circuit verification. With the aid of a simulator, the designer creates a circuit that under ideal conditions meets a set of specifications. A major concern for circuit designers using today's process technologies is the circuit's resilience to process variation. To help understand how the circuit operates under global variation, corner simulations are run. These simulations evaluate the circuit performance under combinations of change for common global variations such process, voltage, and temperature. There may also be local transistor to transistor process variation. To understand how this variation affects the circuit, Monte Carlo simulation is employed. These methods for exploring global and local variation are very expensive. This expense increases dramatically as more sources of variation are explored. As a result, only the most common sources of variation of the most critical specifications of the most critical circuits are thoroughly validated. The design team also has no real measure of the quality of the verification performed

* Support from SRC contract 2005-TJ-1357 and an SRC Graduate Fellowship.

on the design. The correctness of the design is almost solely the responsibility of each designer. The lack of feedback to the designer and large cost to verify the circuit under variation are major concerns when using this simulation only methodology.

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 [1–6]. 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 formal model for each different system. 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. 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 [7–10]. The second approach uses simulation traces to generate a formal model which is then analyzed using a state space exploration engine [11]. This paper describes a new method using the second approach.

Dastidar, et al. [11] generate a finite state machine (FSM) from a systematic set of simulation traces. This 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. A global timer is not a part of the state space which results in graphs that may include cycles. Simulation traces are run from start to finish without stopping allowing our model to preserve the original simulation trace.

The novelty of our approach is that the model allows for dynamic variation of parameters. Standard simulation based methods allow for changes in initial conditions and parameters, but these values are then fixed for the duration of the simulation run. Our model explores the system under ranges of initial conditions as well as ranges of dynamically changing parameter values. This additional behavior improves our ability to uncover variation induced errors.

The verification flow supported by our tool, **LEMA**, is shown in Fig. 1. Our previous work [4, 5, 12, 6] describes how a subset of VHDL-AMS can be compiled into an 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 [6]. This paper describes the flow

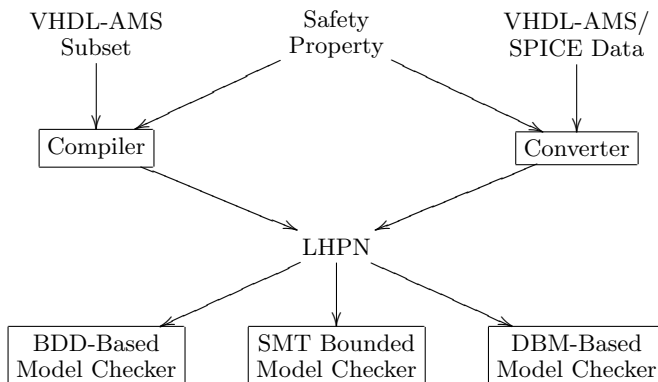


Fig. 1. LEMA: LHPN Embedded/Mixed-signal Analyzer

on the right side of Fig. 1 which takes simulation data, generates an LHPN, and uses one of these model checkers to verify the given property of the system. The remainder of this paper gives a brief introduction to LHPNs, describes the algorithms used to generate an LHPN model from a set of simulation traces, and concludes with a discussion of interesting metrics that can be extracted from the simulation data during model generation.

2 Motivating Example

The switched capacitor integrator circuit shown in Fig. 2 is a circuit used as a component in many AMS circuits such as ADCs and DACs. Although only a small piece of these complex circuits, the switched capacitor integrator proves to be a useful example illustrating the type of problems that can be present in AMS circuit designs. Discrete-time integrators typically utilize switched capacitor circuits to accumulate charge. Capacitor mismatch can cause gain errors in integrators. Also, the CMOS switch elements in switched capacitor circuits inject charge when they transition from closed to open. This charge injection is difficult to control with any precision, and its voltage-dependent nature leads to circuits that have a weak signal-dependent behavior. This can cause integrators to have slightly different gains depending on their current state and input value. Circuits using integrators run the risk of the integrator saturating near one of the power supply rails. Therefore, the verification property to check for this circuit is whether or not the voltage V_{out} can rise above 2000mV or fall below -2000mV . It is essential to ensure that this never happens during operation under any possible permutation of component variations. For simplicity, we assume for this example that the major source of uncertainty is that the capacitor C_2 can vary dynamically by ± 10 percent from its nominal value. This circuit, therefore, must be verified for all values in this range [13].

