Les Cahiers du GERAD ISSN: 0711–2440

A Vertex Cut Algorithm for Model Order Reduction of Electronic Circuits

P. Kitanov, O. Marcotte,
W. Schilders, S.M. Shontz

G–2011–21

April 2011
Revised: October 2011

A Vertex Cut Algorithm for Model Order Reduction of Electronic Circuits

Petko Kitanov

Applied Mathematics
University of Guelph
Guelph (Ontario) Canada
pkitanov@uoguelph.ca

Odile Marcotte

GERAD & Department of Computer Science
Université du Québec à Montréal
Montréal (Québec) Canada
odile.marcotte@gerad.ca

Wil H. A. Schilders

Numerical Mathematics for Industry
Eindhoven University of Technology & Dutch Platform for Mathematics
Eindhoven, The Netherlands
w.h.a.schilders@tue.nl

Suzanne M. Shontz

Computer Science and Engineering
The Pennsylvania State University
University Park, PA, USA
shontz@cse.psu.edu

April 2011
Revised: October 2011

Les Cahiers du GERAD
G–2011–21

Abstract

In this article we address the model order reduction problem for resistor networks by using methods from graph theory. We formulate this problem through graph theory concepts, propose algorithms for solving it, and present the computational results we have obtained for real-world resistor networks. The results demonstrate that graph-theoretical methods produce networks that contain fewer edges and are sparser than networks produced by state-of-the-art methods.

Key Words: Circuit simulation, graph algorithms, model order reduction, parasitic extraction, path resistance, resistor networks, vertex cut.

Résumé

Dans cet article nous nous servons de méthodes de théorie des graphes pour étudier le problème de la réduction du modèle dans le contexte des réseaux de résistances. Nous formulons le problème grâce à des concepts de la théorie des graphes, proposons des algorithmes pour le résoudre et présentons les résultats que nous avons obtenus pour des réseaux concrets. Nos résultats démontrent que des méthodes de la théorie des graphes permettent de construire des réseaux qui contiennent moins d'arêtes et sont plus épars que les réseaux produits par les méthodes précédentes.

Mots clés : théorie des graphes, réduction de modèles, réseaux de résistance.

Acknowledgments: The authors of this article met in August 2008 at the Fields-MITACS Industrial Problem-Solving Workshop. They are very grateful to the organizers of this workshop for inviting them to participate in the Fields-MITACS IPSW. Odile Marcotte and Suzanne Shontz would also like to express their gratitude to Professor Cor Hurkens and Professor Judith Keijsper, from the Eindhoven University of Technology, for their generous hospitality. The work of Professor Marcotte was funded in part by an NSERC Discovery Grant and the work of Professor Shontz was funded in part by NSF grant CNS-0720749 and NSF CAREER Award OCI-1054459. Finally the authors wish to thank Dr. Joost Rommes (from NXP Semiconductors N.V.) and Dr. Maria V. Ugryumova (Eindhoven University of Technology) for their help.

1 Introduction

Coupling effects between components in VLSI chips play an increasingly important role, and need to be addressed thoroughly during the design and verification phases. Analysis of these effects is performed by co-simulating the nonlinear circuits together with the extracted parasitics. Because of the increasing amount of parasitics, full device-parasitic simulations are too costly and often impossible. Hence reduced models are sought for the parasitics, which can reproduce the original circuit behavior when re-coupled to the devices.

Parasitic circuits are very large network models containing millions of basic circuit elements: R, RC, or RLC(k), and also millions of nodes. Recall that R (resp. RC, RLC(k)) stands for resistor (resp. resistor-capacitor, resistor-inductor-capacitor) network. A special subset of the nodes is the set of terminals, i.e., the input/output nodes specified by the designer as well as the nodes connecting the parasitics to the nonlinear devices in the circuit. Parasitic networks corresponding to modern circuits contain millions of nodes and R/RC/RLC(k) elements, and thousands of terminals. Ideally a reduced order model for the parasitics has fewer nodes and circuit elements than the original network, but includes the same terminal nodes in order to guarantee that it is connected to the same devices as the original network.

The analysis is complicated by the presence of many terminals, because it introduces additional structural and computational challenges during model order reduction (MOR). Existing MOR methods are, in general, unsuitable for circuits with many terminals since they produce dense reduced models. The circuits corresponding to these reduced models contain fewer nodes, but more circuit elements (Rs, Cs, Ls), than the original circuits, and their simulation often takes more time than the simulation of the original circuits. In addition, if terminal connectivity is affected, additional elements such as current/voltage controlled sources must be introduced to model the reconnection of reduced parasitics to other devices.

The emerging problem is to develop efficient MOR schemes for large multi-terminal circuits that are accurate, preserve terminal connectivity, and most importantly, preserve network sparsity. In this article, we describe methods that achieve these goals by employing graph theoretical algorithms. The work builds on [1], where a method called reduceR was developed, and on ideas first presented in [2]. In the present article we exploit the strength of modern graph theory to obtain a much higher degree of reduction, and more significantly, reduced models that are much sparser than before. Although the problem of achieving a maximum reduction of a given circuit is likely to be NP-hard and our approach is a heuristic one, the methods described in the present article achieve tremendous reduction rates even for circuits with tens of thousands of nodes, and yield an accurate approximation of the input/output behavior of the original circuits.

In this article, we restrict ourselves to resistor networks so as to demonstrate clearly the use of the graph-theoretical methods. Application to RC networks is possible but requires MOR techniques for differential-algebraic systems; these techniques must be developed further and combined with the techniques described in the present article in order to tackle RC networks. The article is structured as follows. Section 2 contains a review of work related to ours. Section 3 contains background material on the model order reduction problem for resistor networks. In Section 4 we give a precise formulation of the reduction problem, and in Section 5 we present algorithms for reducing resistor networks. Section 6 contains the results of our computational experiments, and Section 7 contains our conclusions.

2 Related work

Model order reduction consists of several techniques that reduce a very large system of differential (difference, differential-algebraic) equations, modeling some physical problem, to a system of much smaller dimension. It is commonly applied in many areas of applied mathematics. We want to reduce the complexity of the system and at the same time preserve the input-output behavior of the original system as much as possible. In particular, the physical characteristics of the reduced system should approximate closely those of the original system. MOR is widely used for simulating and analyzing dynamical systems, electronic circuits, micro- and nano-electromechanical systems, and systems arising in computational fluid dynamics. For example, very large RLC circuits are usually modeled by systems that include millions of equations; if we want to

simulate their behavior within an acceptable amount of time, we first need to simplify the original system while preserving the relevant properties.

