Improving Verification Coverage of Analog Circuit Blocks by State Space-Guided Transient Simulation

Sebastian Steinhorst and Lars Hedrich
Electronic Design Methodology Group, Department of Computer Science
University of Frankfurt/Main, Germany
Email: {steinhorst, hedrich}@em.cs.uni-frankfurt.de

***Abstract*—In this contribution a novel methodology for verification of analog circuit blocks with the aim of full coverage of the analog state space is proposed. On a discretized state space model of the analog system, an efficient state space-guided input stimuli generation algorithm produces piecewise linear input stimuli for every input of the system under verification. Processed by a conventional transient circuit simulator, the simulation results are covering the system's complete dynamic behavior. Simulation by complete state space-covering input stimuli guarantees the verification results to be sound for every possible state and input stimulus of the circuit under verification, which increases the significance of property verification and equivalence checking of transistor netlists versus behavioral models. The application to example circuits shows the feasibility of the approach.**

I. Introduction

Due to the increasing system complexity and decreasing time to market, verification of analog circuit designs has become a more and more crucial part of the design flow. While formal verification methods are established in the digital domain, industrial analog circuit design flows are lacking formal or at least formalized verification methodologies. Analog circuit verification still depends on the designer's experience and expertise to manually define appropriate test benches for simulation-based design flows and to select the right input signals in order to detect possible design errors. Driven by the perennial demand for higher design efficiency, new approaches offering more automation of the verification process are of vital importance.

While approaches to assertion-based simulation are emerging, which are mainly automating previously manual efforts, they are not targeting the fundamental problem of analog circuit verification: verification coverage. Todays established common verification methodology is analyzing the circuit's behavior by simulation using test benches. Specification conformance is checked by performing several transient simulations with input signals which are considered representative for the future operating conditions of the circuit. Although this approach to discover design errors has been working for decades, redesigns have occurred frequently due to missing some critical behavior of the circuit during simulation.

In the area of post-production testing of analog circuits, several approaches to automatic test pattern generation (ATPG) have been developed for a systematization of the test procedure. Their common method is to start with a set of given faults and trying to compute a test stimulus that covers every element of the fault set using either sensitivity analysis [1], controllability and observability computation [2], or statistical distance computation [3]. Shifting the perspective to design-time verification, automatized approaches are very rare. An approach [4] of generating constrained randomized stimuli cannot guarantee to cover the complete state space of the design under verification (DUV).

In recent years, academic formal verification techniques for analog circuits have emerged [5], [6], [7], [8]. Formal verification proves the absence of faults in every possible state of a system, regardless of the input conditions. Besides the technical difficulties of these approaches, the need for a complete different way of thinking when dealing with formal verification discourages circuit designers from accepting formal methods in their design flows. While the introduction of designer-oriented methodologies for model checking of analog circuits [9] facilitated the access to a certain degree, gaining acceptance will be a long time coming. Consequently, there is a need for formal approaches that seamlessly integrate in existent simulation-based design flows.

The purpose of this contribution is to propose a property verification and equivalence checking methodology for analog circuit blocks based on a novel algorithm for formal automatic input stimuli generation. Therewith, it overcomes the incompleteness of transient simulation and the designer-unfriendliness of formal approaches by combining a formal approach and conventional transient circuit simulation.

This paper is organized as follows. In section II, the discrete state space modeling will be introduced which is necessary for application of the input stimuli generation algorithm described in section III. Enabled by the possibility of simulation with guaranteed complete state space coverage, the new approach is used for verification of an example circuit in section IV. Moreover, a second order delta-sigma modulator is equivalence-checked against a behavioral model using the stimuli generation approach for developing an analog equivalence checking methodology in section V. Section VI 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 [8]. 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 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. 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 [10], circuits with more than 200 parasitic capacitances and full BSIM3 transistor models can be handled. This is achieved by reducing the state space to the dominant state variables of a system and separating the parasitic ones which are mathematically proven not to affect the system behavior above a defined threshold.

This work was partly developed within the project VeronA (project label 01 M 3079) which is funded within the Research Program ICT 2020 by the German Federal Ministry of Education and Research (BMBF).

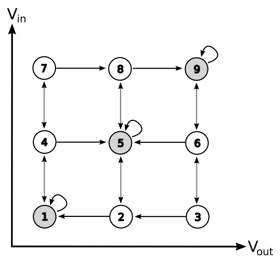

Fig. 1. Schematic illustration of a graph structure created by the discrete modeling process.

