Equivalence Checking of Nonlinear Analog Circuits for Hierarchical AMS System Verification

Sebastian Steinhorst[1] Lars Hedrich[2]

[1] TUM CREATE, Singapore, Email: sebastian.steinhorst@tum-create.edu.sg
[2] University of Frankfurt/Main, Germany, Email: hedrich@em.cs.uni-frankfurt.de

***Abstract*—In this contribution a novel formal methodology for equivalence checking of analog circuits is proposed. In order to prove the behavioral equivalence of two circuit implementations such as a transistor netlist and a corresponding behavioral model, guaranteed coverage of the complete reachable state space for each of the two circuits under verification is obtained by an efficient input stimuli generation algorithm. These input stimuli are processed by a conventional circuit simulator to obtain simulation results covering each system's complete dynamic behavior. By automatically comparing the simulation results using specific error measures, the level of equivalence of both systems is determined. Simulation by complete state space-covering input stimuli guarantees the equivalence checking results to be sound for every possible state and input stimulus of the circuits under verification, which allows safe application of analog behavioral models in hierarchical AMS system simulation flows. The application to example circuits shows the feasibility of the approach.**

I. Introduction

Industrial analog/mixed-signal (AMS) design flows rely on analog circuit simulation as their central tool for design characterization of the analog parts. While the final design has to be considered on transistor level, behavioral modeling of circuit blocks is necessary in order to efficiently simulate the complete system design or to make complete system simulation possible at all during the design process. Especially when dealing with analog circuits, replacement of transistor level blocks by behavioral models brings up a critical uncertainty. If a behavioral model does not sufficiently cover the behavior of the modeled transistor block, simulation results are misleading.

Up to now, analog transistor blocks are compared to their behavioral models by manually observing their transient simulation response to input stimuli selected by circuit designers. While this approach can reveal some errors in the model, there is no confidence that the model will behave as expected for all possible input signals and internal states. Consequently, formal methods are necessary to overcome this uncertainty for analog behavioral modeling to complement the successful application of formal equivalence checking in the digital domain of AMS-systems.

Analog equivalence checking will target this verification gap by introducing a methodology of validating analog circuit blocks, allowing to prove that a behavioral model will not introduce unexpected behavior into a hierarchical AMS-system simulation. In contrast to the definition of equivalence in the digital domain, analog equivalence checking in the context of this paper proves that a determined worst case difference of the two circuit implementations is not exceeded for all possible input signals and internal states. In addition to application to behavioral models, the proposed approach could also be used to check structural transformations or extracted netlists.

A first formal approach to equivalence checking of analog circuits proposes to transform the state space representation of two circuit implementations into a canonical form and compare sampling points within this state space according to their dynamic and output behavior [1]. While this approach is successful for behavioral models with an internal structure not too different from the modeled transistor block, strong abstraction of the behavioral model can result in reporting complete inequality due to the need for an internal mapping of state variables which is only possible with certain similarities in the systems' structure. Allowing to measure the level of equivalence of two waveforms but not offering a method for complete coverage of the system's state space, other approaches [2], [3] cannot be considered as formal verification.

The new approach for equivalence checking of analog circuits presented in this paper will retain formal completeness but will work well with any kind of circuit abstraction. Due to the application of conventional transient circuit simulation, each step of the equivalence checking process will be observable by the circuit designer and is mostly based on tools he already is used to. The key feature of this equivalence checking flow is the application of complete state space-covering input stimuli that allow to efficiently simulate and compare the completely covered state space behavior of two circuit implementations under verification. A first approach to generating such complete state space-covering input stimuli has been presented in [4]. In contrast to other non-formal approaches that simulate the system behavior with statistical chaotic excitation signals [5] or approaches in the area of test generation using either sensitivity analysis [6], controllability and observability computation [7], or statistical distance computation [8], the stimuli generation approach utilized in this paper is based on a systematic traversal of the system's state space and hence complete behavioral coverage can be proven. This is the significant difference to other recent semi-formal approaches that either limit the the investigated input space to highly likely regions [9] or by using a simplified modeling by application of linearized models [10].