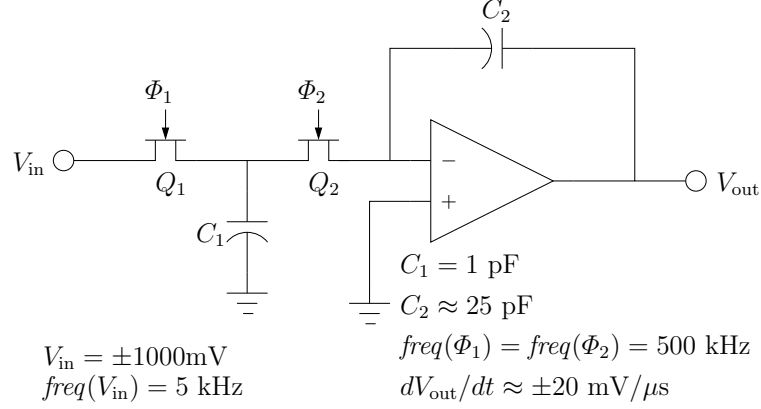


Fig. 2. A schematic of a switched capacitor integrator.

3 Labeled Hybrid Petri Nets

An LHPN is a Petri net model developed to represent AMS circuits [4, 12]. The model is inspired by features in both hybrid Petri nets [14] and hybrid automata [15]. An LHPN is a tuple $N = \langle P, T, B, V, F, L, M_0, S_0, Q_0, R_0 \rangle$ where:

P : is a finite set of places;

T : is a finite set of transitions;

B : is a finite set of Boolean signals;

V : is a finite set of continuous variables;

$F \subseteq (P \times T) \cup (T \times P)$ is the flow relation;

L : is a tuple of labels defined below;

$M_0 \subseteq P$ is the set of initially marked places;

S_0 : is the set of initial Boolean signal values;

Q_0 : is the set of initial ranges of values for each continuous variable and;

R_0 : is the set of initial ranges of rates for each continuous variable.

A key component of LHPNs are the labels. Some labels contain *hybrid separation logic* (HSL) formulas which are a Boolean combination of Boolean variables and separation predicates (inequalities relating continuous variables to constants). These formulas satisfy the following grammar:

$$\phi ::= \mathbf{true} \mid \mathbf{false} \mid b_i \mid \neg\phi \mid \phi \wedge \phi \mid c_i x_i \geq c_j x_j + c$$

where b_i are Boolean variables, x_i and x_j are continuous variables, and c_i , c_j , and c are rational constants in \mathbb{Q} . Note that any inequality between two real variables can be formed with \geq and negations of \geq inequalities. The labels permitted in LHPNs are represented using a tuple $L = \langle En, D, BA, VA, RA \rangle$:

- $En : T \rightarrow \phi$ labels each transition $t \in T$ with an enabling condition;
- $D : T \rightarrow |\mathbb{Q}| \times (|\mathbb{Q}| \cup \{\infty\})$ labels each transition $t \in T$ with a lower and upper bound $[d_l, d_u]$ on the delay for t to fire;
- $BA : T \rightarrow 2^{(B \times \{0,1\})}$ labels each transition $t \in T$ with Boolean assignments made when t fires;
- $VA : T \rightarrow 2^{(V \times \mathbb{Q} \times \mathbb{Q})}$ labels each transition $t \in T$ with a continuous variable assignment range, consisting of a lower and upper bound $[a_l, a_u]$, that is made when t fires;
- $RA : T \rightarrow 2^{(V \times \mathbb{Q} \times \mathbb{Q})}$ labels each transition $t \in T$ with a range of rates, consisting of a lower and upper bound $[r_l, r_u]$, that are assigned when t fires.

