Equivalence Checking of System-Level and SPICE-Level Models of Linear Analog Filters

Kemal Çağlar Coşkun* Muhammad Hassan† Rolf Drechsler*†
*University of Bremen, Institute of Computer Science, 28359 Bremen, Germany
†Cyber-Physical Systems, DFKI GmbH, 28359 Bremen, Germany
muhammad.hassan@dfki.de {kcoskun,drechsle}@informatik.uni-bremen.de

***Abstract*—Due to the increasing complexity of analog circuits and their integration into *System-on-Chips* (SoC), the analog design and verification industry would greatly benefit from an expansion of system-level methodologies, which provide speed benefits in comparison to SPICE simulations and allow interoperability with digital tools at the system-level. However, a key barrier to the expansion of system-level tools for analog circuits is the lack of confidence in system-level models implemented in SystemC AMS. To overcome this, functional equivalence of system-level models to respective SPICE-level models needs to be demonstrated. In this paper, we develop a novel, graph-based methodology to formally check equivalence between system-level and SPICE-level representations of linear analog filter circuits, such as *Low-Pass Filters* (LPF). To do this, we propose an intermediate representation in the form of a *Signal-flow Graph* (SFG), which acts as a mapping function from the SPICE-level to the system-level. We create the intermediate representation with linear graph modeling from the SPICE-level model and use graph manipulation to transform the intermediate representation to the equivalent system-level model. We demonstrate the applicability of the proposed methodology by successfully applying it to two example filters.**

***Index Terms*—Equivalence checking, formal verification, linear circuits, filters, circuit analysis, transfer functions**

I. Introduction

The rising complexity of analog circuits and the ever increasing system integration of analog and digital components have created a bottleneck for analog design verification. A major challenge in this regard is the simulation speed of SPICE-level models [1]. They often fail for large systems due to convergence related problems or are prohibitive in terms of computational time required. Traditionally, SPICE-level simulations [2] are used often with manual inspection of the results. These simulations, while slow, are still considered a golden standard and cannot be ignored. However, different levels of design abstractions and alternate representations (e.g., a behavioral model) of the circuit can be used to achieve significantly better simulation performance and earlier design verification of the Design Under Verification (DUV).

As a consequence, analog designs are moving towards a top-down approach. In this regard, *Virtual Prototyping* (VP) at the abstraction of *Electronic System Level* (ESL) is nowadays an established practice [1], [3], [4], [5], [6], [7], [8], [9], [10], [11]. The *Timed Data Flow* (TDF) *Model of Computation* (MoC) available in SystemC AMS offers a good trade-off between accuracy and simulation-speed at the system-level, and can provide a speed increase of over $100,000\times$ [1] in comparison to SPICE-level simulations. TDF defines time domain processing, and is used to model the pure algorithmic or procedural description of the underlying design. In particular, TDF provides utilities to implement *Laplace Transfer Functions* (LTF)[1] of linear systems. Because of earlier availability and significantly faster simulation speed as opposed to SPICE-level simulations [1], the TDF models provide a design refinement methodology and enable early verification for analog/mixed-signal systems.

However, one of the main challenges in adopting SystemC AMS system-level models is the lack of equivalence checking methodologies for SystemC AMS and SPICE-level models. Equivalence checking proves the general functional equality of two implementations of a design. The implementations can be of different abstraction levels and different description methods such as transistor netlists and system-level languages. While equivalence checking methods are well established in the digital domain [12], [13], [14], analog circuit design flows are lacking formal or at least formalized verification methodologies [15], [16], [17]. When speaking about equivalence checking methodologies, we broadly consider approaches like state-space coverage, model-checking, and reachability. Regardless of the specific approach, confidence in adopting SystemC AMS system-level analog models is low. As a consequence, completely relying on SystemC AMS system-level models gets difficult. Due to the rising complexity of analog designs, this becomes a serious problem.

Contribution: In this paper, we propose a novel equivalence checking methodology, which is to the best of our knowledge the first of its kind. Essentially, our approach operates directly on the SPICE-level models by combining several transformations of linear graph modeling techniques. The main challenge is to show that the SPICE-level model is equivalent to the system-level LTF functional model implemented in SystemC AMS. In particular, the developed method is restricted to the class of linear analog filters, e.g., *Low-Pass Filters* (LPF), *High-Pass Filters* (HPF), etc. We leverage *Signal-flow Graphs* (SFG) as

This work was supported in part by the German Federal Ministry of Education and Research (BMBF) within the project AUTOASSERT under contract no. 16ME0117.