Existing methods for linear model order reduction can be divided into two groups [3]. The methods in the first group are based on projection methods, whereas those in the second group are based on the singular value decomposition (SVD). The SVD-based methods are sometimes called balancing techniques. Methods in the first group are usually based on Krylov subspace projection methods. An important contribution is found in Odabasioglu et al. [4], where an algorithm is proposed for generating provably passive reduced-order N-port models for RLC interconnect circuits. The approach proposed in [4], called PRIMA, is a general method for obtaining passive reduced-order macromodels for arbitrary RLC circuits. For small- and medium-sized problems, the *truncated balance realization* (TBR) technique (see [3,5]) is often employed. Hybrid techniques that combine some of the features of both groups of methods are studied by Li et al. in [6]. Many of the algorithms in existence today are based on hybrid techniques. The book by Schilders et al. [7] is currently the most comprehensive overview of MOR methods both for linear and nonlinear problems.

The applicability of traditional MOR techniques to very large circuits with many terminals is limited because of computational limitations and sparsity and reconnectivity considerations (mentioned in the Introduction). While the multi-terminal problem has been addressed in numerous works (such as [8] and [9]), it is not clear whether the performance of these methods will remain good as the number of ports increases, especially when there are thousands of ports. Recent developments in model reduction for very large multi-terminal R-networks were achieved in Rommes and Schilders [1], who present a method called reduceR that uses graph-theoretical tools, fill-in minimizing, node reorderings, and node elimination in order to obtain sparse reduced R-networks. The Sparse implicit projection (SIP) method of Ye et al. [10], designed to build sparse reduced models for multi-terminal RLC networks, also includes reordering actions followed by the elimination of unimportant internal nodes; the authors make several important analogies between related methods based on node elimination (e.g., TICER in [11], and [12]) and moment-matching MOR by projection (e.g., PRIMA in [4]). Indeed the fundamental projection behind SIP goes back to the PACT methods of [13] and [14] for reducing multi-terminal RLC networks.

Few researchers have explored graph-theoretical techniques for model order reduction. Graph reduction techniques are mainly studied in theoretical computer science for analyzing functional languages [15]. In [16], an algorithm is presented that uses a set of graph reduction rules to identify structural conflicts in process models for a generic workflow modeling language. Graph reduction techniques for analyzing process models are studied in [17]. For general background on graph theory we refer the reader to [18–20].

3 Model order reduction of resistive circuits

To understand the electrical properties of a purely resistive circuit (R-network), one takes measurements on the substrate of that circuit in order to set up Maxwell's equations. Then those equations are discretized, and one obtains a version of Kirchhoff's equations. In abstract terms, the circuit can be viewed as a graph whose edges correspond to wires and vertices (or nodes) to intersections between wires. Some vertices are *external* (i.e., linked to vertices in other circuits) and the others are *internal*. To each edge corresponds the resistance of that edge. Therefore we can view a purely resistive circuit as an edge-weighted undirected graph whose vertices are either white or black (white meaning internal and black external). A tiny example, i.e., a *model* of a purely resistive circuit, is displayed in Figure 1.

The example in Figure 2 is more realistic, since it is part of a graph extracted from a real circuit provided by the company NXP Semiconductors N.V. (see Section 6). Note that the external vertices are represented by squares and the graph contains many "bottlenecks", i.e., small subsets of vertices whose removal breaks the graph into smaller graphs. In graph theory, these bottlenecks are called *vertex cuts* (see below for a definition).

As explained in Section 1, simulation of resistive circuits can take an enormous amount of time, and may indeed be impossible. Thus one would like to replace the original model by a smaller model containing the same black (external) vertices and having the following property: the path resistance between any two black

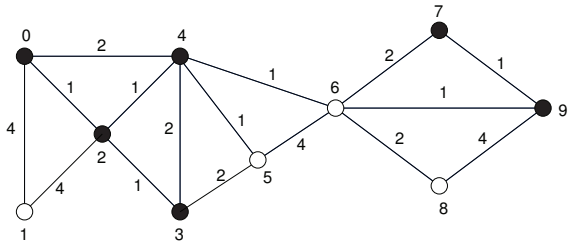

Figure 1: A tiny example

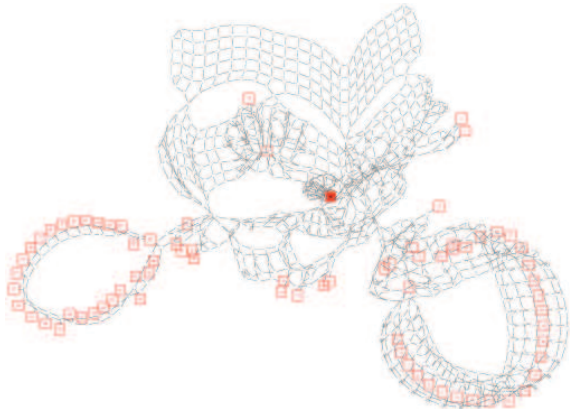

Figure 2: The bicycle

vertices is the same in the smaller model as in the original one. This property guarantees that the two models have the same physical characteristics. Before explaining how to reduce the original model, we review some concepts from graph theory (see [20]).

Definition 3.1 *An undirected graph $G = (U, \mathcal{E})$ consists of a set U of vertices and a set $\mathcal{E}$ of edges, where an edge e is defined as a pair of (distinct) vertices.*

An edge $\{u, v\}$ in a graph G will be denoted by uv (note that uv is identical to vu). The vertices u and v are called the *ends* or *endpoints* of edge uv.

Definition 3.2 *An undirected graph $G = (U, \mathcal{E})$ is said to be* complete *if its edge set includes every pair of vertices.*

Definition 3.3 *Let G be an undirected graph. A* path *in G is a sequence $(u_0, u_1, u_2, \ldots, u_\ell)$ of distinct vertices with $\ell \geq 1$ and such that $u_i u_{i+1}$ is an edge for $i = 0, 1, \ldots, \ell - 1$. We say that G is* connected *if there exists a path between any two vertices in G, and* disconnected *otherwise.*

Definition 3.4 *Let $G = (U, \mathcal{E})$ be an undirected connected graph. A* vertex cut *in G is a set of vertices whose removal disconnects G. If a vertex cut has one element only, we say that this vertex is a* cut-vertex. *An* edge cut *in G is a set of edges whose removal disconnects G. In particular, if X is a nonempty proper subset of U, the set of edges with one end in X and the other in $U \backslash X$ is an edge cut that will be denoted by $[X, U \backslash X]$. In this case X and $U \backslash X$ are called the* shores *of the edge cut.*

In the graph of Figure 1, $\{2, 4\}$ and $\{6\}$ are vertex cuts, 6 is a cut-vertex, and the set $\{35, 45, 46\}$ is an edge cut with shores $\{0, 1, 2, 3, 4\}$ and $\{5, 6, 7, 8, 9\}$.

