Delayed Frontal Solution for Finite-Element based Resistance Extraction

N.P. van der Meijs and A.J. van Genderen
Delft University of Technology
Department of Electrical Engineering
Mekelweg 4, 2628 CD Delft, The Netherlands
email: nick@cas.et.tudelft.nl

Abstract

To save memory, layout-to-circuit extractors that use the Finite-Element Method for resistance extraction usually solve the corresponding set of equations with a frontal solution method. We show that this method is inefficient when used with a scanline ordering of the elements. As an improvement, we introduce the *Delayed Frontal Solution* algorithm, which allows us to replace the scanline ordering by the minimum-degree ordering. Thus, extraction times are reduced with more than one order of magnitude at a small cost of extra memory.

1 Introduction

The values of the interconnect resistances in modern integrated circuits tend to increase. This is caused by the fact that, due to down-scaling, the widths of the wires are decreased. Also, due to the increase in chip size, the average length of the wires increases.

Significant values for interconnect resistances in integrated circuits may cause malfunctioning of the circuit, e.g. because of too large delay time values or because of too great a drop in voltage along supply lines. Therefore, it is important to compute the values of the interconnect resistances from the layout of the circuit, so that the behavior of the circuit can be verified before it is fabricated.

Common approaches to the extraction of VLSI interconnect resistances include the Finite-Element Method (FEM) [1, 2, 3] and the Finite-Difference Method (FDM) [4]. These methods solve the resistance extraction problem by discretizing the governing differential equation (i.e. the Laplace equation) and solving the resulting set of algebraic equations. This set of equations is sparse, symmetric and positive definite, and is usually solved by Gaussian elimination [2, 3, 5].

32nd ACM/IEEE Design Automation Conference ®

When compared to other methods for resistance extraction, such as Polygonal Decomposition (as in [6]), Conformal Transformation (as in [7]) and the Boundary-Element Method (BEM) (as in [8]), the advantages of the FEM include general applicability, robustness, good accuracy and the possibility of accurately extracting RC models [2, 9]. Disadvantages, however, are those of longer computation times and higher memory requirements.

Thus, significant improvements of the efficiency of the FEM will be valuable. This paper will present such an improvement, which is based on optimizing the order in which the Gaussian elimination steps are applied. Since the FDM is a special case of the FEM, this paper will discuss the FEM only but the results will also be valid for the FDM.

Although there exist good heuristic ordering methods [5], most notably the minimum degree heuristic [10, 11], it is a problem to apply these heuristics in a circuit extraction context. In order to minimize storage requirements, a circuit extractor usually solves the equations 'on the fly' in a so-called frontal solution method. There is no freedom to change the elimination order, because this is fixed by the extraction method (often a scanline method).

This paper shows that the frontal solution method can be combined with the minimum-degree heuristic (or other, similar, heuristics) by reserving extra storage for the variables/nodes of the set of equations. It gives a detailed description of the proposed *Delayed Frontal Solution* algorithm. Combined with the minimum-degree heuristic, this algorithm improves the overall time– and/or space–efficiency by producing fewer fill-ins, and is compatible with a typical scanline implementation of extraction. Particular results include the following:

- A small amount of extra node memory can provide a 2-times speedup while simultaneously *reducing* the total amount of memory that is needed.
- A larger amount of extra node memory will increase the total amount of memory but can provide a 10-times speedup.

The rest of this paper is structured as follows. Section 2 presents a background on the FEM and a typical implementation of it in a circuit extractor. Section 3 discusses the

properties of the frontal solution method and the minimum-degree heuristic, and describes the new Delayed Frontal Solution algorithm. Section 4 presents an implementation in a layout-to-circuit extractor and a comparison with the standard frontal solution method. Section 5 presents the conclusions and additional remarks.

2 Background

2.1 Matrix Formulation of the FEM

For the purpose of resistance extraction, the finite-element method can be cast into a modeling method that directly produces an equivalent circuit model instead of a specific field solution. Thus, the result is an admittance matrix $\mathbf{Y}$ that relates the terminal currents to the terminal potentials. The terminals are formed by the pins and the connections of the wires to the active devices.