[1]A transfer function model captures the frequency response of an analog circuit and provides a suitable platform for applying non-simulation / formal techniques to verify the circuit against its specification.

978-1-6654-9431-1/22/$31.00 ©2022 IEEE

an intermediate representation between the SPICE-level and the system-level model. We demonstrate the applicability of the developed methodology by successfully applying it to complex filters.

Summarizing the main contributions of this paper:

- Novel equivalence checking methodology for system-level and SPICE-level models
- Leverages SFGs and linear graph modeling techniques
- Applicable to the complete class of linear analog filters irrespective of their order and complexity
- Demonstration of equivalence checking on complex filter models

II. Related Work

In their survey of equivalence checking, Zaki *et al.* [16] summarized the literature until 2007 and pointed out that all the presented methods employ a priori knowledge of the DUV in the development process. A further comparison of some equivalence checking methods was presented in [17] by Tarraf *et al.* along with the proposal of a new equivalence checking method based on reachability. It is observed in this work that the definition of the coverage measures is a difficult task and that many methods balance completeness against pessimism.

In their study of equivalence checking on the state-space, Hedrich and Barke [15] compared the vector fields of the systems on a point grid to check the equivalency of two different representations of circuits. The method is applicable to single-input single-output circuits that can be described by a set of nonlinear time-invariant first-order differential equations. [18] extended the method in [15] to circuits that are defined by differential-algebraic equations, [19] applied the method to new examples, and [20] generalized it to multi-input multi-output circuits. The equivalence checking method proposed in [15] is applicable to many circuits, but some important dynamics might be missed since the points on the grid are fixed distances apart on the canonical state space.

In an investigation into simulation-based equivalency checking, Singh and Li [21] developed mapping techniques for comparing signals in different domains and decreased the high computational burden of simulation-based approaches by developing techniques that reduce the input space. However, the method relies on typical system-level simulation stimuli, which cannot completely cover all behavior, and the authors highlight this point by calling their method semi-formal. Ain *et al.* [22] also worked on simulation-based equivalency and developed a systematic methodology with a focus on circuit features. However, no attempt was made to mitigate the possible incompleteness of the externally given test bench. The coverage issue of simulation-based verification was addressed by Saglamdemir *et al.* [23] through an optimization-based method for automatic generation of inputs. Unfortunately, a discussion on whether all essential input shapes can be represented with the given set of input parameters is missing. Another problem that was not addressed is the possibility of the optimization to return a local minimum.

In summary, a common deficiency of many methods in the existing literature is that they do not check equivalence with complete coverage of behavior. Therefore, even if these methods claim equivalence, the models might still behave differently in an overlooked gap. Our proposed equivalence checking methodology does not have this issue as the analysis and modification methods used, such as linear graph modeling and graph reduction, statically analyze the structure of the models.

III. Preliminaries

In this section, we present a brief summary of SFGs and introduce our motivating example that led to the development of our method. Please note, for brevity we refrain from giving a proper introduction to SystemC AMS, and encourage the reader to go through the SystemC AMS user guide [24].

A. Signal-Flow Graph

Consider the system of explicit algebraic equations shown in Eq. 1:

$$\bar{x} = f(\bar{x}) \tag{1}$$

where $\bar{x}$ is an array of variables. An SFG as introduced in [25] is a representation of Eq. 1 in the form of a graph. But to simplify the algebra [26], it is common to restrict the SFG to a linear form that represents a system of linear explicit algebraic equations written as $\bar{x} = \mathbf{A}\bar{x}$ when arranged in matrix form. However, it is easier to construct the SFG from the open form (Eq. 2) with every variable (x_i) given as a sum of all variables including itself (x_j) scaled by some constant a_{ji}.

$$x_i = \sum_j a_{ji} x_j \tag{2}$$

Since we consider the class of linear filter circuits in this work, the linear SFG is sufficient for our purposes. Therefore, we restrict our SFGs to represent equations of the form Eq. 2, where x_i, x_j, and a_{ji} depend on the Laplace variable s.