Definition 3.5 *Let $G = (U, \mathcal{E})$ be an undirected graph and X a subset of U. Then the* subgraph induced by *X is the graph $(X, \mathcal{F})$, where $\mathcal{F}$ denotes the set of edges both ends of which belong to X. The edge set $\mathcal{F}$ may also be denoted by $\mathcal{E}(X)$.*

Definition 3.6 *Let G be an undirected graph. A* connected component *of G is a maximal induced subgraph of G that is connected. A connected graph G is said to be* 2-connected *if it does not contain any cut-vertex. If G is connected, a* 2-connected component *of G is defined as a maximal 2-connected subgraph of G.*

The 2-connected components of the graph displayed in Fig. 1 are induced by the sets $\{0,1,2,3,4,5,6\}$ and $\{6,7,8,9\}$, respectively. Note that the 2-connected components of a graph may have vertices in common but that their edge sets are pairwise disjoint.

We are now ready to discuss Kirchhoff's equations for a purely resistive circuit. Note that in such a circuit, the current flows from one end of an edge to the other end. The orientation does not matter since it can be changed by modifying the sign of the current. In the sequel we choose an orientation for each edge and "replace" the edge uv by the arc (u,v), in which the current flows from u to v. The graph thus obtained from G is a directed graph G'. We denote by n the number of vertices of G (or G') and by m the number of edges of G (or the number of arcs of G'). The system of Kirchhoff's equations is

$$R\,I - P\,V = 0,\ P^t\,I = J,$$

where R is an $m \times m$ diagonal resistance matrix, P the arc-vertex incidence matrix of G (of dimension $m \times n$), I an m-dimensional vector of currents flowing in the arcs, V an n-dimensional vector of voltages at the vertices, and J an n-dimensional vector of terminal currents flowing into the interconnection system.

From the above linear system one derives the equation $\left(P^t R^{-1} P\right) V = J$. Let M denote the matrix $P^t R^{-1} P$. From M one can deduce the resistances between all pairs of vertices. Observe that for $u \neq v$, the element of matrix M at the intersection of row u and column v equals $-1/r_{uv}$, where r_{uv} denotes the resistance of edge uv. The diagonal elements of M are defined in such a way that the sum of the entries of M in any row or column equals 0. Thus the definition of M depends only upon the graph G and not on the orientation chosen for the arcs of G'. Note that M is singular because it has an extra row corresponding to a ground node; nonetheless the above system has a unique solution.

Assume now that X is a vertex cut consisting of white vertices in the graph G. The removal of X disconnects the graph G, i.e., the subgraph induced by $U \backslash X$ is disconnected. Assume that one of the connected components induced by $U \backslash X$ contains some white vertices. Then the union of X and the set of black vertices in this component is also a vertex cut, which we denote by U_2. The set $U \backslash U_2$ can then be partitioned into U_1 (consisting of white vertices only) and U_3 (the rest of the graph) such that any edge in $\mathcal{E}\left(U \backslash U_2\right)$ belongs to $\mathcal{E}\left(U_1\right)$ or $\mathcal{E}\left(U_3\right)$. We now explain how to "eliminate" the vertices in U_1. Note that the so-called Y-Δ transformation is the special case of this elimination procedure where U_1 is a singleton and $X = U_2$ consists of white vertices (for a discussion of the Y-Δ transformation see the chapter by Nick P. van der Meijs in [7]).

The partition of U into the sets U_1, U_2, and U_3 induces the following decomposition of the matrix P:

$$\begin{pmatrix} A & 0 & 0 \\ B & C & 0 \\ 0 & D & 0 \\ 0 & E & F \\ 0 & 0 & G \end{pmatrix},$$

where A (resp. D, G) is the arc-vertex incidence matrix of the subnetwork induced by U_1 (resp. U_2, U_3), the submatrix $[B,C]$ is the arc-vertex incidence matrix of the edge cut with shores U_1 and $U_2 \cup U_3$, and the submatrix $[E,F]$ is the arc-vertex incidence matrix of the edge cut with shores $U_1 \cup U_2$ and U_3. This partition of P in turn induces a partition of J into the vectors J_1, J_2, and J_3, and the following partition of the matrix M:

$$\begin{pmatrix} A^t R_1^{-1} A + B^t R_2^{-1} B & B^t R_2^{-1} C & 0 \\ C^t R_2^{-1} B & N & E^t R_4^{-1} F \\ 0 & F^t R_4^{-1} E & F^t R_4^{-1} F + G^t R_5^{-1} G \end{pmatrix},$$

where N is defined as $C^t R_2^{-1} C + D^t R_3^{-1} D + E^t R_4^{-1} E$. The vector V is similarly partitioned into the vectors x, y, and z.

We now wish to eliminate from the system $\left(P^t R^{-1} P\right) V = J$ the subvector x, which corresponds to a set containing white vertices only. Using Gaussian elimination we obtain a system whose constraint matrix (denoted $M_{reduced}$) is the following:

$$\begin{pmatrix} H^t R_6^{-1} H + E^t R_4^{-1} E & E^t R_4^{-1} F \\ F^t R_4^{-1} E & F^t R_4^{-1} F + G^t R_5^{-1} G \end{pmatrix},$$

where H is defined in such a way that

$$-C^t R_2^{-1} B A' B^t R_2^{-1} C + C^t R_2^{-1} C + D^t R_3^{-1} D = H^t R_6^{-1} H$$

and A' denotes the inverse of $A^t R_1^{-1} A + B^t R_2^{-1} B$. Such a matrix H exists. Indeed, one may choose for H the arc-vertex incidence matrix of any directed graph obtained by assigning orientations to the edges of the complete graph on U_2. The path resistance between two vertices in U_2 (within the subgraph induced by $U_1 \cup U_2$) is given by the corresponding entry of the matrix on the left-hand side of the equation. To complete the definition of the reduced graph, it suffices to assign the value of the path resistance to the edge linking these two vertices in the complete graph.

To illustrate the construction of the reduced model, we consider again the weighted graph in Figure 1. We first reduce it by taking $X = \{6\}$, $U_2 = \{0, 2, 3, 4, 6\}$, $U_1 = \{1, 5\}$, and $U_3 = \{7, 8, 9\}$. Then we reduce it by taking $X = \{6\}$, $U_2 = \{6, 7, 9\}$, $U_1 = \{8\}$, and $U_3 = \{0, 2, 3, 4\}$. The final graph is displayed in Figure 3. Note that the edges 06, 26, and 03 have an infinite resistance and could have been deleted from the graph. This example shows that it is not always necessary to create a complete graph on U_2 when reducing the original graph. Note that in this case, the number of edges of the reduced graph is smaller than that of the original graph.