This paper is organized as follows. In section II, the discrete state space modeling will be discussed which is necessary for application of the input stimuli generation algorithm described in section III. The new equivalence checking methodology, enabled by the possibility of simulation with guaranteed complete state space coverage, is

This work was financially supported in part by the Singapore National Research Foundation under its Campus for Research Excellence And Technological Enterprise (CREATE) programme.

presented in section IV with a description of the applied error measure in section V. The new methodology will be applied to circuit examples in section VI. Section VII concludes.

II. Discrete State Space Modeling

For the application of the stimuli generation algorithm, the analog circuit description has to be transferred into a discrete graph data structure as described in detail in [11]. For this purpose, by applying modified nodal analysis (MNA) to the circuit netlist, a nonlinear first order differential algebraic equation (DAE) system

$$f(\dot{x}(t), x(t), u(t)) = 0$$

representing the input vector $u(t)$ and the vector of the system variables $x(t)$ is created. The state space with n_d dimensions of the resulting DAE system is spanned by the system variables of the energy-storing elements $x_e(t)$ and the input variables. By comparing length and angle of the system variables' derivatives $\dot{x}_e(t)$, the system's state space is divided into areas of homogeneous behavior, represented by hypercubes. The transition relation of the hypercubes and the transition time are defined by the system variables' derivatives. Finally, the transition system represents a graph data structure, considering each hypercube as a vertex of the graph with directed edges defined by the transition relation with the corresponding transition times. Each vertex of the graph is labeled with the state space variable values at the center of the corresponding hypercube. With these considerations, an analog transition system can be defined as follows.

Definition 1 (Analog Transition System (ATS))
The ATS M_{ATS} is a four-tuple $M_{ATS} = (\Sigma, R, L_V, T)$ where

- *Σ is a finite set of states of the system.*
- *$R \subseteq \Sigma \times \Sigma$ is a total transition relation, hence for every state $\sigma \in \Sigma$ there exists a state σ' such that $(\sigma, \sigma') \in R$.*
- *$L_V : \Sigma \rightarrow \mathbb{R}^{n_d}$ is a labeling function that labels each state with the vector of n_d variables containing the values in this state of the state space variables of the DAE system.*
- *$T : R \rightarrow \mathbb{R}_0^+$ is a labeling function that labels each transition from σ to σ' with a real valued positive or zero transition time that represents the time required for the trajectory in the state space between these states. Within the structure M_{ATS}, a path π beginning at state σ is a sequence of states $\pi = \sigma_0, \sigma_1, \sigma_2, ..., \sigma_n$ with $\sigma_0 = \sigma$ and $(\sigma_i, \sigma_{i+1}) \in R$ for $0 \leq i < n$.*

Figure 3 summarizes the described discrete state space modeling process on which the stimuli generation algorithm will operate.

While the complexity of the state space modeling process is exponential in the number of energy-storing elements and inputs of a circuit, relevant analog circuit blocks usually do not exceed a system order of 8, which can be handled well by this approach. Moreover, by application of an Eigenvalue-based model order reduction of the DAE system [1], circuits with more than 200 parasitic capacitances can be handled. This is achieved by reducing the state space to the dominant state variables of a system and discarding the parasitic ones which are mathematically proven not to affect the system behavior above a defined threshold.

Figure 1 illustrates a schematic ATS graph structure with 9 vertices modeling an imaginary analog circuit with an input and one state space variable connected to the output. The DC operating points of the modeled circuit are represented by vertices 1, 5, and 9. Thus they have a loop transition to themselves stating that the circuit stays in this steady state infinitely until a change of input occurs. A transition induced by an input change is modeled by a bidirectional edge, implying that this transition can only be taken if there is an input change in the according direction. Any non-steady state of the system has outgoing directed edges representing the dynamic behavior of the circuit.

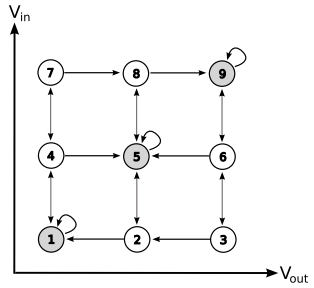

Fig. 1. Schematic illustration of an ATS graph structure created by the discrete modeling process.

The described modeling process is implemented as an extension to the Infineon Titan Simulator [12] and the analog modeling and verification tool amcheck [11].

III. Stimuli Generation Algorithm

On the discretized state space model, represented by a graph structure, input stimuli covering the complete reachable area of the state space of the analog circuit can be computed by traversing the graph. The stimulus is created by recording the input values and accumulated times of traveled edges during graph traversal.