Without going into detail (but see, e.g. [1, 12]), we state that the finite-element method discretizes the interior of the interconnections and produces a system of equations that can be formulated as a matrix problem as in Equation (1):

$$\mathbf{A}\,\mathbf{x} = \mathbf{b}. \tag{1}$$

Here, $\mathbf{A}$ is a symmetric, positive definite and sparse matrix, $\mathbf{x}$ is a vector of unknown potentials and $\mathbf{b}$ is a vector of currents fixed by the boundary conditions.

If $\mathbf{A}$, $\mathbf{x}$ and $\mathbf{b}$ are partitioned into parts associated with the terminals (subscript t) and the internal variables (subscript i), Equation (1) may be written as

$$\begin{bmatrix} \mathbf{A}_{tt} & \mathbf{A}_{it} \\ \mathbf{A}_{ti} & \mathbf{A}_{ii} \end{bmatrix} \begin{bmatrix} \mathbf{x}_t \\ \mathbf{x}_i \end{bmatrix} = \begin{bmatrix} \mathbf{b}_t \\ \mathbf{b}_i \end{bmatrix} \tag{2}$$

where $\mathbf{b}_i$ is zero and it is not required to know $\mathbf{x}_i$ because we seek a compact relation between the terminal currents and voltages only. Hence, we can eliminate $\mathbf{x}_i$ and obtain

$$\mathbf{x}_t = \left[\mathbf{A}_{tt} - \mathbf{A}_{it}\mathbf{A}_{ii}^{-1}\mathbf{A}_{ti}\right]\mathbf{b}_t = \mathbf{Y}\ \mathbf{b}_t. \tag{3}$$

This solution procedure is in fact equivalent to Gaussian elimination.

From the admittance matrix $\mathbf{Y}$, we can directly create an admittance network where an admittance between nodes i and j has a value $G_{ij} = -Y_{ij}$. Thus, we directly obtain an equivalent circuit model of the layout, without solving a set of field problems.

2.2 Graph Formulation of the FEM

Instead of solving the system of equations and determining a resistance network only afterwards, from the resulting matrix $\mathbf{Y}$, we can begin with a circuit formulation of Equation (1). Such an interpretation of the FEM [1, 2, 12] initially produces a complex resistance network that models the resistive interconnections in detail. Subsequently, this network is reduced by a set of node eliminations. Each node elimination is in fact a Gaussian elimination step. If G_{ij}^k is the admittance between node i and j before the elimination of node k, and G_{ij}^{k+1} after the elimination of node k (we assume that node k is eliminated in step k), we have

$$G_{ij}^{k+1} = G_{ij}^k + \frac{G_{ik}^k G_{jk}^k}{\sum_{x \neq k} G_{kx}^k}. \tag{4}$$

Equation (4) states that the star network of the node that is eliminated is replaced by a full network (a *clique*) on the remaining nodes.

Layout-to-circuit extractors usually use this circuit circuit or graph formulation [2]. It is easier to implement efficiently and is compatible with the data structures for the device connectivity. This paper will therefore use this circuit context, but the results will also be valid for an array-based implementation.

2.3 Frontal Solution Method

Equation (4) shows that a node k can be eliminated as soon as all the admittances G_{kx} connected to node k are known. Doing this for each node can significantly reduce the amount of storage required (see the results in Section 4), when compared to first building the complete network and then doing the eliminations. Early elimination is in fact of paramount importance when large layouts must be handled.

The admittances connected to a node are all known when all the finite elements incident to that node have been processed (or assembled, in finite-element parlance). If the elements are processed systematically from one side of the layout to the other, the operations occur in a front that moves along the layout. Hence, this method is called the *frontal method*[13].

The frontal method can efficiently be implemented in a scanline based layout extractor [2]. Given the scanline processing sequence, the frontal method will result in the lowest possible number of network nodes that are in the core memory of the computer at any time.

3 Elimination Ordering

3.1 Standard Methods

The computational complexity of Gaussian elimination depends on the elimination order. For a discussion, we need the following notations:

N_t The number of terminal nodes.