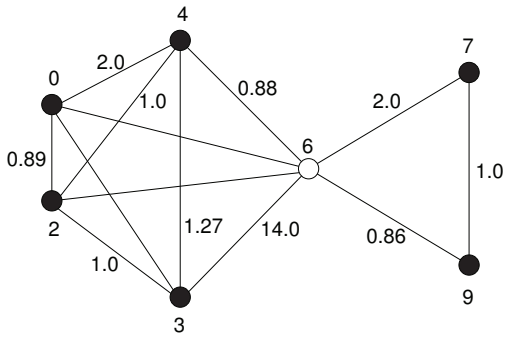

Figure 3: A reduced graph with the same physical properties as the tiny example

4 Problem statement

From the previous section it is clear that the reduction of the model can be expressed in two equivalent forms: in terms of graphs or in terms of matrices. The term *block* will thus be used to denote the set of vertices $U_1 \cup (U_2 \backslash X)$ or the submatrix corresponding to this set. The set X is included in the *border*. The basic operation of model order reduction is a *block reduction*. The goal of model order reduction is to obtain a system that can be solved in less time than the original system, and we use the number of edges of the underlying graph as a measure of the time required to solve a system. We have chosen this criterion (the number of edges) because trying to minimize the number of vertices (or nodes) tends to produce dense matrices, which might make the reduced circuits harder to simulate. In any case, trying to minimize the number of edges and the number of vertices at the same time raises the question of assigning weights to each of the criteria. We now present a formal statement of the problem.

Given an undirected graph $G = (U, \mathcal{E})$ in which every vertex is either black or white, we are looking for a partition of U of the form $\{B_1, B_2, \ldots, B_k, B_0\}$, where B_i (for $1 \leq i \leq k$) is a *block* and B_0 is the *border*. The removal of B_0 from the graph G "disconnects" G into subgraphs induced by the B_is, i.e., there is no edge in G between a vertex in B_i and a vertex in B_j for any $i \neq j$. In what follows B_i^e denotes the subset of black vertices in B_i. Let B_{0i} be defined as

$$\{v \in B_0 \mid (\exists u \in B_i)\,(uv \in E)\}\,.$$

We process the blocks in increasing order of their indices and define the *reduction* associated to block B_i (and denoted by $R\,(B_i)$) as

$$|E\,(B_i \cup B_{0i})\,| - (|B_i^e| + |B_{0i}|)\,|\,(|B_i^e| + |B_{0i}| - 1)\,|/2 + m_i,$$

where m_i is the number of edges with both endpoints in B_{0i} that were included into the border in steps $1, 2, \ldots, i-1$. In this expression the second term represents the number of edges in the complete graph on $B_i^e \cup B_{0i}$, which "replaces" the subgraph induced by $B_i \cup B_{0i}$.

If all the $R\,(B_i)$ are nonnegative, the reduction associated to the partition is the sum of the $R\,(B_i)$. If $R\,(B_i)$ is negative for some i, we can decide not to reduce block B_i! Then the reduction associated to block B_i equals 0. Let $\mathcal{P}$ denote the partition $\{B_1, B_2, \ldots, B_k, B_0\}$. The reduction associated to $\mathcal{P}$ is defined as

$$R\,(\mathcal{P}) = \sum_{i=1}^{k} \max\{R\,(B_i)\,, 0\}\,.$$

Our problem is thus to find a partition $\mathcal{P}$ that maximizes $R(\mathcal{P})$, i.e., minimizes the number of edges of the reduced graph. Such a partition is called an *optimal partition*. Once a partition has been found, the following procedure must be applied to each block B_i (in the order $1, 2, \ldots, k$): B_i is replaced by the complete graph on $B_i^e \cup B_{0i}$ (i.e., the white vertices are eliminated), and the resistances on the edges of this complete graph are then computed (using the definition of matrix H given above).

5 Algorithms for model order reduction in the resistive case

The combinatorial problem described in the previous section is probably NP-complete, although we do not have a formal proof of this fact yet. In this article we present heuristic algorithms based on the search for small cardinality vertex cuts. Even if there is no guarantee that such vertex cuts are subsets of the border in an optimal solution, the characteristics of the circuit graphs on which we ran the algorithm led us to think that we would obtain a good solution if we tried to "isolate" black vertices by finding small cardinality vertex cuts separating black vertices from other black vertices. Recall that the border must include white vertices only; thus, the vertex cuts computed by the algorithm must consist of white vertices.

Obviously, if two black vertices are adjacent, they cannot be separated by a vertex cut consisting of white vertices. Hence we introduce the notion of black region. A *black region* is a set of vertices inducing a subgraph in which the black vertices are "close" to one another. The definition of black region depends upon a parameter, *distBlack*, which represents the maximum number of consecutive white vertices between any two black vertices in the region. For instance, when *distBlack* equals 0, the subgraph induced by a black region is a maximal connected subgraph consisting of black vertices only. When *distBlack* equals 1, the subgraph induced by a black region has the following properties:

- for any two black vertices in the region, there is a path between those two vertices that does not contain two consecutive white vertices,
- if two black vertices do not belong to the same region, then any path between these two vertices contains at least two consecutive white vertices.

In the example of Figure 4 (the "toy example"), we assume that *distBlack* equals 0. Then there are five black regions: $\{1, 37, 38, 39, 53, 56, 57\}$, $\{17\}$, $\{32\}$, $\{42\}$, and $\{47\}$.

In Section 3 we introduced the notion of "shore" in the context of edge cuts. In this section we use the term in the context of vertex cuts.

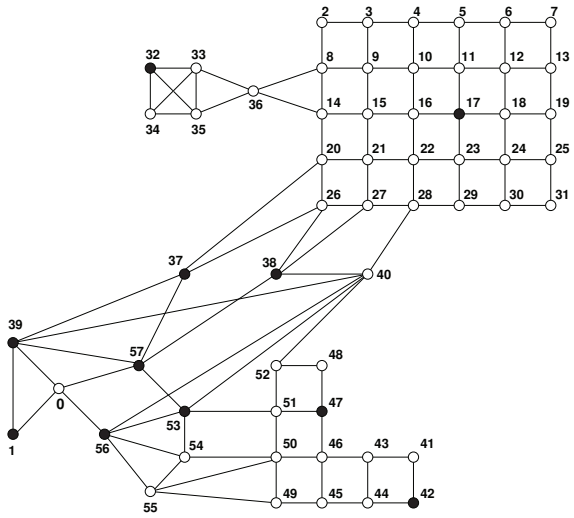

Figure 4: A toy example

Definition 5.1 *Let $G = (U, \mathcal{E})$ be an undirected graph and X a vertex cut in G. The* shores *of X are the vertex sets of the connected components of the subgraph of G induced by $U \backslash X$.*

We also define the border of a subset of vertices in a graph.