Figure 1 illustrates a schematic 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 corresponding direction. Any non-steady state of the system has outgoing directed edges representing the dynamic behavior of the circuit.

The described modeling process is implemented as an extension to the Infineon Titan Simulator [11] and the analog modeling and verification tool amcheck [8].

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. Consequently, 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 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 [12]. 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 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 $i \rightarrow j \rightarrow i$ is computed avoiding edges in the set $closed$ in line 4. Line 5 iterates over each edge k visited on the computed path $i \rightarrow j \rightarrow i$ 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.

Algorithm 1: Stimuli Generation Algorithm

```
Input: vertex $i$ 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 directed edges;
2  closed = $\emptyset$;
3  foreach edge $j$ reachable from $i$ do
4      calculate path covering as many edges as possible from
       $i \rightarrow j \rightarrow i$ avoiding edges in closed;
5      foreach edge $k$ visited on path $i \rightarrow j \rightarrow i$ do
6          put $k$ to closed;
7          remove $k$ from open;
8          store parameter vector of visited vertices and path time;
9      end
10 end
```

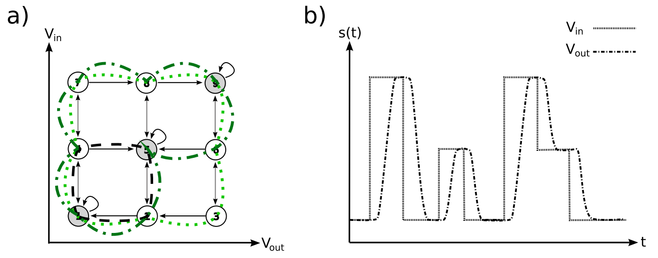

Fig. 2. Path generated by the stimuli generation algorithm (a) and the corresponding input/output behavior (b).

The implicitly mentioned path finding algorithm in line 4 can be any form of a longest path finding algorithm such as Dijkstra's 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.

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. Application to Property Verification

The new methodology was applied to a Sallen-Key biquad low-pass filter with a cut-off frequency of 1000 Hz. The circuit schematic

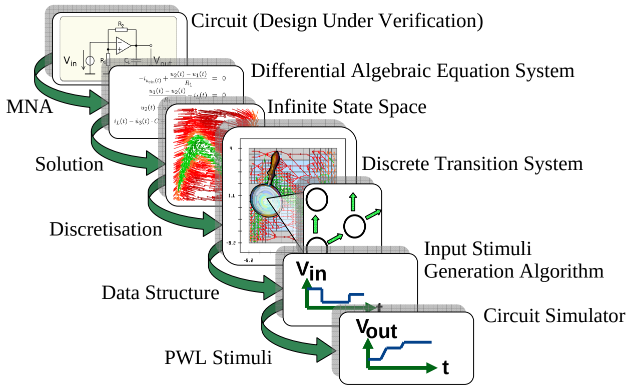
Circuit (Design Under Verification)
Differential Algebraic Equation System
Infinite State Space
Discrete Transition System
Input Stimuli Generation Algorithm
Circuit Simulator
MNA
Solution
Discretisation
Data Structure
PWL Stimuli
Vin
Vout
t

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

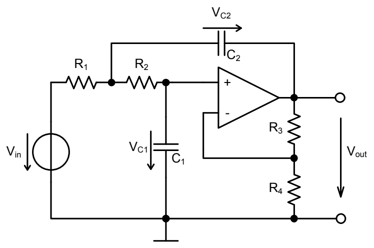

Fig. 4. Circuit schematic of the Sallen-Key biquad low-pass filter.

is illustrated in figure 4 with $C_1 = 5\,\text{nF}$, $C_2 = 50\,\text{nF}$, $R_1 = R_2 = R_4 = 10\,\text{k}\Omega$, $R_3 = 500\,\Omega$. The input voltage range is $\pm 1\,\text{V}$ and the operational amplifier has a supply voltage of $\pm 2\,\text{V}$.

The discrete model generation of the graph structure containing 2086 vertices of which 987 are reachable from DC operating points took 12 seconds on a single core of a Core 2 Quad with 2.83 GHz and 8 GB of RAM. The generation of the input stimulus with 9191 time and input value pairs with an overall time length of $758\,\text{ms}$ took 7 seconds.