Fig. 1 shows an example SFG with its equivalent system of linear explicit algebraic equations given in Eq. 3. The edges of the SFG represent the summation terms in the equations and the values of the nodes in the SFG are equal to the sum of the incoming edges. For example, the edge going from x_3 to x_1 with weight a_{31} represents the summation term $a_{31}x_3$ in the explicit equation of x_1 in Eq. 3.

$$x_1 = a_{21}x_2 + a_{31}x_3\,, \quad x_2 = a_{12}x_1\,, \quad x_3 = a_{23}x_2 \tag{3}$$

B. Motivating Example: Fifth-order LPF

As our motivating example for equivalence checking, we consider a single-input ($V1$) single-output (V_o) analog fifth-order passive LPF (Fig. 2) typically used in *Radio Frequency* (RF) receivers. LPF is designed to allow signals with a frequency lower than a certain cutoff frequency, and attenuate the signals

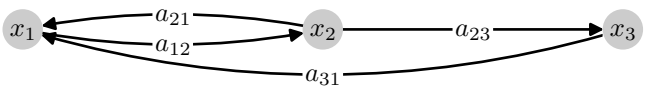

Fig. 1. An example SFG

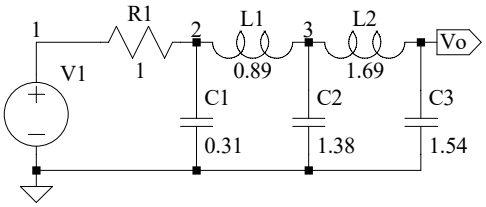

Fig. 2. Motivating example: Analog fifth-order low-pass filter with components *Resistor* (R), *Capacitor* (C), and *Inductor* (L).

with a frequency higher than that cutoff frequency. The LPF circuit is taken from [27] with the following specifications:

- Low-frequency gain: 1
- Bandwidth: 1.0019 rad/s
- Cutoff frequency: 0.1595 Hz

The circuit is implemented in LTSpice [28] and exported as a netlist. The system-level model of the LPF is implemented in SystemC AMS TDF MoC, where its LTF (described by the numerator and denominator coefficients) is shown in Eq. 4.

$$\frac{1.009}{s^5 + 3.226s^4 + 5.252s^3 + 5.249s^2 + 3.26s + 1.009} \quad (4)$$

The coefficients are determined using modified nodal analysis [29]. SystemC AMS provides a dedicated solver for continuous time linear transfer functions in the Laplace domain under the class *sca_tdf::sca_ltf_nd*.

IV. Signal-Flow Driven Equivalence Checking Methodology

In this section, we propose an SFG-based equivalence checking methodology for system-level and SPICE-level linear analog filter models. First, we describe the overview of our proposed methodology, followed by techniques to create and optimize an SFG. In the end, we illustrate our methodology using our motivating example from Section III-B.

A. Methodology Overview

A block-diagram overview of our methodology for equivalence checking between system-level and SPICE-level models is seen in Fig. 3. To generate a complete set of equations from the netlist, we use the linear graph modeling method [30], which consists of a normal tree generator and an equation generator. We chose this method of analysis since it preserves the structure of the circuit the best, loses the least amount of information, and is also applicable to circuits with one- and two-port nonlinear elements [30]. We then create an SFG of the circuit with our SFG creator and reduce it to a minimal form with our SFG simplifier. The simplification methods of the SFG simplifier are detailed in Section IV-C and consist of removal of a non-input node, parallel edge elimination, and reflexive edge elimination. Our equivalence checker compares this minimal SFG to the system-level model of the circuit.

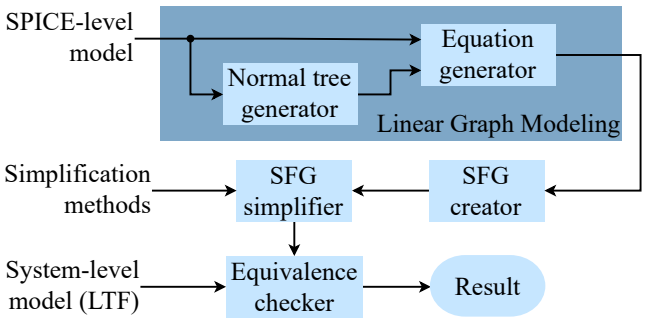

Fig. 3. Overview of the proposed equivalence checking methodology

B. Creating the Signal-Flow Graph with Linear Graph Modeling

For the creation of an SFG, a set of linear explicit algebraic equations in the form of Eq. 2 must be obtained from the SPICE-level model. There are various circuit analysis methods that can be used for this purpose, which use different methods and different sets of representative variables. In principle, various possible SFGs exist for a single circuit since SFGs are created from sets of equations that were generated by one of these circuit analysis methods.