Definition 5.2 *Let G be an undirected graph and S a subset of vertices of G. The* border *of S is the set of vertices u of G that do not belong to S but are adjacent to at least one member of S.*

For instance, in the toy example, the shores of the vertex cut $\{46, 49\}$ are the sets $\{41, 42, 43, 44, 45\}$ and $V \backslash \{41, 42, 43, 44, 45, 46, 49\}$ (where V denotes the vertex set of the graph). The border of the largest black region is the set $\{0, 20, 26, 27, 40, 51, 54, 55\}$.

Before presenting our algorithms, we formalize what we mean by "isolating black vertices" and discuss briefly the procedure that returns a minimum cardinality vertex cut separating two vertices. We will use the phrase *white vertex cut* to denote a vertex cut consisting of white vertices only. We say that a vertex cut separates vertex u from vertex v if u and v do not belong to the same shore of the vertex cut. Similarly a vertex cut separates two black regions if those regions are not included in the same shore of the vertex cut. A procedure for finding a vertex cut (of any color) between two vertices is described in Sections 7.3 and 8.2 of [20]. This procedure reduces the vertex cut problem to an edge cut problem, which in turn is reduced to a network flow problem in which all the capacities equal 1. To make sure that this procedure returns a white vertex cut (if there is one), it suffices to assign an infinite capacity to each arc replacing a black vertex in the directed graph on which the network flow algorithm is applied. Although the procedure described in [20] returns a vertex cut separating two vertices, it can be modified in a straightforward manner to return a vertex cut separating two subsets of vertices.

The *basic algorithm*, given below, starts by computing the connected components of the graph and the black regions of every component. It then processes each component and tries to "isolate" black regions within that component. The algorithm uses a FIFO list (i.e., a queue) whose elements are sets of vertices waiting to be subdivided. Initially the FIFO list contains one set only, the set of all vertices of the current component (denoted S). The algorithm finds a minimum cardinality white vertex cut containing at most *limitCut* vertices and separating the largest black region from the vertex (denoted by v) that is furthest from it. The reduction corresponding to the shore that contains v (denoted by $PieceV$) is the difference between

- the number of edges in the subgraph induced by $PieceV$ and its border, and
- the number of edges in the complete graph on the union of the set of black vertices in $PieceV$ and the border of $PieceV$.

A good vertex cut has been found if the reduction corresponding to $PieceV$ is at least a certain percentage ($percentLimit$) of the number of edges of the subgraph induced by $PieceV$ and its border, or if the cardinality of $PieceV$ is sufficiently large with respect to that of S, i.e., if the product of a parameter called $factor$ and the cardinality of $PieceV$ is at least the cardinality of S. When a good vertex cut has been found, each of its shores (not only that containing v) is evaluated. If the shore contains fewer than $minNbBlack$ vertices **or** its percentage of black vertices is at least $blackLimit$ **or** its reduction is large enough (see the above criterion), then the shore need not be separated into smaller pieces and is considered a block. Otherwise the shore is appended to the FIFO list.

We mentioned above that a network flow algorithm can be used to find a vertex cut separating two subsets of vertices. Before calling this algorithm, however, one must check that the subsets are *compatible*, i.e., there is no path between the two subsets in which all the internal vertices are black. Our first algorithm is summarized below and consists of the *Main algorithm* and the sub-algorithm *Separate a component into pieces*. The input to the main algorithm consists of the edges of an undirected graph, along with the resistances of those edges and the color (white or black) of every vertex.

Main algorithm

Compute the connected components
Compute the set of black vertices within each connected component
Compute the set of black regions within each connected component
for each connected component **do**
  separate the component into pieces by finding vertex cuts in the component
end for

Separate a component into pieces

insert the set $SInitial$ of all vertices into the FIFO list Q
while Q is not empty **do**
  let S be the first set in Q
  remove S from Q
  find a black vertex in S belonging to a black region R of maximum cardinality
  let $Source$ be the intersection of S and R
  let $Targets$ denote the set of all targets (i.e., S, for the time being)
  $goodVertexCut \leftarrow$ **false**
  while (**not** $goodVertexCut$) **and** (there are more targets) **do**
    let v denote the target furthest from $Source$
    if $Source$ and v are compatible **then**
      find a minimum cardinality white vertex cut $VCut$ separating $Source$ from v
      if cardinality of $VCut \leq limitCut$ **then**
        $PieceV \leftarrow$ shore of $VCut$ containing v
        $goodVertexcut \leftarrow$ (the reduction of $PieceV$ is large enough) **or** $(factor * |PieceV| \geq |S|)$
      end if
    end if
    remove v from $Targets$
  end while
  if $goodVertexCut$ **then**
    for each shore T determined by $VCut$ **do**
      if there is no need to decompose T **then**
        create a block corresponding to T
      else
        append T to the FIFO list Q
      end if
    end for
  else
    create a block corresponding to S

30: **end if**
31: **end while**

The main algorithm produces a partition of the vertex set of the graph into blocks and a border, along with the information that a given block should be reduced (or not). In a block that cannot be reduced, one can nonetheless try to apply the Y-Δ transformation. Actually the circuit graphs we use in our tests have many vertices of degree three (see the next section). Thus the repeated application of the Y-Δ transformation may result in a substantial reduction in the number of vertices (and possibly the number of edges) of a block that cannot be reduced otherwise. Indeed it is even possible to apply a sequence of Y-Δ transformations to a block whose reduction is positive and thus obtain a "better" reduced subgraph. Therefore, after executing the main algorithm, we can examine every block in order to find out which reduction is the best one: the "global reduction" or a sequence of Y-Δ transformations. Of course we choose the best reduction for each block, and if the graph obtained still contains more edges than the complete graph on the set of black vertices, we replace it by this complete graph. We will call *basic algorithm* the main algorithm followed by the *improvement phase* comparing the reduction obtained by the main algorithm with the repeated application of the Y-Δ transformation.

