New Directions for Equivalence Checking of System-Level and SPICE-Level Models of Linear Circuits

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

***Abstract*—In this paper, we present a novel, graph-based methodology to formally check equivalence between system-level and SPICE-level representations of *Single-Input Single-Output* (SISO) linear analog circuits. To achieve this, we introduce a canonical representation in the form of a *Signal-Flow Graph* (SFG), which is used to functionally map the system-level and SPICE-level models. We create SFG representations for SPICE-level models and system-level models, and use graph manipulation techniques to transform the SFG representations into the canonical representation. We demonstrate the applicability of the methodology by successfully applying it to complex circuits.**

I. Introduction

The increasing complexity of analog circuits and their integration into *System-on-Chips* (SoC) have created a bottleneck for analog design verification. A major challenge in this regard is the simulation speed of traditional SPICE-level simulations [1]. Even though these simulations cannot be ignored due to their better accuracy, an expansion of system-level methodologies using SystemC AMS would be greatly beneficial. In particular, the *Timed Data Flow* (TDF) *Model of Computation* (MoC) available in SystemC AMS can provide a speed increase of over $100,000$ times in comparison to SPICE-level simulations [1] and allows 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. An increase in confidence is attainable with equivalence checking, which proves the general functional equality of two implementations of a design. While equivalence checking methods are well established in the digital domain [2], analog circuit design flows are lacking formal or at least formalized verification methodologies [3].

Contribution: In this paper, we present our first-of-its-kind equivalence checking methodology [4], [5]. Essentially, our approach transforms the system-level and SPICE-level models into a canonical representation for comparison.

Summarizing the main contributions of this paper:

- We leverage SFGs as an intermediate, mutual representation for SPICE-level and system-level models. To create SPICE-level SFGs, we use linear graph modeling.
- We transform the SFGs of both models to a canonical form with several graph operations.

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

- The methodology spans the complete class of complex SISO linear analog circuits.
- We demonstrate the applicability by applying our methodology to a filter model and a linear analog computer.

II. Signal-Flow Driven Equivalence Checking

A. Proposed Methodology

A block-diagram overview of our methodology for equivalence checking between system-level and SPICE-level models is seen in Fig. 1. To generate a set of equations, we use the "Linear Graph Modeling" method [6], then create SFGs with SFG creators, reduce them to a canonical form with the "SFG simplifier", and compare them with the "Equivalence checker".

A linear SFG [7] is a representation of the equation

$$x_i = \sum_j a_{ji} x_j + \sum_k b_{ki} u_k \tag{1}$$

in the form of a graph, where x_i and x_j are variables of the circuit, a_{ji} and b_{ki} are constants, and u_k are inputs.

The system-level SFG is created from a SystemC AMS description that uses linear operators such as addition, multiplication by a constant, and *Laplace Transfer Functions* (LTF). Since programming code is already written in an explicit form similar to Equation (1), it can be directly transformed into an SFG.

For the creation of the SPICE-level SFG, a set of linear explicit algebraic equations in the form of Equation (1) are obtained with the linear graph modeling method [6], which consists of two main steps.

First, a normal tree, which is a special type of minimum spanning tree of the circuit graph, is created. 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. Secondly, depending on which components are on the normal tree, explicit

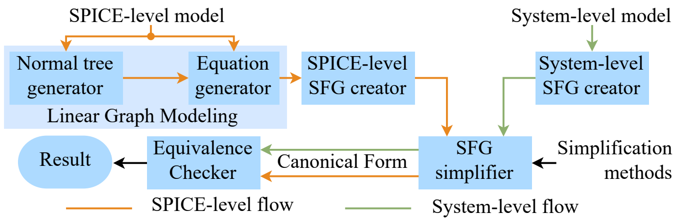

Fig. 1. Overview of the proposed equivalence checking methodology

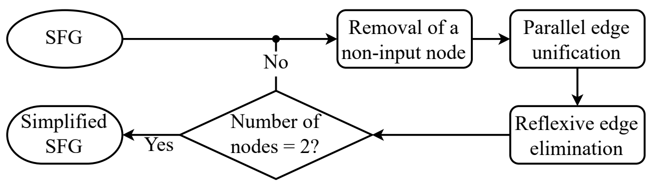
SFG
Removal of a non-input node
Parallel edge unification
No
Simplified SFG
Yes
Number of nodes = 2?
Reflexive edge elimination

Fig. 2. Overview of SFG simplification process

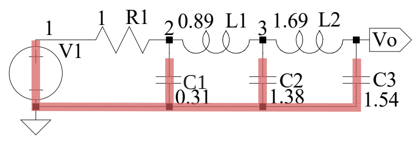

Fig. 3. Analog fifth-order low-pass filter and its highlighted normal tree.

expressions for all variables are generated by the equation generator through either elemental, compatibility (Kirchhoff's voltage law), or continuity (Kirchhoff's current law) equations.