The linear graph modeling method uses the voltages on and currents through the circuit components as representative variables of the circuit. This is in contrast to nodal-analysis and loop-analysis, where the circuit equations are in terms of node voltages and loop currents, respectively.

The linear graph model determines how the circuit variables relate to each other by informing whether to get the explicit equations of a variable through elemental, compatibility (Kirchhoff's voltage law), or continuity (Kirchhoff's current law) equations. The first step of the linear graph modeling method is to create a normal tree, which is a special type of minimum spanning tree of the circuit graph. This is done by the normal tree generator by repetitively adding the edges of the circuit graph in the following order: Voltage sources, capacitors, resistors, inductors, and current sources.

The normal tree must include voltage sources and may not include current sources. Inductors in the normal tree and capacitors in the tree links are dependent energy storage elements. Edges of the circuit graph that are not included in the normal tree, are called the tree links of the normal tree.

For all unknown variables, an explicit expression is generated by the equation generator according to the following rules:

- Voltages of components on the normal tree, from elemental equations.
- Currents of components on the normal tree, from continuity equations.
- Voltages of components on the tree links, from compatibility equations.
- Currents of components on the tree links, from elemental equations.

This method generates equations in the form of Eq. 2, which are used to construct an SFG as explained in Section III-A.

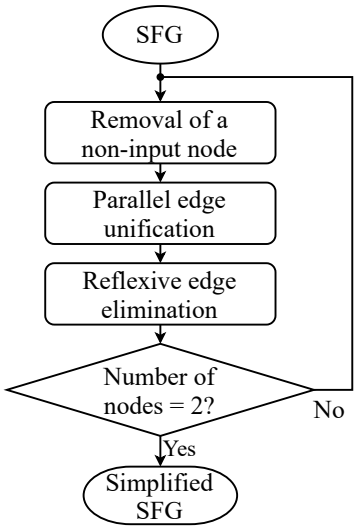

Fig. 4. Overview of SFG simplification process

C. Reducing the Signal-Flow Graph

The overview of the SFG simplification process is given in Fig. 4. Simplification rules [31] given below are applied by the SFG simplifier repeatedly in the given order until a minimal graph with only a single edge between its input and output nodes is obtained.

a) Removal of a non-input node: Input nodes in an SFG are nodes whose values can be set arbitrarily. The voltages of voltage sources and currents of current sources are examples of input nodes. A non-input node n_x may be removed after creating edges from every ancestor of n_x to every descendant of n_x. The weights of these new edges are such that, for a new edge (a_x, d_x), its weight is

$$w((a_x, d_x)) = w((a_x, n_x)) \cdot w((n_x, d_x))$$

where a_x is an ancestor node of n_x and d_x is a descendant node of n_x.

b) Parallel edge unification: Parallel edges are edges whose source and destination nodes are equal. According to the distributive law for parallel edges, these can be merged into a single edge by summing their weights.

c) Reflexive edge elimination: Reflexive edges are edges of a node that point to itself. A reflexive edge with weight w can be removed by dividing the weight of every incoming edge to its node by $1 - w$.

D. Illustration

In this section, we illustrate our methodology on the LPF model from Section III-B. As the first step of linear graph modeling, we obtain the circuit's normal tree in Fig. 5. Comparing this with the circuit in Fig. 2, it is seen that it is indeed a minimum spanning tree by observing that all nodes of the circuit are present in the tree without forming any loops. It can also be seen that the priority order as explained in Section IV-B was followed since all voltages and capacitors of the circuit are present on the tree.

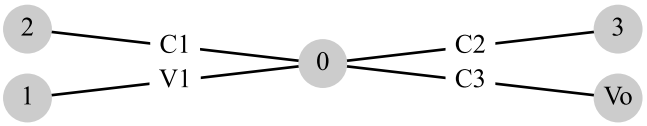

Fig. 5. Normal tree of the low-pass filter

In the second step, we use this normal tree and the rules given in Section IV-B to get the explicit equations; Eq. 5 and Eq. 6, which are required for building the SFG. In Eq. 5, the equations for the components on the normal tree are given.

$$\begin{aligned} V_{C1} &= \frac{1}{0.31s} I_{C1}\,, \quad I_{C1} = I_{R1} - I_{L1}\,, \\ V_{C2} &= \frac{1}{1.38s} I_{C2}\,, \quad I_{C2} = I_{L1} - I_{L2}\,, \\ V_{C3} &= \frac{1}{1.54s} I_{C3}\,, \quad I_{C3} = I_{L2}\,, \\ & \qquad\qquad\quad I_{V1} = -I_{R1}\,. \end{aligned} \tag{5}$$