We now describe the steps of the basic algorithm on the toy example pictured in Figure 4. First the algorithm tries to separate the largest black region (the set $\{1, 37, 38, 39, 53, 56, 57\}$, in this case) from the rest of the graph. In order to do this, it computes a minimum cardinality white vertex cut separating the region $\{1, 37, 38, 39, 53, 56, 57\}$ from the vertex that is furthest from this region (i.e., vertex 7). The vertex cut returned by the algorithm is $\{20, 27, 40\}$ and its shores (V_1 and V_2) are evaluated. Let V_1 denote the shore containing the largest black region (i.e., the set $\{0, 1, 37, 38, 39, 41..57\}$, where 41..57 denotes the set of all vertices comprised between 41 and 57) and V_2 the shore including vertex 7 (i.e., $V \backslash (V_1 \cup \{20, 27, 40\})$). Replacing the subgraph induced by $V_2 \cup \{20, 27, 40\}$ by the complete graph on $\{17, 32, 20, 27, 40\}$ reduces the number of edges by more than 75% (we assume that *percentLimit* equals 75.0). Thus V_2 is considered a block of the decomposition and will not be appended to the FIFO list.

On the other hand V_1 cannot be reduced in the same way and is appended to the FIFO list. Again the algorithm computes a minimum cardinality white vertex cut separating the largest black region (within V_1) from the vertex that is furthest from this region within V_1 (i.e., vertex 42). The vertex cut returned by the algorithm is $\{46, 49\}$ and its shores are $V_4 = \{41..45\}$ and $V_3 = V_1 \backslash (V_4 \cup \{46, 49\})$. Since V_4 contains only one black vertex and we have set *minNbBlack* to 2, it will not be decomposed and is not appended to the FIFO list. The shore V_3 will not be decomposed either because its proportion of black vertices is too high (*blackLimit* has been set to 40.0). Hence when the main algorithm terminates, the border is the set $\{20, 27, 40, 46, 49\}$ and the blocks are the three sets V_2, V_4, and V_3. Each of V_2 and V_4 can be reduced, that is, replaced by the complete graph on the union of its border and its set of black vertices.

The set V_3 cannot be reduced but we can apply the Y-Δ transformation to vertex 52 and its neighbors. Therefore the singleton $\{52\}$ becomes a block and its neighbors 48 and 51 become border nodes (40 is also a neighbor of 52 but was already a border node). Because of this transformation all the neighbors of 47 are border nodes, which means that the singleton $\{47\}$ is now also a block. Thus at the end of the improvement phase, the border (denoted by B_0 in Section 4) is the set $\{20, 27, 40, 46, 48, 49, 51\}$, while the remaining vertices are partitioned into the following blocks: $\{0, 1, 26, 37, 38, 39, 50, 53..57\}$, $\{47\}$, $\{52\}$, $\{41..45\}$, and $\{2..19, 21..25, 28..36\}$. The first and second blocks will not be reduced while the third, fourth, and fifth blocks will be. The resulting graph is pictured in Figure 5.

The basic algorithm is "greedy" in the sense that it chooses a good vertex cut as soon as it has found one. In that case the evaluation of the vertex cut relies on one of its shores only (the shore containing the target v). In theory, to find a "good" vertex cut, the basic algorithm can take time proportional to the product of the number of nodes and the total number of edges (note that finding a vertex cut whose size is bounded a priori takes time linear in the number of edges). The graphs on which we conducted experiments (see Section 6) are sparse, so that their number of edges is approximately 1.5 times their number of vertices. Therefore in pratice, the basic algorithm does not consume much time, especially in the case of networks that contain several small cardinality vertex cuts.

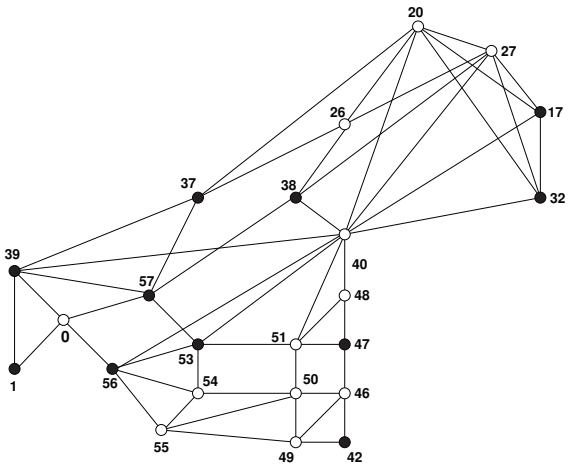

Figure 5: The reduced graph corresponding to the toy example

A procedure consuming more time (the *alternate algorithm*) computes a minimum cardinality vertex cut separating two given black regions (for all possible pairs of regions) and separates the current piece along the vertex cut affording the maximum reduction. To compute the latter, the alternate algorithm takes into account the reductions corresponding to all the shores of the vertex cut. In the case of our toy example, one has to compute 10 vertex cuts, one for each pair of black regions. For the toy example, this computation will still lead the algorithm to choose the vertex cut $\{20, 27, 40\}$ and the final partition of the vertex set will be the same as that found by the basic algorithm. For larger examples, however, the alternate algorithm may produce a better decomposition than the basic algorithm.

The basic algorithm can be modified in another way. In the above pseudocode, the FIFO list contains one set only (the set of all vertices) before the execution of the outer while loop. One can also compute the 2-connected components of the subgraph induced by the current set using a well-known linear time algorithm (see [20]). Then one can include into the FIFO list the 2-connected components that are separated from the rest of the graph by white cut-vertices. If a 2-connected component is separated from a neighboring component by a black cut-vertex, it is merged with its neighbor. This process is repeated until one obtains a union of 2-connected components that is separated from the rest of the graph by white vertices. This union is then appended to the FIFO list.

In the next section we report on experiments with four algorithmic variants:

1. the computation of 2-connected components followed by the basic algorithm,
2. the basic algorithm without pre-computation of 2-connected components (see the above pseudocode),
3. the computation of 2-connected components followed by the alternate algorithm, and
4. the alternate algorithm without pre-computation of the 2-connected components.

6 Experiments

To carry out our tests we used three networks supplied by the company NXP Semiconductors N.V. (see `www.nxp.com`). Actually each of these networks has many connected components. Network A (resp. B, C) has 14 (resp. 323, 101) components. The sizes of the components vary a lot, but generally speaking, one can make the following observations about them:

- the components are sparse, i.e., the average degree of a vertex is around three, and they actually contain many vertices of degree three,
- many of them contain grid-like structures; and
- almost all of them contain many small cardinality vertex cuts.

These characteristics of the components, especially the third one, led us to think that an algorithm based on the computation of vertex cuts would produce good reductions of the components. One of the components of Network A is depicted in Figure 2 (see Section 3).