Applied to the Sallen-Key biquad low-pass filter in an analog circuit simulator, a detail of the complete transient response is plotted in figure 5 showing the input stimulus and circuit response between $64\,\text{ms}$ and $80\,\text{ms}$. The circuit tends to overshoot around the cut-off frequency, which is clearly visible in the transient response to the input stimulus but not necessarily detected by transient analysis with user-defined signals. Moreover, using a property specification for overshoot behavior and an automatic evaluation on the simulation result, complete and therefore formal property verification coverage could be obtained without user-interaction.

While the input stimulus generation algorithm is designed to com-

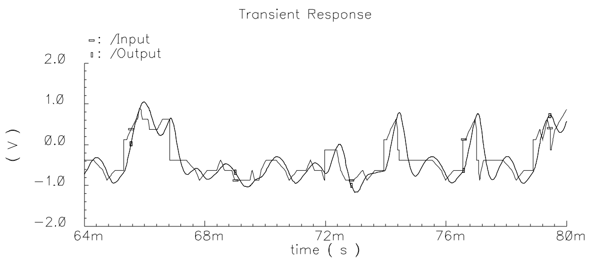

Fig. 5. Detail of complete generated input stimulus and transient output response of the circuit schematic of the Sallen-Key biquad low-pass filter between $64\,\text{ms}$ and $80\,\text{ms}$.

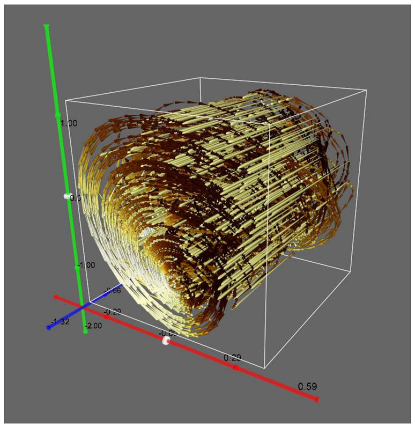

Fig. 6. Transient response to the generated input stimulus of the Sallen-Key biquad low-pass filter plotted over V_{in} (blue axis), V_{C_1} (green axis), and V_{C_2} (red axis) proving complete coverage of the reachable area.

pletely cover the modeled circuit, which can be easily verified, it is necessary to prove that the generated stimulus really forces the circuit to adopt every reachable state of the system during transient simulation. In order to prove complete coverage of the system variables, in figure 6 the transient circuit response to the input stimulus is plotted over V_{in}, V_{C_1}, and V_{C_2}, clearly showing that the trajectories completely cover the reachable state space.

V. Application to Equivalence Checking

Based on the new stimuli generation approach, another application is to develop an equivalence checking methodology for analog circuits, 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 a difference error measure, 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:

$$A \subseteq B \tag{1}$$

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:

$$B \subseteq A \tag{2}$$

If both conditions (1) and (2) hold, circuit A and circuit B are considered as equivalent with respect to the user-defined maximum error:

$$A \subseteq B \quad \wedge \quad B \subseteq A \quad \Longrightarrow \quad A \approx B \tag{3}$$

In practical applications, often only one direction of the proof is necessary. Especially for reduced models generated with model order reduction techniques, the full-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

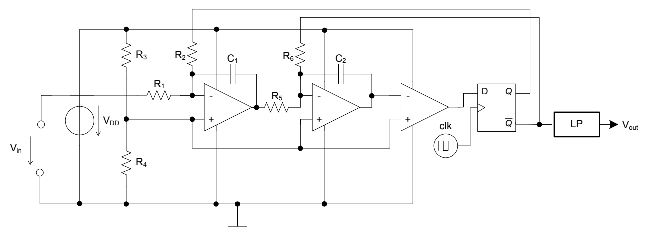

Fig. 7. Circuit schematic of the second order delta sigma modulator.

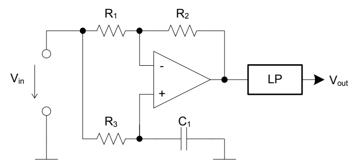

Fig. 8. Simple behavioral model for the second order delta sigma modulator using an all-pass filter for signal delay modeling.

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.

The equivalence checking methodology is now applied to a second order delta sigma modulator. Due to the high clock frequencies of delta sigma modulators, they require plenty of simulation time. Hence, for faster simulation of mixed signal systems containing delta sigma modulators, behavioral models are used. With the new stimuli generation approach, the comparison of a transistor netlist implementation of a second order delta sigma modulator versus a simple unclocked behavioral model using an all-pass filter as a delay component is performed.