The "SFG simplifier" reduces the SFGs from the system-level and SPICE-level implementations to canonical forms as given in Fig. 2. The simplification rules [8] used during this process are:

a) Removal of a non-input node: A non-input node n_x may be removed after creating edges from its ancestors (a_x) to its descendants (d_x). These new edges (a_x, d_x) are created with weights $w((a_x, d_x))$ equal to $w((a_x, n_x)) \cdot w((n_x, d_x))$.

b) Parallel edge unification: According to the distributive law for parallel edges, these can be merged into a single edge by summing their weights.

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

B. Illustration

We illustrate our methodology on a single-input ($V1$) single-output (V_o) analog fifth-order passive low-pass filter (Fig. 3) taken from [9]. Its system-level model is given as

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

and is implemented in SystemC AMS, which provides an LTF solver.

As the first step of our methodology, we obtain the circuit's normal tree, highlighted in Fig. 3. Then, we use this normal tree to get the explicit equations. For example, the equations for $C1$ are given as $V_{C1} = \frac{1}{0.31s} I_{C1}$ and $I_{C1} = I_{R1} - I_{L1}$.

Since these equations are in the form of Eq. 1, the SFG given in Fig. 4a can be created directly. This SFG is then reduced according to the rules given in Section II-A. An intermediate result during this reduction is given in Fig. 4b. 12 nodes are

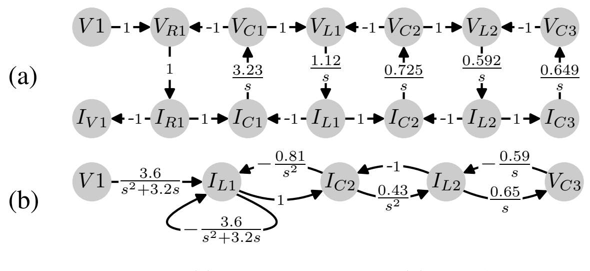

Fig. 4. The initial **(a)** and an intermediate **(b)** SFG for the low-pass filter.

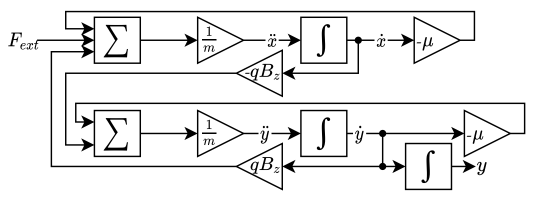

Fig. 5. System-level block diagram of the analog simulator for a particle in a magnetic field.

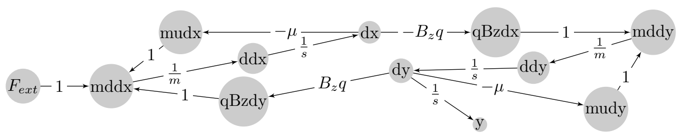

Fig. 6. Initial system-level SFG of the analog simulator for a particle in a magnetic field.

removed in total. It is concluded that the models are equal, since the final SFG is equal to the system-level LTF seen in Eq. 2.

III. Experimental Evaluation

We consider the analog computing circuit, from [10], which simulates the behavior of a charged particle under a magnetic field. As input, we add an external force (F_{ext}) and use the y position of the particle as output. The SystemC AMS implementation corresponds to the system-level block diagram in Fig. 5.

The SPICE-level model of the circuit is implemented by using template circuits that act as inverting summers, integrators, and gains. The initial SPICE-level SFG was obtained with 85 nodes and 132 edges and reduced to canonical form. The LTF of this final form was obtained as

$$-\frac{1}{s^3 + 2 \cdot 10^9 s^2 + 2 \cdot 10^{18} s} \tag{3}$$

The SystemC AMS code resulted in the system-level SFG in Fig. 6. After the simplification process and after substituting numeric values, the canonical form with the same transfer function given in Eq. 3 is obtained. Therefore, it is concluded that both representations are equal.

The total run time for this example was 4.9 s.

Future Work: In future, we plan to extend the method to multi-input multi-output systems and analyze systems with external noise input.

References

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

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

[3] 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.

[4] K. Ç. Coşkun, M. Hassan, and R. Drechsler, "Equivalence Checking of System-Level and SPICE-Level Models of Linear Analog Filters," in *Design and Diagnostics of Electronic Circuits and Systems (DDECS)*, Prague, 2022.

[5] K. Ç. Coşkun, M. Hassan, and R. Drechsler, "Equivalence Checking of System-Level and SPICE-Level Models of Linear Circuits," *Chips*, vol. 1, no. 1, pp. 54–71, Jun. 2022.

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

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

[8] 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.

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

[10] B. Ulmann, *Analog and Hybrid Computer Programming*. Walter de Gruyter GmbH & Co KG, Jun. 2020.