Our intention was to compare the four variants mentioned at the end of the previous section, as well as a previously published algorithm (see the work of Lenaers [21], which is based on the article of Zečević and Šiljak [22]). We set the parameters of the algorithm as follows: $percentLimit = 80.0$, $minNbBlack = 5$, $blackLimit = 20.0$, $distBlack = 4$, $limitCut = 16$, and $factor = 10$. Note that these values are not exactly the same as those used for the toy example, because we had to make sure that the algorithm did not spend too much time on very large instances. Our main goal was to obtain reduced graphs with as few edges as possible, not to minimize the running time of the algorithm.

We can report that our algorithms are fairly robust, in the sense that the parameter values given above can be modified without introducing major changes in the results. The only exception to this observation concerns *percentLimit*: if its value is too high (say, equal to 90.0), the algorithm might not be able to find a "good" vertex cut even if the graph contains one. We can also report that the time consumed for finding the partition into blocks is generally significantly less than the time spent computing the resistances of the reduced network. We have also verified, for each network, that the solution of a system involving the reduced matrix was identical to the solution of the corresponding system involving the original matrix.

In Table 1 we present the results for Network A and its five largest components (each of the remaining components contains two vertices only). The last line of the table displays the results for the whole network, and the first five lines correspond to the largest components. The first three columns contain respectively the number of vertices, the number of black vertices, and the number of edges in the component or network. The fourth (resp. fifth) column contains the number of vertices (resp. edges) in the network produced by the algorithm of Lenaers. Similarly each of the following groups of columns gives the number of vertices and the number of edges in the network produced by an algorithmic variant. The number of edges produced by the algorithm of Lenaers is set in boldface type; so is the smallest number of edges produced by any of our algorithmic variants.

In Table 2 we give results for Network B and the 20 components of Network B that have more than 1000 vertices, and in Table 3, results for Network C and the 12 components of Network C that have more than 1000 vertices. Finally, in Table 4, we summarize the results for the three networks and express as percentages the number of vertices and the number of edges obtained by the algorithm of Lenaers and all the algorithmic variants. The examination of the four tables reveals that for almost all components, the number of edges produced by the algorithm of Lenaers is greater than the smallest number of edges produced by any of our algorithmic variants. In the few cases where this does not occur, the difference between the two numbers (set in boldface type in the tables) is small. We also note that the superior results of our algorithms are especially obvious in the case of large components.

Table 1: Results for Network A

Original network			Lenaers		Variant 1		Variant 2		Variant 3		Variant 4	
V	B	E	V	E	V	E	V	E	V	E	V	E
2300	59	3683	79	**709**	63	878	60	902	252	997	114	**637**
1542	39	2476	39	**741**	39	741	39	741	240	638	242	**633**
1210	76	1936	104	**1167**	157	823	150	849	157	**823**	151	850
341	28	555	40	**121**	83	110	83	**110**	83	110	83	110
244	54	333	236	**330**	150	214	147	**214**	150	214	147	215
5658	274	8997	516	**3077**	510	2775	497	2825	900	2791	755	**2454**

The first three columns contain respectively the number of vertices (V), the number of black vertices (B), and the number of edges (E) of the original network. The other columns contain the numbers of vertices and edges of the five reduced networks (note that the number of black vertices does not change). The number of edges of the component produced by the algorithm of Lenaers is in boldface type; so is the smallest number of edges produced by any of our algorithmic variants. The results for the whole network are displayed in the last line while the other lines contain the results for the largest components.

Table 2: Results for Network B

Original network			Lenaers		Variant 1		Variant 2		Variant 3		Variant 4	
V	B	E	V	E	V	E	V	E	V	E	V	E
16053	509	25594	989	**18100**	2303	**6373**	2095	6922	3045	10026	3606	9914
13168	430	21209	823	**13787**	3361	9407	3090	**9029**	3243	9212	3130	9131
12821	160	23222	952	**8017**	270	4937	270	4937	249	**4609**	249	4609
4836	123	7877	294	**3026**	374	2332	411	**2063**	307	2324	590	2584
3526	18	5776	18	**153**	18	153	18	153	18	153	20	**143**
3102	93	5105	170	**2631**	1029	3031	1019	2965	190	**2794**	709	4012
2580	72	4155	168	**2212**	373	1153	345	**1142**	504	1585	500	1494
2549	74	4048	231	**1656**	292	1073	273	**1064**	706	1753	470	1240
2440	57	4009	57	**1596**	255	1180	171	**1055**	115	1511	101	1359
2297	7	3526	8	**14**	13	**16**	7	21	13	16	7	21
2091	58	3430	136	**1199**	157	**774**	188	793	174	907	148	780
2040	54	3302	95	**1110**	115	834	142	**758**	109	1232	95	1218
1851	45	3082	45	**990**	145	719	148	**696**	105	822	45	990
1475	57	2430	94	**1351**	454	1296	426	1214	111	**1030**	57	1596
1436	45	2356	77	**873**	45	990	45	990	95	760	94	**738**
1381	60	2272	73	**1188**	118	1073	391	1167	117	**1033**	60	1770
1287	37	113	61	**657**	71	634	65	**519**	71	634	37	666
1095	33	1798	40	**522**	33	528	33	528	66	**462**	33	528
1059	42	1751	69	**706**	100	753	117	746	87	684	89	**674**
1030	44	1637	57	**145**	88	**129**	50	337	88	129	50	337
99112	3399	161183	6012	**62685**	11885	**40768**	10758	42399	11684	45059	11544	49104

Table 3: Results for Network C

Original network			Lenaers		Variant 1		Variant 2		Variant 3		Variant 4	
V	B	E	V	E	V	E	V	E	V	E	V	E
36393	197	58054	881	**8268**	1661	5127	614	**3263**	345	7866	596	4018
11783	309	18762	626	**7420**	1373	5300	1496	**4885**	2919	8039	2620	8313
10776	45	16551	63	**471**	54	**521**	46	871	54	521	46	801
7632	184	12222	530	**6239**	2695	7605	2674	7027	388	**4997**	3371	9353
5471	107	8559	304	**2837**	434	2132	239	**1576**	1152	2841	683	2032
4657	96	7459	96	**4560**	944	2635	579	**2084**	835	2399	857	2438
3694	52	5891	52	**1326**	52	1326	52	1326	114	1211	94	**1199**
3394	89	5475	161	**2198**	799	2497	1593	3758	201	2714	229	**2442**
3285	87	5291	103	**2360**	192	2366	988	2975	177	2099	203	**1897**
2964	17	4483	18	**93**	20	**55**	18	81	20	55	18	81
2961	67	4834	102	**1689**	156	1883	195	1394	140	**1187**	175	1243
1157	5	2176	5	**10**	5	10	5	**10**	5	10	5	10
103549	1978	164213	3880	**39011**	9715	33311	9536	**31459**	7680	35793	9929	36091