Figure 7 shows the transistor netlist implementation of the second order delta sigma modulator with $C_1 = C_2 = 200\,\text{pF}$, $R_1 = R_2 = R_5 = R_6 = 100\,\text{k}\Omega$, $R_3 = R_4 = 5\,\text{k}\Omega$, $V_{DD} = 1\,\text{V}$, $f_{clk} = 1\,\text{MHz}$. The sequential bitstream is directed into a low-pass filter for further processing. A simple behavioral model for this circuit is implemented by an all-pass delay circuit as illustrated in Figure 8 with $C_1 = 10\,\text{nF}$, $R_1 = R_2 = R_3 = 1\,\text{k}\Omega$.

The complete state space-covering input stimulus generated for the delta sigma modulator contains 19163 time and value pairs. After simulation of both circuit implementations with the stimulus, the difference error measure is applied to the the output voltage waveforms, reporting an output error of 10.32% when excluding the startup time of $0.2\,\text{ms}$. For proving the complete coverage of the reachable state space of the internal capacitors C_1 and C_2, the transient response to the input stimulus is plotted over V_{C_1} and V_{C_2} as depicted in Figure 9. The trapezoidal appearance is caused by the correlation of the inverting integrators, restricting the reachable value combinations of V_{C_1} and V_{C_2}. The complete verification process took 145 seconds on a single core of a Core 2 Quad with 2.83 GHz and 8 GB of RAM.

VI. Conclusions

In this contribution, a novel approach to verification 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. This can be applied to conventional test benches to increase verification confidence. Moreover, 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. The transient simulation output waveforms determined by the input stimuli can easily be inspected by a verification engineer, hence, although offering an automated approach for increasing verification coverage, the process is transparent for the designer.

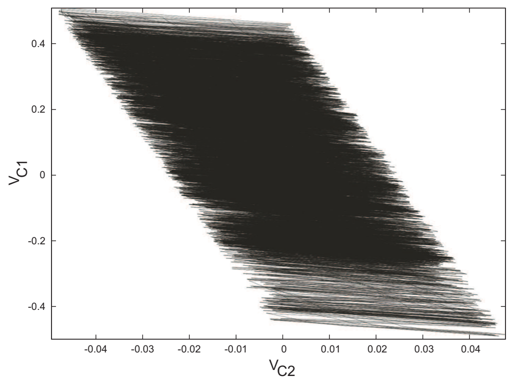

Fig. 9. Transient response to the generated input stimulus of the delta sigma modulator transistor netlist plotted over V_{C_1} and V_{C_2}, proving the complete coverage of the reachable area of this critical state space dimensions.

The application to example circuits has demonstrated the feasibility of the approach and future work will include the refinement of the state space sampling strategy in order to further extend the manageable complexity of the analog blocks.

VII. Acknowledgments

The authors would like to thank Peter Jores (AE/EIM3, Robert Bosch GmbH, Reutlingen, Germany) as well as Georg Denk and Carsten Hammer (Titan Simulator Development, Infineon AG, Munich, Germany) for very fruitful discussions and their helpful suggestions to this work.

References

[1] 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.

[2] Mani Soma, Sam Huynh, Jinyan Zhang, Seongwon Kim, and Giri Devarayanadurg. Hierarchical ATPG for Analog Circuits and Systems. *IEEE Des. Test*, 18(1):72–81, 2001.

[3] 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.

[4] G. Bonfini, M. Chiavacci, R. Mariani, and E. Pescari. A mixed-signal verification kit for verification of analogue-digital circuits. In *DATE '06: Proceedings of the conference on Design, automation and test in Europe*, pages 88–93, 2006.

[5] Erich Barke, Darius Grabowski, Helmut Graeb, Lars Hedrich, Stefan Heinen, Ralf Popp, Sebastian Steinhorst, and Yifan Wang. Formal approaches to analog circuit verification. In *DATE*, pages 724–729. IEEE, 2009.

[6] Mohamed H. Zaki, Sofiène Tahar, and Guy Bois. Formal verification of analog and mixed signal designs: A survey. *Microelectronics Journal*, 39(12):1395 – 1404, 2008.

[7] Tathagato Rai 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.

[8] 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.

[9] S. Steinhorst and L. Hedrich. Model Checking of Analog Systems using an Analog Specification Language. In *Proc. of the Conference on Design, Automation and Test in Europe 2008 (DATE'08)*, pages 3247–329, 2008.

[10] 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.

[11] 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.

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