Whereas the equations for the components on the tree links are given in Eq. 6.

$$\begin{aligned} V_{R1} &= V1 - V_{C1}\,, \quad I_{R1} = \frac{1}{1} V_{R1}\,, \\ V_{L1} &= V_{C1} - V_{C2}\,, \quad I_{L1} = \frac{1}{0.89s} V_{L1}\,, \\ V_{L2} &= V_{C2} - V_{C3}\,, \quad I_{L2} = \frac{1}{1.69s} V_{L2}\,. \end{aligned} \tag{6}$$

While the linear graph modeling approach uses substitution from this point on, we leave the equations as they are, and construct an SFG right away to preserve the structure of the circuit. The equations 5 and 6 are in the form of Eq. 2, from which the SFG given in Fig. 6 can be created.

This SFG is then transformed according to the rules given in Section IV-C. The first step of the transformation, which resulted in the graph in Fig. 7a, is an example of a non-input node removal, where node V_{C1} was removed. An example of reflexive edge elimination is seen in the 11th step of the simplification process, where the reflexive edge from I_{L1} to I_{L1} in Fig. 7b is removed to get the graph in Fig. 7c. The parallel edge unification rule is not needed in this case since no parallel edges were formed during the simplification process. After the successive removal of 12 nodes according to these rules, the minimal SFG in Fig. 8 is obtained. For equivalence checking, it is observed that the SFG has equal input-output behavior to the system-level LTF seen in Eq. 4. The numbers in these figures were printed with reduced floating-point precision due to space considerations.

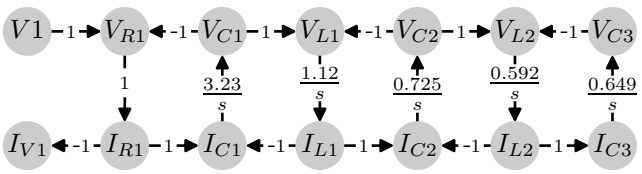

Fig. 6. SFG of the low-pass filter

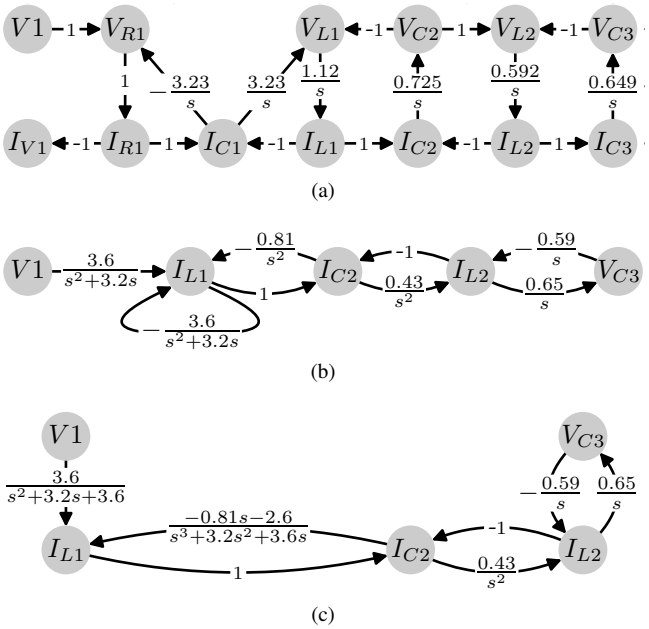

Fig. 7. Some intermediate results of the simplification process for the low-pass filter

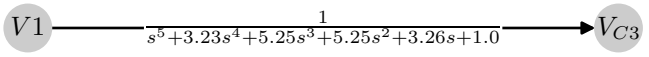

Fig. 8. Reduced SFG of the low-pass filter

V. Experimental Evaluation

In this section, we demonstrate the general applicability of our proposed equivalence checking methodology by applying it on an example circuit that has a non-planar SFG, which is a graph that cannot be drawn in a plane without crossing any edges. First, the experimental setup is briefly discussed. Later, we demonstrate our methodology described in Section IV on the circuit by creating and simplifying an SFG for it. We then show that the simplified SFG is equal to the system-level representation of the circuit.

A. Experimental Setup

The filter used for this demonstration is the single-input (V_S) single-output (V_{RL}) analog third-order passive band-stop filter seen in Fig. 9. The component values of the filter are chosen such that it is of type Butterworth, which has a maximally flat response on the passband. The center frequency and frequency difference of the filter are both 1 rad/s (0.1592 Hz) and the gain at low and high frequencies is 0.5. Therefore, the filter blocks signals with frequencies between 0.0984 Hz and 0.2575 Hz but allows signals with frequencies outside this band.