Table 4: Summary of results in absolute and relative terms

	Lenaers		Variant 1		Variant 2		Variant 3		Variant 4	
Network A	516	3077	510	2775	497	2825	900	2791	755	2454
%	9.12	34.20	9.01	30.84	8.78	31.40	15.91	31.02	13.34	27.28
Network B	6012	62685	11885	40768	10758	42399	11684	45059	11544	49104
%	6.07	38.89	11.99	25.29	10.85	26.30	11.79	27.96	11.65	30.46
Network C	3880	39011	9715	33311	9536	31459	7680	35793	9929	36091
%	3.75	23.76	9.38	20.29	9.21	19.16	7.42	21.80	9.59	21.98

The first, third, and fifth rows contain the numbers of vertices and edges of the five reduced networks. The second, fourth, and sixth rows contain these numbers expressed as percentages (with respect to the original network); for instance the number of vertices of the reduced network produced by the first algorithmic variant on Network A is 9.01% of the number of vertices of the original network.

Finally, we conducted preliminary experiments on Network D, a network with around 1.5 million resistors and 738 connected components. Only 26 of the components have more than 10000 vertices, and the proportion of black vertices within a component varies greatly. For 18 out of these 26 components, the number of black (external) vertices is at most 150 and a very large reduction (usually more than 90%) is obtained by replacing the component by the complete graph on its set of black (external) vertices. On the other hand, for 6 of the remaining components, the number of external vertices in each component exceeds 400 and, as a result, the components are difficult to reduce. We plan to continue our investigation of circuits that have a high proportion of external vertices and to improve our methods for these circuits.

7 Conclusions and future work

In this article we present algorithms that produce significant reductions in the size of purely resistive circuits while preserving their physical properties. The algorithms also produce better results than state-of-the-art methods, especially for large circuits. In particular our algorithms achieve reductions of almost 75% in the number of edges for two of the networks and a reduction of more than 80% for the third network. There is still scope for the design of better algorithms, however. In particular, it is possible that dynamic programming principles may enable one to design algorithms that produce even smaller models. We also plan to apply our present and future algorithms to other types of circuits, and even models from other disciplines of engineering. Finally, we plan to continue the study of the combinatorial problem that was formally defined in the article; in particular we hope to prove that it is NP-complete.

References

[1] J. Rommes and W. Schilders, "Efficient methods for large resistor networks," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 29, no. 1, 2010.

[2] R. Anderssen, P. Hjorth, A. Kane, P. Kitanov, K. Ladipo, O. Marcotte, B. Orser, S. Shontz, W. Sun, and B. Wane, "Model order reduction for electronic circuits: Mathematical and physical approaches, P.G. Hjorth and S.M. Shontz, eds." in *Proceedings of the 2nd Fields-MITACS Industrial Problem-Solving Workshop, to appear*, August 2008.

[3] A. Antoulas, *Approximation of Large-Scale Dynamical Systems*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2005.

[4] A. Odabasioglu, M. Celik, and L. Pileggi, "PRIMA: Passive reduced-order interconnect macromodeling algorithm," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 17, no. 8, pp. 645–654, 1998.

[5] B. Moore, "Principal component analysis in linear systems: Controllability, observability, and model reduction," *IEEE Trans. Autom. Control*, vol. AC-26, no. 1, pp. 17–32, 1981.

[6] J. Li, F. Wang, and J. White, "Efficient model reduction of interconnect via approximate system grammians," in *Proc. of the International Conference on Computer-Aided Design, San Jose, CA*, November 1999, pp. 380–383.

[7] W. Schilders, H. van der Vorst, and J. Rommes, *Model Order Reduction: Theory, Research Aspects and Applications*. Springer, Mathematics in Industry 13, 2008.

[8] P. Feldmann and F. Liu, "Sparse and efficient reduced order modeling of linear subcircuits with large number of terminals," in *Proc. of the International Conference on Computer-Aided Design, San Jose, CA*, 2004, pp. 88–92.

[9] H. Yu, L. He, and S. Tar, "Block structure preserving model order reduction," in *Proceedings of the 2005 IEEE International BMAS Workshop*, 2005, pp. 1–6.

[10] Z. Ye, D. Vasilyev, Z. Zhu, and J. Phillips, "Sparse implicit projection (SIP) for reduction of general many-terminal networks," in *Proceedings of the 2008 IEEE/ACM International Conference on ICCAD*, 2008, pp. 736–743.

[11] B. Sheehan, "Realizable reduction of RC networks," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 26, pp. 1393–1407, 2007.

[12] P. Elias and N. van der Meijs, "Extracting circuit models for large RC interconnections that are accurate up to a predefined signal frequency," in *Proceedings of the 33rd Design Automation Conference*, June 1996, pp. 764–769.

[13] K. Kerns and A. Yang, "Stable and efficient reduction of large, multiport RC networks by pole analysis via congruence transformations," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 16, pp. 734–744, 1997.

[14] ——, "Preservation of passivity during RLC network reduction via split congruence transformations," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 17, pp. 582–591, 1998.

[15] P. Harrison and M. Reeve, "The parallel graph reduction machine, ALICE," *Lecture Notes in Computer Science, Graph Reduction, Springer*, vol. 279, pp. 181–202, 1987.

[16] W. Sadiq and M. Orlowska, "Applying graph reduction techniques for identifying structural conflicts in process models," *Advance Information Systems Engineering, Lecture Notes in Computer Science, Springer*, vol. 1626, no. 2, pp. 195–209, 2009.

[17] ——, "Analyzing process models using graph reduction techniques," *Information Systems*, vol. 25, no. 2, pp. 117–134, 2000.

[18] B. Bollobas, *Modern Graph Theory*. Springer-Verlag, Graduate Texts in Mathematics, 1998.

[19] J. Gross, *Graph Theory and its Applications*. CRC Press, 1999.

[20] J. Bondy and U. Murty, *Graph Theory*. Springer-Verlag, Graduate Texts in Mathematics, 2008.

[21] P. Lenaers, "Model order reduction for large resistive networks," Eindhoven University of Technology, Tech. Rep., 2008.

[22] A.I. Zečević and D.D. Šiljak, "Balanced Decompositions of Sparse Systems for Multilevel Parallel Processing," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 41, no. 3, pp. 220–233, 1994.