Definition 2 (Complete State Space-Covering Input Stimulus on an ATS)
A complete state space-covering input stimulus for the analog circuit model represented as ATS is a path π_{is} from a defined starting state σ_0 that visits every state of the set of reachable states ϕ of the ATS. The set Π_{is} consists of all π_{is} satisfying this requirement.

$$\Pi_{is} = \{\pi_{is} \,|\, \exists\, n : \pi_{is} = \sigma_0, ..., \sigma_n \wedge \bigcup_{0 \leq i \leq n} \sigma_i = \phi\} \quad (1)$$

$$\pi_{is} \in \Pi_{is} \quad (2)$$

This corresponds to the idea not only conducting a subsequent transient simulation that brings the system under verification into every reachable state but also allowing to simulate every possibility how every state can be reached. In the following, the algorithm assumes the goal of complete state and dynamic transition coverage. In order to do this more efficiently than just generating a sequence of counterexamples, an algorithm is needed which satisfies the following requirements:

- When the algorithm terminates, every reachable state and transition of the circuit model, represented by the vertices and edges of the graph, must have been visited at least once.
- The number of traveled edges on the paths covering the complete state space shall be minimized as each timed transition taken between two vertices of the graph results in an increment

Algorithm 1: Complete-Coverage Input Stimuli Generation Algorithm.

Input: ATS modeling the analog circuit, vertex σ_i representing a DC operating point
Output: List of tuples (value, time) representing piecewise linear stimuli for every input variable covering the complete state space

1 open = all reachable dynamic edges;
2 closed = $\emptyset$;
3 **foreach** *edge* j *reachable from* σ_i *in open* **do**
4 calculate path covering as many edges as possible from $\sigma_i \rightarrow j \rightarrow \sigma_i$ avoiding edges in closed;
5 **foreach** *edge* k *visited on path* $\sigma_i \rightarrow j \rightarrow \sigma_i$ **do**
6 put k to closed;
7 remove k from open;
8 store value vector $L_V(\sigma_l)$ of visited vertices σ_l and path time;
9 **end**
10 **end**

of the time length of the input stimulus and is therefore affecting simulation time.

- During stimulus generation, if available, vertices representing DC operating points shall be visited periodically. This ensures that the circuit can recover from the traversal of corners of its dynamic behavior by starting and ending in its steady states.

Combining the above requirements into an algorithm reveals the NP-hardness of the optimization problem as it is a modification of the traveling salesperson problem [13]. Accordingly, a heuristic approach is necessary for efficient computation. Due to the fact that any path that covers all reachable edges and states of the graph is a valid solution, an efficient algorithm using a heuristic approach will produce a valid solution with an assumed suboptimal path length. Algorithm 1 shows a possible efficient approach and is described in the following.

For a given DC operating point σ_i, the algorithm computes a list of tuples (value, time) representing piecewise linear stimuli for every input variable covering the complete state space. Initialization of variables includes setting the set *open* to all reachable edges and initializing the set *closed* as empty in lines 1 and 2. Starting in line 3, for each edge j in the set open, a path covering as many edges as possible from $\sigma_i \rightarrow j \rightarrow \sigma_i$ is computed avoiding edges in the set *closed* in line 4. Line 5 iterates over each edge k visited on the computed path $\sigma_i \rightarrow j \rightarrow \sigma_i$ and puts k to the set *closed* in line 6, removes it from set *open* in line 7 and stores the parameter tuple creating the piecewise linear stimulus to a file in line 8.

The implicitly mentioned path finding algorithm in line 4 can be any form of a longest path finding algorithm on a directed acyclic graph such as a modification of Dijkstra's algorithm applied with negative edge weights for longest path detection. For obtaining the longest path with respect to the number of vertices visited, edge weights have to be set to -1. Other optimization criteria are possible, such as considering euclidean vertex distance or the original edge weights containing transition time values. At first, computing the shortest time path might seem like a obvious solution, but as the goal of the stimuli generation algorithm is complete edge and vertex coverage, the number of inevitably revisited edges during single closed loop runs has shown to be high, resulting in worse overall stimuli time length.