The SPICE-level model of the filter is a netlist description, whereas the system-level LTF model is implemented in SystemC AMS TDF MoC. The SystemC AMS LTF model as shown in Eq. 7 was found with the modified nodal analysis method.

$$\frac{0.5s^6 + 1.5s^4 + 1.5s^2 + 0.5}{s^6 + 2s^5 + 5s^4 + 5s^3 + 5s^2 + 2s + 1} \tag{7}$$

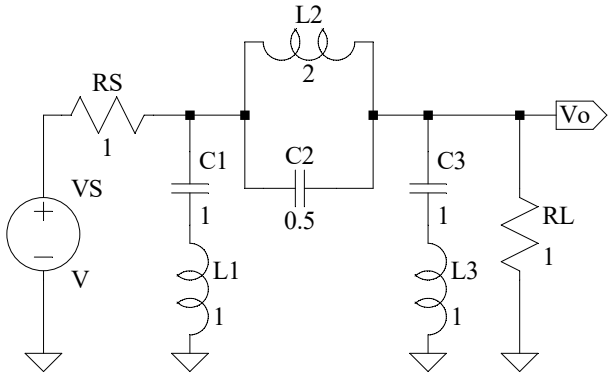

Fig. 9. Butterworth band-stop filter example

B. Equivalence Checking

The process used to create the SFG once again consists of obtaining the normal tree, finding the explicit equations 8 and 9 for the variables of the circuit, and putting these equations together to create the SFG. The normal tree seen in Fig. 10 includes all voltages and capacitors. The inclusion of RS in the normal tree instead of RL is arbitrary.

The explicit equations are found as

$$\begin{aligned} V_{C1} &= \frac{1}{s} I_{C1}\,, & I_{C1} &= I_{L1}\,, \\ V_{C2} &= \frac{1}{0.5s} I_{C2}\,, & I_{C2} &= I_{L3} + I_{RL} - I_{L2}\,, \\ V_{C3} &= \frac{1}{s} I_{C3}\,, & I_{C3} &= I_{L3}\,, \\ V_{RS} &= I_{RS}\,, & I_{RS} &= I_{L1} + I_{L3} + I_{RL} \end{aligned} \tag{8}$$

for the components on the normal tree, and

$$\begin{aligned} V_{L1} &= V_S - V_{RS} - V_{C1}\,, & I_{L1} &= \frac{1}{s} V_{L1}\,, \\ V_{L2} &= V_{C2}\,, & I_{L2} &= \frac{1}{2s} V_{L2}\,, \\ V_{L3} &= V_S - V_{RS} - V_{C2} - V_{C3}\,, & I_{L3} &= \frac{1}{s} V_{L3}\,, \\ V_{RL} &= V_S - V_{RS} - V_{C2}\,, & I_{RL} &= V_{RL} \end{aligned} \tag{9}$$

for the other components. These equations are in the form of Eq. 2 and are used to construct the SFG seen in Fig. 11.

The graph reduction rules given in Section IV-C are then applied to this SFG for simplification. An intermediate graph with 6 nodes is seen in Fig. 12 whereas the final simplified SFG is in Fig. 13.

Comparing the final SFG with the system-level LTF given in Eq. 7, it is observed that both models are equivalent. Therefore, any result generated with the system-level model can be analyzed more confidently.