N_i The number of internal nodes, usually $N_i \gg N_t$.

N The total number of nodes, $N = N_i + N_t$.

$d_k^{(0)}$ The degree (number of admittances connected to it) of node k in the original, uneliminated circuit.

d_k The elimination degree of node k; the number of admittances connected to it just prior to its elimination.

Without loss of generality, we assume that the nodes are eliminated in the order $1 \ldots N_i$.

The calculation of $\mathbf{Y}$ as in Equation (4) requires a total of $N_i = O(N)$ eliminations. Equation (4) shows that the cost of eliminating a node k is $O(d_k^2)$ (provided that an adequate graph data structure is used). If the circuit graph would be dense, $d_k = O(N)$ and the total complexity would become $O(N^3)$. This is in agreement with the matrix inversion in Equation (4). However, our circuit graphs are sparse and we can have $d_k < O(N)$.

Note, however, that in general d_k is not equal to $d_k^{(0)}$. It can be smaller, equal or (much) larger. This depends on the elimination order.

It is often assumed, see e.g. [2, 3], that the scanline based frontal elimination scheme provides a reasonably good ordering scheme. This is in fact an implicit ordering scheme, since it is determined by the scanline direction and the input data. However, it suffers from directional dependencies, as can be seen from the results in Section 4.2. Exploiting this dependency by rotating the input data is only helpful in a limited number of trivial cases and therefore not practical.

Furthermore, as is confirmed in Section 4.3, the method is not efficient in the case of large, rectangular regions with a fine discretization because of the generation of so-called *fill-ins*. This is, for example, the case with N-well resistance extraction.

In order to improve the running times for such unfavorable (but fairly common) layout geometries and discretizations, we must improve the order in which the nodes are eliminated.

Since the problem of computing an optimal elimination order is NP-complete [14], heuristic methods are needed. Indeed, many heuristics have been suggested (see [5] for a summary and comparison) and one of the best heuristics for the type of problems considered (i.e. symmetric, sparse and positive definite) is the minimum degree heuristic [10, 11]. With this heuristic, nodes are eliminated in order of ascending elimination degree d_k. Ties are broken arbitrarily.

The minimum degree heuristic can, for example, be used in a pre-processing phase to derive a good order for frontal processing. However, this approach is incompatible with a scanline-based extraction program that performs all extraction steps (including device recognition, connectivity extraction and parasitics extraction) in one pass over the layout data.

3.2 Delayed Frontal Solution

Since a pre-processing step for determining a good elimination order cannot be used, we need another way to combine the advantages of frontal solution (low memory requirements) with those of the minimum degree heuristic (low computation complexity).

A discussion of our solution needs the following preliminaries:

- A *conductor* is a collection of resistively connected nodes (a connected component in the circuit graph).
- We call a node *ready* when all its admittances are known, and we call a conductor ready when all its nodes are ready.
- The *degree* (not the elimination degree) of a node is the number of admittances connected to it. The degree may change during the elimination process.
- A *priority queue* is a data structure over an ordered set of elements that supports the following operations:

insert	Insert a new item into the queue.
deleteMin	Delete and return the item which is first in the order.
changeOrder	Change the ordering-key of an item.

We also need a delete operation:

delete	Delete and return a specified item.

Our method now combines the frontal solution method with the minimum-degree heuristic using the *Delayed Frontal Solution* algorithm as presented in Algorithm 1.

Algorithm 1 (Delayed Frontal Solution) The algorithm requires two priority queues, Q_1 and Q_2, that will contain the nodes ordered by their degree. Q_1 and Q_2 have a maximum capacity $|Q_1|_{max} = |Q_2|_{max} = Q_{max}$, assumed fixed for the moment.

1. **[Node ready]** If a node is ready, insert the node into Q_1. Do not yet eliminate the node.
2. **[Queue full]** If, after insertion, Q_1 is full, delete the node with the lowest degree from Q_1 in order to make room for the next insertion. Also, eliminate the node from the admittance network. As a result of this elimination, the degree of other nodes still in Q_1 may change. Execute the corresponding **changeOrder** operations for the affected nodes.
3. **[Conductor ready]** When a conductor becomes ready, first delete all nodes (if any) of that conductor from Q_1 and insert them into Q_2. Then, delete them from Q_2 in minimum-degree order and eliminate them from the admittance network (also execute the **changeOrder** operations) until Q_2 is empty. ■

The description of Algorithm 1 needs the following additional remarks.