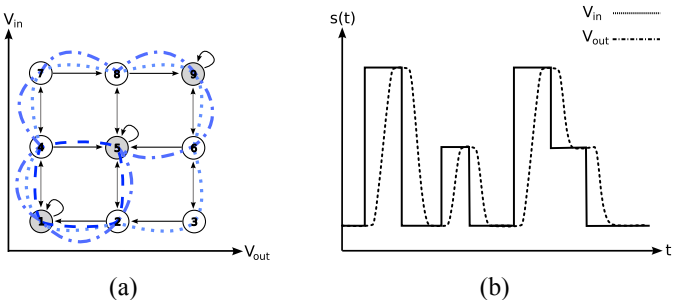

Fig. 2. Path generated by the stimuli generation algorithm (a) and the corresponding input/output behavior (b).

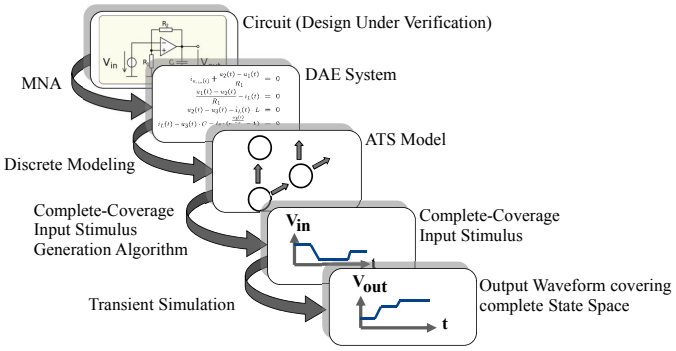

Fig. 3. Discrete state space modeling of analog circuits for complete-coverage input stimuli generation.

In order to avoid revisiting edges, all edges in the set *closed* have to be assigned with a positive value. Trying to minimize the sum of edge weights, the path finding algorithm will automatically avoid those edges that are contrary to the optimization criterion.

The asymptotic runtime complexity of the stimuli generation algorithm for a graph with n reachable edges is dominated by the loop over each edge in the set *open* and the call to Dijkstra's algorithm having quadratic complexity inside this loop. This results in an asymptotic worst case complexity of $O(n^3)$.

Figure 2a shows a possible traversal result generated by the stimuli generation algorithm applied to the graph from Figure 1, starting from vertex 1. The first loop $1 \rightarrow 4 \rightarrow 7 \rightarrow 8 \rightarrow 9 \rightarrow 6 \rightarrow 3 \rightarrow 2 \rightarrow 1$ is created due to vertex 9 being the most distant vertex from 1, thus covering the most edges of the graph. Vertex 5 is still unvisited, therefore a second loop run is needed, traveling vertices $1 \rightarrow 4 \rightarrow 5 \rightarrow 2 \rightarrow 1$. The input-output behavior representing the stimulus and a possible transient response is illustrated in Figure 2b. While any traversal policy covering the complete graph is valid, further investigation of better strategies is necessary as they directly result in shorter simulation times. The input stimuli generation flow based on discrete state space modeling is illustrated in Figure 3.

IV. Equivalence Checking Methodology

The simulation of a single circuit using complete state space-covering input stimuli can reveal corner case behavior not identified by user-defined input stimuli. However, the focus of this contribution

is to introduce an equivalence checking methodology for analog circuits based on the stimuli generation approach, giving certainty about the level of equality between two circuit implementations. The idea is to generate a stimulus for the system that covers the system's complete state space during a transient simulation. If another circuit is simulated using the same input stimulus, the level of equivalence of the two systems is determined by the level of deviation of the transient responses of the two circuits.

For each of the two circuits to compare, in the following referred to as circuit A and circuit B, complete state space-covering input stimuli are generated for every input of the circuit. Subsequently, four simulation runs are needed. Circuit A is simulated with stimuli of A and B, followed by simulating circuit B with stimuli of A and B. The simulation results are automatically compared using an error measure as described in detail in section V, reporting equivalence if a user-specifiable maximum error value is not exceeded.

If circuits A and B show equivalent behavior when simulated with stimuli generated from circuit A, then the complete behavior of circuit A is included in circuit B:

$$A \subseteq B \tag{3}$$

If circuit A and B show equivalent behavior for simulation with stimuli generated from circuit B, then the complete behavior of circuit B is included in circuit A:

$$B \subseteq A \tag{4}$$

If both conditions (3) and (4) hold, circuit A and circuit B are considered as equivalent with respect to the user-defined maximum error:

$$A \subseteq B \quad \wedge \quad B \subseteq A \quad \Longrightarrow \quad A \approx B \tag{5}$$

Figure 4 illustrates the described equivalence checking methodology.