The semantics of the LHPN model are briefly illustrated using an LHPN model of the switched capacitor integrator shown in Fig. 3. A formal description of the semantics for LHPNs can be found in [12]. The output voltage, V_{out} , is modeled by the LHPN shown in Fig. 3a. The rate of the output voltage changes based on the value of V_{out} and the input voltage. The square wave input voltage, V_{in} , is modeled using the LHPN shown in Fig. 3b. V_{in} is modeled as a stable, multi-valued continuous quantity. Stable, multi-valued continuous quantities are modeled using continuous variables with a rate of zero and are updated using a variable assignment after a time delay. The LHPN shown in Fig. 3c is used to detect a failure. The enabling condition on the transition is the negation of an HSL formula for the safety property being verified. When this transition is enabled and fires, a failure is detected. In the initial state, p_0 , p_1 , and p_6 are marked; *fail* is **false**; V_{out} is -1000mV ; V_{in} is -1000mV ; the rate of V_{in} is 0; and the rate of V_{out} is 18 to 22 $\text{mV}/\mu\text{s}$. Initially, t_1 is the only enabled transition. However, as time passes, V_{out} crosses 0V enabling t_6 which fires immediately moving the token from p_6 to p_3 . After 100 to 101 μs from the initial state, t_1 fires and sets V_{in} to 1000mV. This change on V_{in} enables transition t_3 which fires immediately and sets the rate of V_{out} to be between -22 and $-17 \text{ mV}/\mu\text{s}$. Transition t_4 fires next in zero time when $V_{out} < 0V$. After this firing, transition t_2 fires after being enabled 99 to 100 μs . This firing sets V_{in} to -1000mV and enables transition t_5 which fires immediately and sets the rate of V_{out} to be between 17 and 22 $\text{mV}/\mu\text{s}$. This behavior continues until the range of V_{out} enables transition t_0 which fires and sets *fail* to **true**.

4 LHPN Model Generation

During the course of traditional analog circuit verification, designers run many different simulations to check that the circuit meets its specification. The goal of this work is to automatically generate an LHPN such as the one shown in Fig. 3 from simulation data. The generated LHPN model of the circuit is conservative and models 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

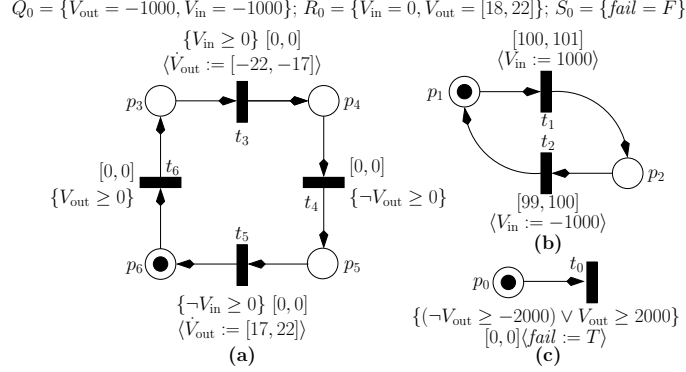


Fig. 3. A simple LHPN example for the switched capacitor integrator.

Algorithm 1: `GenerateLHPNfromData(data, thresholds, property)`

- 1 $binData = binData(data, thresholds);$
 - 2 $rates = calculateRates(data, binData);$
 - 3 $delays = calculateTimes(data, binData, rates);$
 - 4 $N = generateLHPN(binData, rates, delays, property);$
-

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. To help address this issue, Section 6 proposes the use of coverage metrics.

Algorithm 1 describes the process of taking simulation data and generating an LHPN. The input to our algorithm is time series simulation data, thresholds on the state space of the system, and the safety property to be checked specified using an HSL formula. The data is first sorted into bins based on the thresholds. Next, ranges of rates are calculated for each continuous variable within each bin. The algorithm assumes nothing about the dependence or independence of the rates. Each rate is calculated individually for each bin. It is expected that the rates change during different phases of operation. For this reason, it is important that thresholds are selected to separate the different phases of operation into distinct bins. At this point, continuous variables which are mostly stable but occasionally change are identified as variables that can be approximated by discrete transitions. Finally, after these calculations, the LHPN is generated.

Algorithm 1 is illustrated using two simulations of the switched capacitor integrator. In particular, the switched capacitor integrator is simulated with capacitance values of 23pF and 27pF for capacitor C_2 . The simulation data is recorded for the nodes representing the input voltage, V_{in} , and output voltage, V_{out} , during 400 μ s of transient simulation for each capacitance value. Part of the data from these simulations is shown in Tables 1 and 2.

Table 1. Partial simulation result with $C_2 = 23pF$ for the integrator.

Time (μs)	V_{in} (mV)	V_{out} (mV)	Bin	$\Delta V_{in}/\Delta t$ (mV/ μs)	$\Delta V_{out}/\Delta t$ (mV/ μs)	V_{in} time
0.00	-1000	-1000	00	-227.85	21.29	0.0
0.50	-1000	-999	00	0.0	21.74	0.5
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots
46.48	-1000	-0.4	00	-	-	46.48
46.98	-1000	10	01	0.0	21.74	46.98
47.48	-1000	21	01	0.0	21.74	47.48
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots
100.50	-1000	1174	01	-	-	100.50
100.54	-840	1174	01	-	-	100.54
100.62	-520	1176	01	-	-	100.70
100.78	120	1176	11	275.00	-21.08	0.08
101.00	1000	1174	11	0.0	-21.74	0.30
101.03	1.0	1173	11	0.0	-21.74	0.33
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots
154.98	1000	0.3	11	-	-	54.28
155.48	1000	-11	10	0.0	-21.74	54.78
155.98	1000	-21	10	0.0	-21.74	55.28
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots
200.00	1000	-978	10	-	-	99.30
200.04	840	-979	10	-	-	99.34
200.12	520	-980	10	-	-	99.50
200.28	-120	-981	00	-275.00	21.08	0.08
200.50	-1000	-978	00	0.0	21.74	0.30
200.53	-1000	-976	00	0.0	21.74	0.33
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots
400.00	1000	-957	10	-	-	99.34