- Step 3 is not strictly necessary, it can be replaced by empty-ing the queue at the end of the extraction. However, it reduces memory requirements without a negative effect on the minimum-degree order because the elimination of these nodes cannot alter the degree of the nodes from the other conductors.

- Q_{max} is a parameter of the method that can be used to trade memory for CPU time. When $Q_{max} \leq 1$, the method reduces to the frontal solution method and when $Q_{max} = \infty$, the method reduces to the minimum-degree method. Section 4 shows that relatively small values of Q_{max} can already significantly improve the efficiency.

- A sophisticated implementation of Delayed Frontal Solution may make Q_{max} adaptive: an absolute maximum amount of memory available (possibly an OS/hardware limit) minus what is necessary for other data structures. In that case, whenever the memory allocator would fail, it tries to obtain some memory by eliminating nodes (in minimum-degree order). Such an implementation is especially feasible if only the nodes are dynamically allocated or if all other allocations are of (approximately) the same size.

- Because the ordering priorities for Q_1 and Q_2 are positive integer numbers, a simple priority queue implementation using an array (indexed by the degree) of doubly linked lists (each list containing all nodes with the same degree) is adequate (efficient). Moreover, Q_1 and Q_2 may then be merged by keeping the elements of Q_2 in the front of each, shared, doubly linked list.

4 Results

4.1 Implementation

The data in this section are obtained from an actual implementation of the method in the Space layout-to-circuit extractor [15, 16]. Space is a scanline based layout-to-extractor with a vertical scanline that sweeps over the layout from left to right. All extraction operations are performed in one scanline pass over the layout. These operations include device recognition, connectivity extraction, capacitance extraction and finite-element based resistance extraction. For resistance extraction, the finite element mesh is created while scanning, the nodes are eliminated when they leave the queues and the memory is reclaimed immediately.

The computation times times and storage requirements reported in this section are 'all-in'. They include, for example, program start-up time and input-output time. The experiments were done on an HP-9000/735 computer.

4.2 Spiral Resistor

Consider the extraction of the spiral resistor as shown in Figure 1. When using the frontal method in a scanline based layout-to-circuit extractor, the computation times for the orientation shown will be much larger than for the same resistor but rotated over 90 degrees.

The extraction results are summarized in Table 1. For $Q_{max} = 0$, the method is the standard, frontal method. The results show a significant directional dependency: The extraction times are 9.1 and 1.0 seconds for the unrotated and the rotated resistor, respectively. On the other hand, the minimum-degree ordering does not suffer from such directional dependencies. This is reflected by the case $Q_{max} = \infty$. For large values of Q_{max}, the computation times are the lowest but this is at the cost of memory requirements.

These data also show that a small number of extra node storage already provides a significant improvement. The computation times decrease and become less dependent on the orientation. In addition, the total memory that is required is decreased.

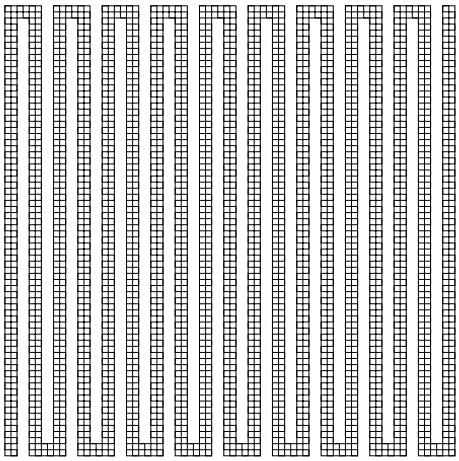

Figure 1: Layout and discretization of a spiral resistor.