In practical applications, often only one direction of the proof is necessary. Especially for reduced models generated with model order reduction techniques, the complete-coverage stimuli have to be generated only for the reduced model in order to prove that the transistor netlist behaves equal for the limited state space of the model during simulation. Of course, the other direction of the proof would fail as the reduced model intentionally does not cover all aspects of the transistor netlist, such as behavior above or below certain operating frequencies. On the other hand, for the comparison of extracted netlists or structural transformations, proving equivalence in both directions is suggested.

V. Error Measures for Waveform Comparison

For a complete automation of the equivalence checking flow, the differences between the simulation results of the two circuits under verification have to be computed by an error calculation algorithm. While there are several measures to calculate the error between two waveforms like Frechet distance [14], modified Hausdorff distance [15], etc., the most intuitive measure is the generation of a difference waveform of the signals. Therefore, for each time point of the two waveforms A and B, the value at this time point of the other waveform is calculated and the difference value is stored. The maximum difference between the waveforms is the reported error value and the results can be inspected by plotting the difference waveform. The difference error measure $\varepsilon_{\text{Difference}}$ is defined as:

$$\varepsilon_{\text{Difference}} = \max\left\{\frac{y_{i_A} - y_{i_B}}{r} \quad \forall \quad (y_{i_A}, y_{i_B})\right\}$$

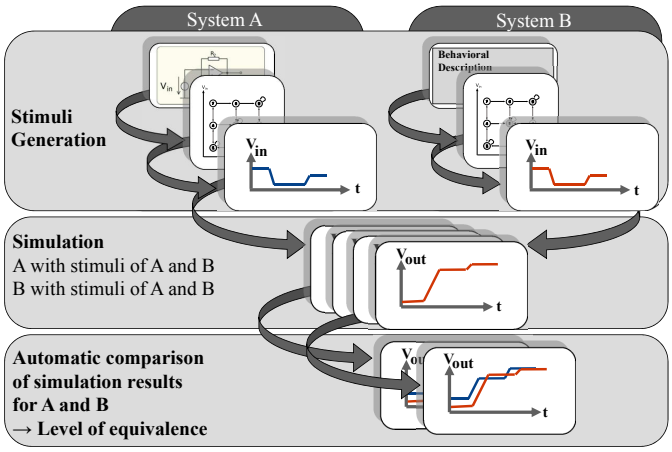

Fig. 4. Equivalence checking flow using complete state space-covering input stimuli.

with:

i	-	index of waveform sample time points
r	-	maximum signal value range
y_{i_A}, y_{i_B}	-	values at time point i for each waveform

The error is normalized with the range of all values to obtain a deviation between 0 and 1. As the time points of two simulation waveforms in general are not equal, an interpolation is necessary for obtaining the error values of the corresponding waveform at an arbitrary time point.

VI. Experimental Results

In this section, the proposed new stimuli based equivalence checking methodology, in the following referred to as stimEC, is applied to example circuits and corresponding behavioral models. The obtained results for each of the circuits are compared to a verification that has been performed using the VERA equivalence checking methodology [1]. The VERA methodology directly compares the dynamic behavior in the state spaces of the systems under verification by mapping the state spaces using a nonlinear transformation to a canonical representation for each system. This is only possible for implementations that do not differ substantially in their internal structure.

A. Biquad Bandpass Filter

The first example circuit considered is a biquad bandpass filter illustrated in Figure 5 with $C_1 = C_2 = 1$ μF, $R_1 = R_2 = R_4 = 10$ kΩ, $R_3 = 30$ kΩ, $R_5 = 20$ kΩ, $V_{DD} = 2.5$ V, $V_{SS} = -2.5$ V, $V_{in} = \pm 0.7$ V. The used op-amp is a simple 8-transistor CMOS design. This transistor netlist representation shall be compared with a VHDL-AMS behavioral model partially illustrated in Listing 1 in order to show that the behavioral model can be used for faster system simulation. Therefore, for the transistor netlist of this circuit, a complete-coverage stimulus containing 8648 time and value pairs with a time length of 300 ms is created and the circuit netlist as well as the behavioral model are simulated with this stimulus. The input stimulus and the simulation results are shown in Figure 6. Using the difference error measure, an error value of 11% is reported. The VERA equivalence checking method reports 1.32% of difference. The higher error reported by the stimEC method is due to the