The first step of Algorithm 1 is to bin the data based upon the thresholds provided. For this example, the thresholds chosen for both V_{in} and V_{out} are 0V. Each data file is analyzed and each time point is assigned to a bin based upon the values of V_{in} and V_{out} . In the data shown in Tables 1 and 2, each digit in the fourth column represents a bin. The first digit represents the V_{in} bin and the second digit represents the V_{out} bin. For instance, at time $100.50\mu s$ in Table 1, the bin assigned is 01 indicating that V_{in} is below 0V and V_{out} is above 0V. When V_{in} moves above 0V at time $100.78\mu s$, the bin assignment changes to 11.

The second step of Algorithm 1 calculates rate of change for each continuous variable. The rate of change is calculated for each bin using two time points within the same bin separated by a given interval. In the data shown in Tables 1 and 2 the interval is set to a length of ten. For example, the rate of change for V_{out} at time $46.98\mu s$ in Table 1 is calculated by looking at its value at this time

Table 2. Partial simulation result with $C_2 = 27pF$ for the integrator.

Time (μs)	V_{in} (mV)	V_{out} (mV)	Bin	$\Delta V_{in}/\Delta t$ (mV/ μs)	$\Delta V_{out}/\Delta t$ (mV/ μs)	V_{in} time
0.00	-1000	-1000	00	-227.85	18.14	0.0
0.50	-1000	-999	00	0.0	18.52	0.5
⋮	⋮	⋮	⋮	⋮	⋮	⋮
54.48	-1000	-0.3	00	-	-	54.48
54.98	-1000	9	01	0.0	18.52	54.98
55.48	-1000	18	01	0.0	18.52	55.48
⋮	⋮	⋮	⋮	⋮	⋮	⋮
100.50	-1000	852	01	-	-	100.50
100.54	-840	852	01	-	-	100.54
100.62	-520	853	01	-	-	100.70
100.78	120	854	11	275.00	-17.96	0.08
101.00	1000	852	11	0.0	-18.52	0.30
101.03	1000	850	11	0.0	-18.52	0.33
⋮	⋮	⋮	⋮	⋮	⋮	⋮
146.98	1000	0.3	11	-	-	46.28
147.48	1000	-9	10	0.0	-18.52	46.78
147.98	1000	-18	10	0.0	-18.52	47.28
⋮	⋮	⋮	⋮	⋮	⋮	⋮
200.00	1000	-981	10	-	-	99.30
200.04	840	-982	10	-	-	99.34
200.12	520	-983	10	-	-	99.50
200.28	-120	-984	00	-275.00	17.96	0.08
200.50	-1000	-981	00	0.0	18.52	0.30
200.53	-1000	-980	00	0.0	18.52	0.33
⋮	⋮	⋮	⋮	⋮	⋮	⋮
400.00	1000	-963	10	-	-	99.34

point and the value ten points later. This value is determined to be $21.74mV/\mu s$. After all the rates have been calculated, the minimum and maximum rates for each bin are determined. These values are the specified rate of change whenever the model is in that specific bin. The range of rates for each bin found from these two simulation runs for V_{out} from the switched capacitor integrator are shown in Table 3.

The third step of Algorithm 1 examines the rates for each continuous variable to determine if it can be reasonably approximated with a multi-valued continuous variable that makes discrete changes. This is true if a variable remains stable for large portions of time (i.e., has a rate of change that is nearly 0). In the switched capacitor integrator example, the square wave input voltage, V_{in} , is an example of this type of signal. In Tables 1 and 2, V_{in} has a rate of change of $0 mV/\mu s$ at most times. For these discrete signals, the algorithm determines the amount of

Table 3. Rates for V_{out} from the switched capacitor integrator.