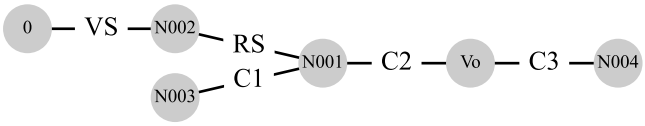

Fig. 10. Normal tree of the Butterworth band-stop filter from Fig. 9

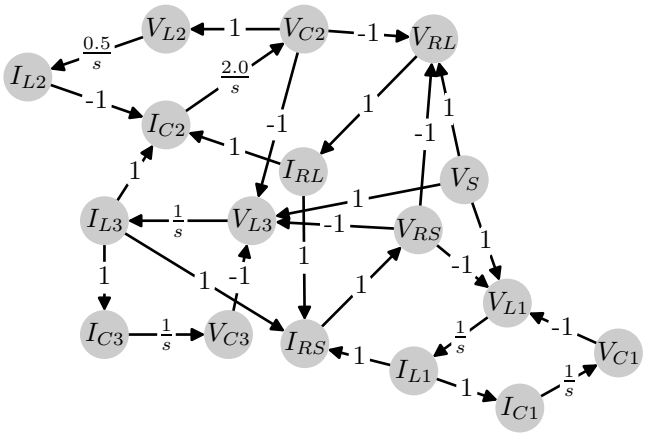

Fig. 11. SFG of the Butterworth band-stop filter

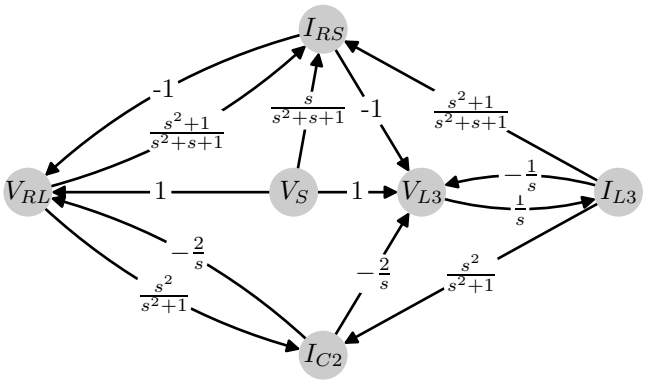

Fig. 12. An intermediate result of the simplification process for the Butterworth band-stop filter

VI. Conclusion

In this work, we combined various analysis and modification techniques in a novel way to create a graph-based, formal equivalence checking method. We used linear graph modeling on a SPICE-level model to create a behavioral SFG and we used graph reduction techniques to compare this graph to an LTF based system-level model implemented in SystemC AMS. By observing the successful application of the methodology to the provided examples, we learned that using graphs to show formal equivalency is a viable option that merits further investigation.

The methods presented in this paper can be extended in multiple ways. Since a slight difference between the SPICE-level and system-level models might be tolerable, the method can be modified to generate an error value between the models. For this, the coefficients of the final transfer function can be compared. Also, the current application scope of this work is restricted to linear filter circuits. A generalization to linear circuits and nonlinear circuits should be investigated. On the system-level side, the methods can be extended to models that are richer than simple LTF models. Another interesting research direction is to leverage the graph-based representation by using established search methods to map possible bugs between the two models.

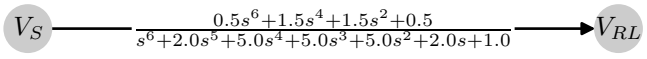

Fig. 13. Reduced SFG of the Butterworth band-stop filter

References

[1] M. Barnasconi, "SystemC AMS Extensions: Solving the Need for Speed," *DAC Knowledge center*, May 2010.

[2] L. W. Nagel, "Spice-simulation program with integrated circuit emphasis," *Electronics Research Laboratory, Univ. of California, Berkeley*, 1973.

[3] C. Grimm, M. Barnasconi, A. Vachoux, and K. Einwich, "An introduction to modeling embedded analog/mixed-signal systems using systemc ams extensions," in *Open SystemC Initiative*, 2008.

[4] M. Barnasconi, C. Grimm, M. Damm, K. Einwich, M. Louërat, T. Maehne, F. Pecheux, and A. Vachoux, "Systemc ams extensions user's guide," *Accellera Systems Initiative*, 2010.

[5] M. Barnasconi, K. Einwich, C. Grimm, T. Maehne, and A. Vachoux, "Advancing the SystemC analog/mixed-signal (AMS) extensions," *Open SystemC Initiative*, 2011.

[6] M. Hassan, D. Große, H. M. Le, T. Vörtler, K. Einwich, and R. Drechsler, "Testbench qualification for SystemC-AMS timed data flow models," in *DATE*, 2018, pp. 857–860.

[7] F. Pêcheux, C. Grimm, T. Maehne, M. Barnasconi, and K. Einwich, "SystemC AMS based frameworks for virtual prototyping of heterogeneous systems," in *ISCAS*, 2018, pp. 1–4.

[8] M. Hassan, D. Große, H. M. Le, and R. Drechsler, "Data flow testing for SystemC-AMS timed data flow models," in *DATE*, 2019, pp. 366–371.

[9] M. Hassan, D. Große, T. Vörtler, K. Einwich, and R. Drechsler, "Functional coverage-driven characterization of RF amplifiers," in *FDL*, 2019, pp. 1–8.