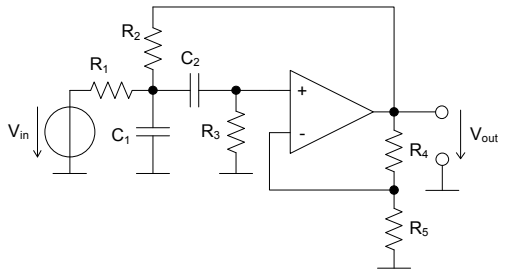

Fig. 5. Circuit schematic of the biquad bandpass filter circuit netlist.

Listing 1. Excerpt from VHDL-AMS behavioral model of the bandpass filter.

```
ENTITY bandpass IS
  GENERIC( den0 : real := ... );
  PORT( terminal inp, outp, gnd : electrical );
end BANDPASS;

ARCHITECTURE behave OF bandpass IS
  QUANTITY uout ACROSS iout THROUGH outp TO gnd;
  ...
BEGIN
  uint == 1E-3*i_uint'dot+4E-7*i_uint;
  uint == 1.0/den1*(-den0*0.001*i_uint
          -den2*uint'dot+num1*uin);
  uout == uint+Rout*iout;
  ...
END behave;
```

differing high-frequency behavior in the beginning of the simulation not equally being captured by the VERA method.

The state space of the biquad bandpass circuit transistor netlist is spanned by the input and the voltages over the capacitors. Hence, a complete coverage of the state space can be proven by plotting the transient response to the input stimulus over V_{C_1} and V_{C_2}. As illustrated in Figure 7, the transient response is covering the reachable states of the circuit.

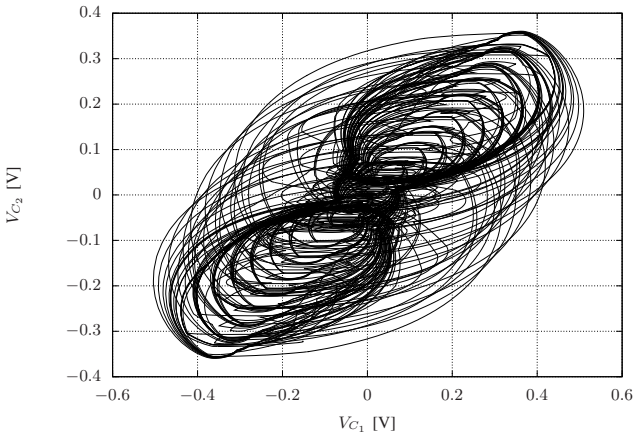

Fig. 7. Transient response to the generated input stimulus of the biquad bandpass filter plotted over V_{C_1} and V_{C_2}.

B. Further Circuit Examples

Table I summarizes the equivalence checking results including runtime information, the number of state space dimensions, and the number of states in the graph structure used for stimuli generation, comparing the new stimEC approach to the VERA approach. In addition to the aforementioned biquad bandpass, three other circuit examples have been processed with stimuli generated for the transistor netlists. The results for a log domain filter, a schmitt trigger and a transistor switch with enable and power down functionality all indicate the feasibility of the new stimEC approach, producing results very similar to the VERA method. Runtimes of the stimEC approach are dominated by the transient simulation runs with the generated stimuli, while the stimuli generation and the error computation on the waveforms only require a small fraction of the overall runtime.

In order to compare the runtime results to the new stimEC approach, approximated runtimes of a conventional systematic simulation have been calculated for the four example circuits. While systematic simulation cannot guarantee the complete coverage of the circuits' behavior, it is a common approach for circuit characterization. For the presented circuit examples, performing such systematic simulations leads to higher simulation times than those of the stimEC approach. Table II summarizes the number of input time steps, the transient response time steps, and the runtimes of the stimEC approach compared to the approximated runtimes of a conventional systematic simulation.