Bin	Place	Range of rates	Comment
00	p_6	[17,22]	$V_{\text{in}} < 0V$; $V_{\text{out}} < 0V$
01	p_3	[17,22]	$V_{\text{in}} < 0V$; $V_{\text{out}} \geq 0V$
11	p_4	[-22,-17]	$V_{\text{in}} \geq 0V$; $V_{\text{out}} \geq 0V$
10	p_5	[-22,-17]	$V_{\text{in}} \geq 0V$; $V_{\text{out}} < 0V$

time that they spend at each discrete value. This is shown in the last column of Tables 1 and 2. The value of this continuous variable is then assigned to change at that specified time. For example, V_{in} is set to -1000mV and remains there for $100\mu\text{s}$ to $101\mu\text{s}$ after which it changes to 1000mV and remains there for $99\mu\text{s}$ to $100\mu\text{s}$. This cycle then repeats.

Using the information derived in the first three steps, Algorithm 1 can now generate an LHPN that models the provided simulation traces. A place is created for each bin discovered in the simulation traces. While in this example a place is produced for every bin assignment, in larger examples, many bins may never be encountered during simulation, so places are not generated for these unreachable bins. The places created for each bin from the integrator example are shown in the second column of Table 3. Next, transitions between bins are created when a transition between two bins is found in the simulation traces. It is theoretically possible that this could result in a fully connected graph, but in practice this is highly unlikely. Each transition is given an enabling condition representing the threshold that is being crossed to move from the first bin to the second. The delay for the transition is set to $[0,0]$ to make it fire immediately as the state of the system moves from one bin to the next. Finally, each transition is given a rate assignment to set the rate to the value for that bin as shown in Table 3. For the integrator example, the result is the LHPN shown in Fig. 3a. Note that the rate assignment is omitted for transition t_6 , since the range of rates for p_6 and p_3 are the same. Similarly, the rate assignment for t_4 can be omitted.

Next, a separate net is created for each discrete multi-valued continuous signal. A place is added for each discrete value of this variable. For the integrator, place p_1 is added for V_{in} equal to -1000mV , and p_2 is added to represent that V_{in} is equal to 1000mV . A transition is added for each discrete change found in the simulation data. The delay of this transition is determined by the time calculated in the previous step. Finally, this transition includes a continuous variable assignment to execute the discrete change. For the integrator example, the LHPN generated to control V_{in} is shown in Fig. 3b.

Finally, the last step is to create an LHPN to check the safety property provided as an HSL formula. This net has a single initially marked place and a single transition. The transition's enabling condition is the complement of the safety property. This transition has a delay of $[0,0]$, and it sets a special Boolean signal *fail* to true when it fires. Therefore, to verify this safety property, a model checker only needs to determine if there exists any state in which *fail* is true. For the integrator example, the LHPN generated to check if the circuit can saturate

is shown in Fig. 3c. Note that to cause analysis to terminate sooner, a Boolean condition, $\neg fail$, can be added to each transition in the LHPNs. This results in a deadlock once a failure is detected.

The LHPNs generated from simulation traces include ranges of rates. While these LHPNs can be directly analyzed using the BDD and SMT model checkers, they cannot be directly analyzed using the DBM method. To enable analysis of these LHPN models by the DBM model checker, a piecewise approximation of the range of rates is created by performing a transformation on the LHPN. In particular, this transformation allows the rate to change nondeterministically between the lower and upper bound on the range of rates. By exploring all of the possible nondeterministic rate changes the state space of the system for the entire range of rates is explored.

To simplify the description, the expansion process is illustrated using the LHPN in Fig. 4a which only has a threshold for V_{in} at 0V and no threshold for V_{out} resulting in just two bins represented with two places. The rate expansion proceeds by adding an additional transition and Boolean signal for each range of rates present in the unexpanded LHPN. The original transition is modified by changing the rate assignment to assign the lower bound of the range. Also, additional Boolean signal assignments are added to enable the firing of the upper bound of the rate and disable all other upper bound rate assignments. For example, in Fig. 4b the rate assignment on t_0 is changed from $[-22, -17]$ to -22 . The Boolean signal $v1$ is set to true enabling the firing of t_2 . The delay bound on t_2 is $[0, \infty]$ allowing t_2 to fire at any time in the future while the enabling condition remains satisfied. When t_2 fires it sets the rate to the upper bound, -17 and sets the Boolean signal $v1$ to false. This translation method has been implemented in the LEMA tool enabling LHPNs with ranges of rates to be analyzed by any of the model checkers in the tool.