[10] M. Hassan, D. Große, and R. Drechsler, "System-level verification of linear and non-linear behaviors of RF amplifiers using metamorphic relations," in *ASP-DAC*, 2021.

[11] ——, "System level verification of phase-locked loop using metamorphic relations," in *DATE*, 2021.

[12] R. Drechsler, Ed., *Advanced Formal Verification*. Kluwer Academic Publishers, 2004.

[13] P. Molitor and J. Mohnke, *Equivalence checking of digital circuits: fundamentals, principles, methods*. Springer Science & Business Media, 2007.

[14] R. Drechsler, *Formal System Verification*. Springer, 2018.

[15] L. Hedrich and E. Barke, "A formal approach to nonlinear analog circuit verification," in *Proceedings of IEEE International Conference on Computer Aided Design (ICCAD)*, Nov. 1995, pp. 123–127.

[16] M. H. Zaki, S. Tahar, and G. Bois, "Formal verification of analog and mixed signal designs: A survey," *Microelectronics Journal*, vol. 39, no. 12, pp. 1395–1404, Dec. 2008.

[17] A. Tarraf, L. Hedrich, N. Kochdumper, M. Rechmal-Lesse, and M. Olbrich, "Equivalence Checking Methods for Analog Circuits Using Continuous Reachable Sets," in *2020 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, Jul. 2020, pp. 7–12.

[18] L. Hedrich and W. Hartong, "Approaches to Formal Verification of Analog Circuits," in *Low-Power Design Techniques and CAD Tools for Analog and RF Integrated Circuits*, P. Wambacq, G. Gielen, J. Gerrits, R. van Leuken, A. de Graaf, and R. Nouta, Eds. Boston, MA: Springer US, 2001, pp. 155–191.

[19] W. Hartong, R. Klausen, and L. Hedrich, "Formal Verification for Nonlinear Analog Systems: Approaches to Model and Equivalence Checking," in *Advanced Formal Verification*, R. Drechsler, Ed. Boston, MA: Springer US, 2004, pp. 205–245.

[20] S. Steinhorst and L. Hedrich, "Advanced methods for equivalence checking of analog circuits with strong nonlinearities," *Formal Methods in System Design*, vol. 36, no. 2, pp. 131–147, Jun. 2010.

[21] A. Singh and P. Li, "On behavioral model equivalence checking for large analog/mixed signal systems," in *2010 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, Nov. 2010, pp. 55–61.

[22] A. Ain, S. Sanyal, and P. Dasgupta, "A Framework for Automated Feature Based Mixed-Signal Equivalence Checking," in *VLSI Design and Test*, ser. Communications in Computer and Information Science, B. K. Kaushik, S. Dasgupta, and V. Singh, Eds. Singapore: Springer, 2017, pp. 779–791.

[23] M. O. Saglamdemir, G. Dundar, and A. Sen, "An analog behavioral equivalence checking methodology for simulink models and circuit level designs," in *2015 International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD)*, Sep. 2015, pp. 1–4.

[24] M. Barnasconi, C. Grimm, M. Damm, K. Einwich, M. Louërat, T. Maehne, F. Pecheux, and A. Vachoux, "Systemc ams extensions user's guide," *Accellera Systems Initiative*, 2010.

[25] S. J. Mason, "Feedback Theory-Some Properties of Signal Flow Graphs," *Proceedings of the IRE*, vol. 41, no. 9, pp. 1144–1156, Sep. 1953.

[26] L. P. A. Robichaud, *Signal Flow Graphs and Applications*. Englewood Cliffs, N.J. :, 1962.

[27] P.-M. Lin, "Signal Flow Graphs in Filter Analysis and Synthesis," in *Circuit Analysis and Feedback Amplifier Theory*. CRC Press, 2006.

[28] Analog Devices, "Ltspice," https://www.analog.com/en/design-center/design-tools-and-calculators/ltspice-simulator.html.

[29] C.-W. Ho, A. Ruehli, and P. Brennan, "The modified nodal approach to network analysis," *IEEE Transactions on circuits and systems*, vol. 22, no. 6, pp. 504–509, 1975.

[30] D. Rowell and D. N. Wormley, *System Dynamics: An Introduction*. Upper Saddle River, NJ: Prentice Hall, 1997.

[31] F. R. Rasim and S. M. Sattler, "Analysis of Electronic Circuits with the Signal Flow Graph Method," *Circuits and Systems*, vol. 8, no. 11, pp. 261–274, Nov. 2017.