VII. Conclusions

In this contribution, a novel approach to equivalence checking of analog circuits was proposed. The generation of complete state space-covering input stimuli enables transient simulation with guaranteed coverage of the entire reachable state space of the circuits under verification. By automatically comparing the transient responses of two circuit implementations to such stimuli, equivalence checking is possible with guaranteed certainty on the results due to the complete coverage of the circuits' behavior. This contributes to an efficient hierarchical AMS verification flow with analog transistor blocks replaced by equivalence-checked behavioral models.

Unlike other approaches to analog equivalence checking, the possible level of abstraction of behavioral models that can be compared to transistor netlists is not restricted. While the flow automatically computes the level of difference/equality of the input-output behavior of the circuits under verification, the transient simulation output waveforms determined by the input stimuli can easily be inspected by a verification engineer. The application to example circuits has demonstrated the feasibility of the approach.

Future work will include the refinement of the state space sampling strategy in order to further extend the manageable complexity of the analog blocks and to speed up the equivalence checking process.

References

[1] W. Hartong, R. Klausen, and L. Hedrich. Formal Verification for Nonlinear Analog Systems: Approaches to Model and Equivalence Checking. *Advanced Formal Verification, R. Drechsler, ed., Kluwer Academic Publishers, Boston*, pages 205–245, 2004.

[2] T. R. Dastidar and P. P. Chakrabarti. A verification system for transient response of analog circuits. *ACM Trans. Des. Autom. Electron. Syst.*, 12(3):1–39, 2007.

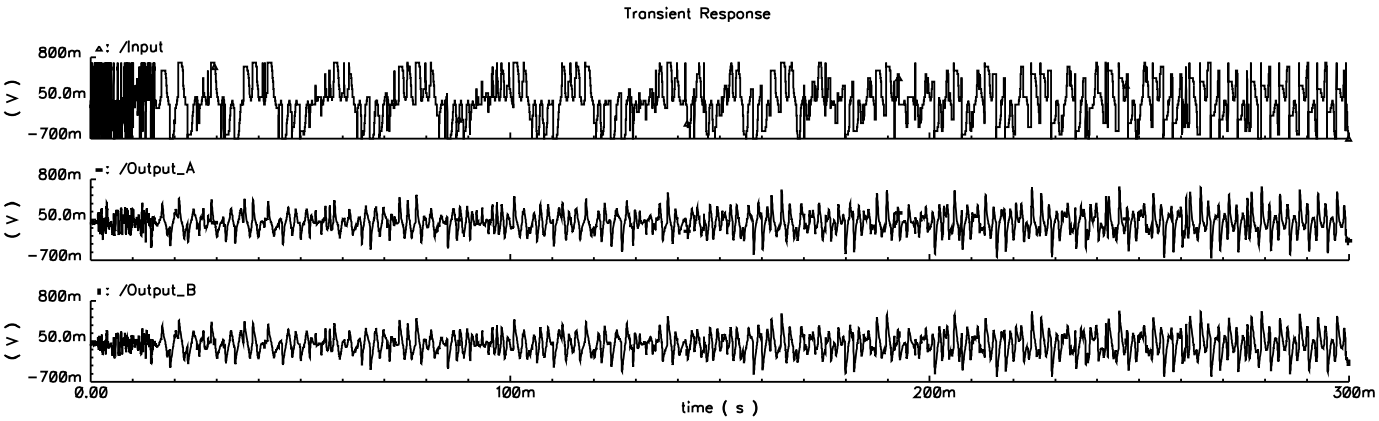

Fig. 6. Complete generated input stimulus and transient output response of the biquad bandpass filter transistor netlist (Output_A) and VHDL-AMS behavioral model (Output_B).

TABLE I

COMPARISON OF THE RESULTS BETWEEN EQUIVALENCE CHECKING BY COMPLETE STATE SPACE-COVERING INPUT STIMULI (STIMEC) AND THE TRANSFORMED STATE SPACE COMPARISON APPROACH (VERA). THE NUMBER OF STATE SPACE DIMENSIONS AND THE NUMBER OF DISCRETE STATES OF THE STATE SPACE GRAPH IS GIVEN FOR THE TRANSISTOR NETLISTS. RUNTIMES ARE COMPUTED ON A SINGLE CORE OF A CORE 2 QUAD WITH 2.83 GHZ AND 8 GB OF RAM.