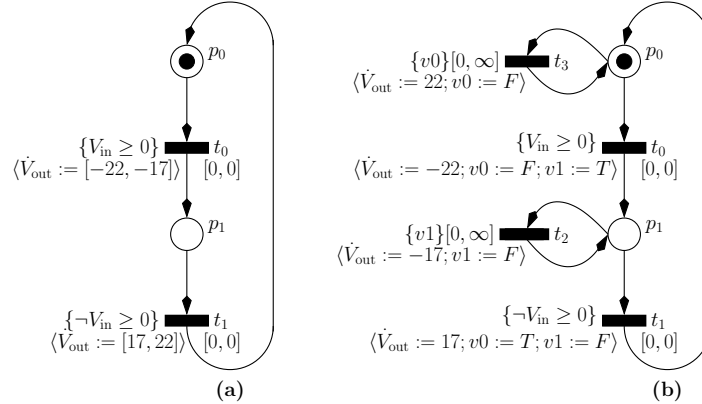


Fig. 4. LHPN demonstrating piecewise approximation of a range of rates.

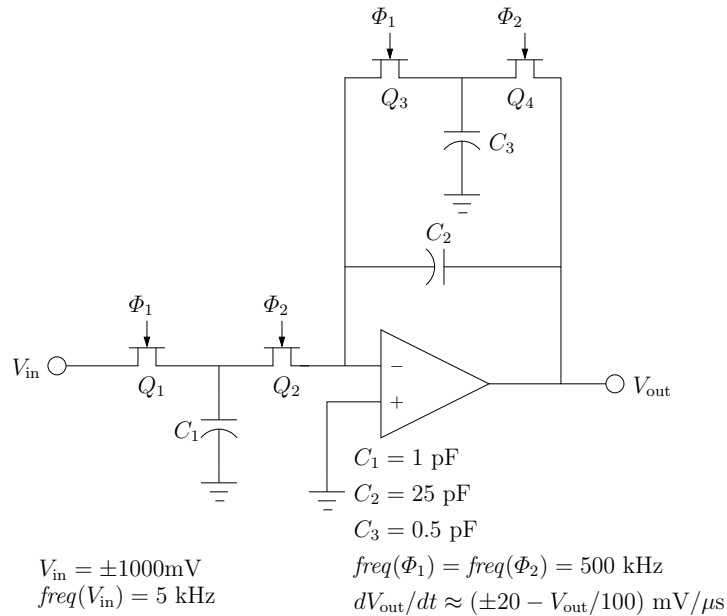


Fig. 5. Circuit diagram of a corrected switched capacitor integrator.

5 Case Study

Using Algorithm 1, two simulation traces of the switched capacitor integrator result in the LHPN shown in Fig. 3. Although neither of the simulation traces indicate a problem with saturation of the integrator, a state space analysis using the DBM model checker finds in less than a second that there is a potential for the circuit to fail. This failure can occur when the integrator charges the capacitor, C_2 , at a rate that is on average faster than the rate of discharge. This situation causes charge to build up on the capacitor and eventually results in V_{out} reaching a voltage above 2000mV. The reason that this method can find this failure is that the LHPN model represents not only each simulation trace, but also the union of the traces. It is this behavior explored by unioning the traces that allows the analyzer to find the flaw in the circuit.

Saturation of the integrator can be prevented using the circuit shown in Fig. 5. In this circuit, a resistor in the form of a switched capacitor is inserted in parallel with the feedback capacitor. This causes V_{out} to drift back to 0V. In other words, if V_{out} is increasing, it increases faster when it is far below 0V than when it is near or above 0V. Using the same simulation parameters and thresholds for this circuit, Algorithm 1 obtains an LHPN with the same structure as the one shown in Fig. 3, but the ranges of rates for each bin are as shown in Table 4. This LHPN also fails the property as the thresholds are too simple to capture the effect of the additional switched capacitor. Due to the addition of this switched capacitor resistor, the rate of change is now very dependent on the

Table 4. Rates for V_{out} in the corrected integrator using two bins.

Bin	Place	Range of rates	Comment
00	p_6	[18,32]	$V_{\text{in}} < 0V$; $V_{\text{out}} < 0V$
01	p_3	[9,22]	$V_{\text{in}} < 0V$; $V_{\text{out}} \geq 0V$
11	p_4	[-22,-9]	$V_{\text{in}} \geq 0V$; $V_{\text{out}} \geq 0V$
10	p_5	[-32,-18]	$V_{\text{in}} \geq 0V$; $V_{\text{out}} < 0V$

Table 5. Rates for V_{out} in the corrected integrator using four bins.

Bin	Place	Range of rates	Comment
00	p_9	[23,32]	$V_{\text{in}} < 0V$; $V_{\text{out}} < -500mV$
01	p_7	[18,27]	$V_{\text{in}} < 0V$; $-500mV \leq V_{\text{out}} < 0V$
02	p_5	[14,22]	$V_{\text{in}} < 0V$; $0 \leq V_{\text{out}} < 500mV$
03	p_3	[9,16]	$V_{\text{in}} < 0V$; $V_{\text{out}} \geq 500mV$
10	p_{10}	[-16,-9]	$V_{\text{in}} \geq 0V$; $V_{\text{out}} < -500mV$
11	p_8	[-22,-14]	$V_{\text{in}} \geq 0V$; $-500mV \leq V_{\text{out}} < 0V$
12	p_6	[-27,-18]	$V_{\text{in}} \geq 0V$; $0 \leq V_{\text{out}} < 500mV$
13	p_4	[-32,-23]	$V_{\text{in}} \geq 0V$; $V_{\text{out}} \geq 500mV$

value of V_{out} . In particular, this variation slows the rate of the voltage change as it approaches the power supply rail. This prevents saturation of the integrator. Based on this knowledge, the thresholds on V_{out} are changed to -500mV, 0V, and 500mV. These new thresholds result in the rates shown in Table 5. The LHPN for this table is shown in Fig. 6, and this LHPN is found to satisfy the property in less than a second using the DBM model checker. Finally, to explore the scalability of our algorithms, Table 6 shows how the size of the LHPN and model checking time scales as the number of thresholds increases.

6 Coverage Metrics

Our proposed method takes simulation traces from the designer and generates an LHPN. While the generated LHPN represents the behaviors that the designer deems to be important, it may miss problems not foreseen by the designer. Therefore, coverage metrics would be very useful to warn the designer about these unexplored portions of the state space where pitfalls may lie. Coverage information gives a quantitative metric about the quality of a set of simulation traces. This promises to aid the simulation only verification methodology as well as our model generation. We propose 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 some obvious examples of new behavior are entering a previously unvisited bin, taking a previously untaken bin to bin transition, or altering the overall rate of a bin. More complex measures of new behavior could be used such as the distance of the new trace from previously seen traces. A metric of this type gives a qualitative measure of the utility of an additional simulation trace. This

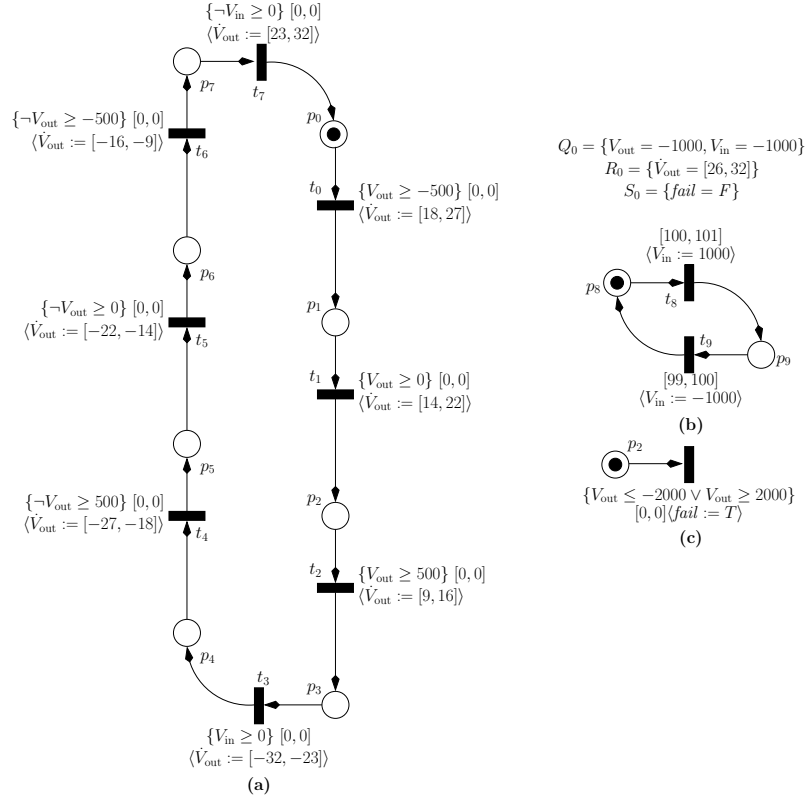


Fig. 6. LHPN for the corrected switched Integrator Example

Table 6. Scalability of model checking as number of thresholds increase.