Circuit (Compared implementation)	**#Dim.**	**#States**	**Error stimEC**	**Runtime**	**Error VERA**	**Runtime**
Biquad Bandpass (Netlist vs. VHDL-AMS)	3	1344	0.07%	16.54s	0.20%	5.97s
Log Domain Filter (Netlist vs. transfer function)	2	13866	5.73%	19.64s	7.26%	9.49s
Schmitt Trigger (Netlist vs. VHDL-AMS)	2	754	3.37%	18.19s	3.51%	5.50s
Transistor Switch (Netlist vs. beh. model)	5	320	0.88%	14.32s	0.61%	8.47s

TABLE II

INPUT TIME STEPS, TRANSIENT SIMULATION RESPONSE TIME STEPS, AND RUNTIME OF THE STIMEC EQUIVALENCE CHECKING APPROACH COMPARED TO THE APPROXIMATED SIMULATION RUNTIMES OF CIRCUIT COMPARISON BY SYSTEMATIC SIMULATION.

Circuit (Compared implementation)	**Inp. Tsteps**	**Sim. Tsteps**	**stimEC**	**Systematic Sim.**
Biquad Bandpass (Netlist vs. VHDL-AMS)	3155	45903	16.54s	$\approx 54s$
Log Domain Filter (Netlist vs. transfer function)	13910	65926	19.64s	$\approx 44s$
Schmitt Trigger (Netlist vs. VHDL-AMS)	4373	61337	18.19s	$\approx 44s$
Transistor Switch (Netlist vs. beh. model)	79	8322	14.32s	$\approx 258s$

[3] D. Nickovic and O. Maler. Amt: A property-based monitoring tool for analog systems. In Jean-François Raskin and P. S. Thiagarajan, editors, *FORMATS*, volume 4763 of *Lecture Notes in Computer Science*, pages 304–319. Springer, 2007.

[4] S. Steinhorst and L. Hedrich. Improving verification coverage of analog circuit blocks by state space-guided transient simulation. In *Circuits and Systems, 2010. ISCAS 2010. IEEE International Symposium on*, May 2010.

[5] H.-G. Ma, X.-F. Zhu, J.-F. Xu, and M.-S. Ai. Circuit state analysis using chaotic signal excitation. *Journal of the Franklin Institute*, 345(1):75 – 86, 2008.

[6] B. Burdiek. Generation of optimum test stimuli for nonlinear analog circuits using nonlinear - programming and time-domain sensitivities. In *DATE '01: Proceedings of the conference on Design, automation and test in Europe*, pages 603–609, 2001.

[7] M. Soma, S. Huynh, J. Zhang, S. Kim, and G. Devarayanadurg. Hierarchical ATPG for Analog Circuits and Systems. *IEEE Des. Test*, 18(1):72–81, 2001.

[8] W. Verhaegen, G. Van der Plas, and G. Gielen. Automated Test Pattern Generation for Analog Integrated Circuits. In *VTS '97: Proceedings of the 15th IEEE VLSI Test Symposium (VTS'97)*, pages 296–301, 1997.

[9] A. Singh and P. Li. On behavioral model equivalence checking for large analog/mixed signal systems. In *Proceedings of the International Conference on Computer-Aided Design*, pages 55–61. IEEE Press, 2010.

[10] M. Horowitz, M. Jeeradit, F. Lau, S. Liao, B.C. Lim, and J. Mao. Fortifying analog models with equivalence checking and coverage analysis. In *Proceedings of the 47th Design Automation Conference*, pages 425–430. ACM, 2010.

[11] W. Hartong, L. Hedrich, and E. Barke. Model checking algorithms for analog verification. In *Proceedings of the 39th conference on Design automation (DAC '02)*, pages 542–547, 2002.

[12] U. Feldmann, U. A. Wever, Q. Zheng, R. Schulz, and H. Wriedt. Algorithms for Modern Circuit Simulation. *Int J Electron Commun*, 46(4):274–285, 1992.

[13] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, USA, 1979.

[14] H. Alt and M. Godau. Measuring the resemblace of polygonal curves. In *Proc. of the 8th Annual Symposium on Computational Geometry*, pages 102–109, 1992.

[15] R. Popp, W. Hartong, L. Hedrich, and E. Barke. Error estimation on symbolic behavioral models of nonlinear analog circuits. In *SMACD '98: Proceedings of the 5th International Conference on Symbolic Methods and Applications to Circuit Design*, pages 223–226, 1998.