No. thresholds	Places	Transitions	Model Checking Time
1	7	7	0.03s
3	11	11	0.06s
5	14	14	0.19s
7	18	19	0.31s
9	22	27	0.62s

type of metric could be used as an aid to determine the benefit of doing further simulations. A global metric for the entire set of simulation traces is also useful and could be created in a similar fashion.

For the integrator example, using just the simulation trace shown in Table 1 with C_2 equal to 23pF would result in the LHPN shown in Fig. 3. Adding the simulation trace shown in Table 2 with C_2 equal to 27pF results in the exact same LHPN structure, but the ranges of rate for V_{out} would be changed. Therefore, the value of the second trace run is less than that of the first, but it still has some value. Finally, if a third trace with C_2 equal to 25pF is added at this point, the resulting LHPN would not change at all as the rates generated from this trace would be contained in those generated from the first two. Therefore, this trace adds no new knowledge, so the coverage metric would say that it has no value. As a final example, if a trace is added that changes V_{in} at twice the frequency (i.e., every $50\mu s$), it now becomes possible for V_{in} to change before V_{out} goes above $0V$. This means that the LHPN generated would now have a new transition from p_6 to p_5 . This LHPN would also have a wider range of delays for when V_{in} changes. Therefore, this additional trace provides new information.

7 Conclusion

Interest in formal and semi-formal methods for validating AMS circuits is increasing. Many of these methods are seriously handicapped by the difficulty of generating formal models. This paper develops a method to generate a conservative, trace preserving formal LHPN model from a set of simulation traces and thresholds on the state space. This LHPN model can be used by several different model checking engines to prove safety properties about the entire state space of the model. Using two variations of the switched capacitor integrator circuit, this paper shows how an adequate LHPN model can be created using two simulation traces and a basic set of thresholds. The model is analyzed using a DBM based model checker to obtain the expected verification results.

While the current version of the tool requires the user to provide thresholds, it would be interesting to explore automated methods to determine important thresholds. Our initial investigations into the autogeneration of thresholds attempt to increase the number of thresholds in regions where the rates change rapidly. Automatic generation of thresholds may also provide the designer with useful information about the circuit.

Another potential benefit of the method described in this paper is that an LHPN model can be translated into a VHDL-AMS or Verilog-AMS model. One problem for AMS designers is creation of abstract models of their circuit for use in a digital or mixed-mode simulation flow. Models can be created by hand but must be updated to remain consistent as circuits change. Using this method, the models could maintain their consistency by running the needed set of simulations after changes and regenerating the HDL model from the LHPN.

References

1. 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
2. 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
3. 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
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: Asia and South Pacific Design Automation Conference (ASPDAC). (2007) 316–323
6. Walter, D., Little, S., Myers, C.: Bounded model checking of analog and mixed-signal circuits using an SMT solver. In: Automated Technology for Verification and Analysis (ATVA). Lecture Notes in Computer Science, Springer (2007)
7. 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)
8. Dang, T., Nahhal, T.: Randomized simulation of hybrid systems for circuit validation. Technical report, VERIMAG (May 2006)
9. 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
10. Fainekos, G.E., Girard, A., Pappas, G.J.: Temporal logic verification using simulation. In: Formal Modelling and Analysis of Timed Systems (FORMATS). Volume 4202 of Lecture Notes in Computer Science., Springer (2006) 171–186
11. Dastidar, T.R., Chakrabarti, P.P.: A verification system for transient response of analog circuits using model checking. In: VLSI Design, IEEE Computer Society Press (2005) 195–200
12. Walter, D.C.: Verification of analog and mixed-signal circuits using symbolic methods. PhD thesis, University of Utah (May 2007)
13. Myers, C.J., Harrison, R.R., Walter, D., Seegmiller, N., Little, S.: The case for analog circuit verification. *Electronic Notes Theoretical Computer Science*. **153**(3) (2006) 53–63
14. David, R., Alla, H.: On hybrid petri nets. *Discrete Event Dynamic Systems: Theory and Applications* **11**(1–2) (January 2001) 9–40
15. Alur, R., Courcoubetis, C., Henzinger, T.A., Ho, P.H.: Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems. In Grossman, R.L., Nerode, A., Ravn, A.P., Rischel, H., eds.: Hybrid Systems. Volume 736 of Lecture Notes in Computer Science., Springer (1992) 209–229