4.3 Standard Cell Chip

The results in this section refer to a part of a standard cell chip in CMOS. It is taken from the top-right corner of a large chip, including the pad ring. The core-part contains 1753 transistors. For this high-performance industrial design, it was required to know the N-well resistances. When extracting poly, active layer and N-well resistances, the standard frontal elimination method results in long computation times. This is caused by the shape of the N-well regions. Delayed frontal elimination, however, works very well. Also for the case of only poly and active layer resistance extraction, Delayed Frontal Solution gives a significant speedup. These results are summarized in Table 2.

Table 1: Extraction results of the spiral resistor. The elimination cost is defined as $\sum d_k^2$ over all nodes.

Q_{max}	0		2		5		100		∞	
rotation (degrees)	0	90	0	90	0	90	0	90	0	90
elimination cost/1000	8894	493	1010	206	355	178	155	163	74	74
maximum degree d_k	77	68	36	31	20	15	9	7	4	4
# nodes in core	153	116	153	118	158	119	285	819	4336	4336
# resistances in core	3005	2348	1042	852	566	459	759	551	7231	7231
cpu time (sec)	9.1	1.0	1.6	0.7	0.8	0.6	0.6	0.6	0.6	0.6
total memory (kbyte)	429	389	322	289	285	265	305	353	1157	1157

Table 2: Extraction results of the standard cell chip. The elimination cost is defined as $\sum d_k^2$ over all nodes. Without resistances, the cpu time is 3.9 sec and the memory is 0.661 Mbyte.

	with N-well				without N-well			
Q_{max}	0	1000	5000	∞	0	1000	5000	∞
elimination cost/1000	811238	76178	42332	16563	22019	3515	2932	2792
maximum degree d_k	214	165	162	156	151	151	151	151
cpu time (min:sec)	20:58.1	2:11.6	1:30.6	59.0	48.5	27.3	27.9	29.2
total memory (Mbyte)	4.06	3.88	8.53	25.6	2.24	2.36	3.17	9.22

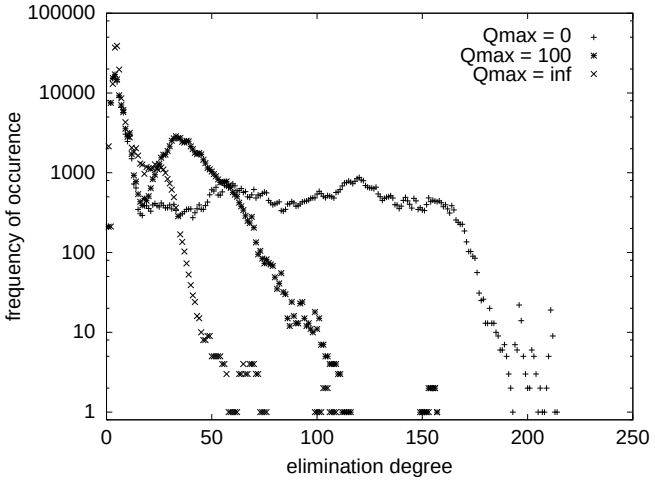

Figure 2: Elimination degree frequency for the standard-cell chip for three different values of Q_{max}, with N-well resistance extraction.

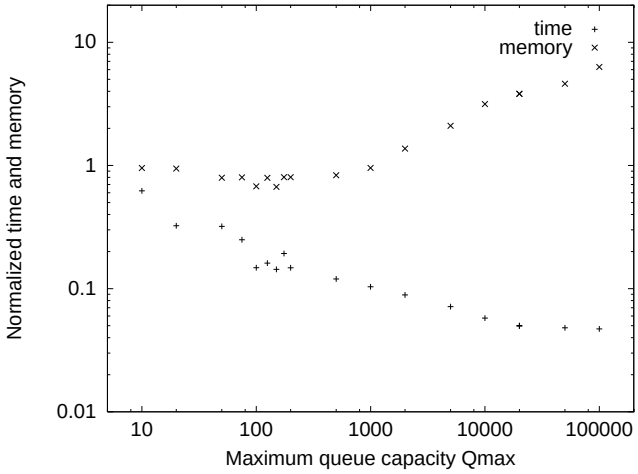

Figure 3: Normalized computation time and memory, as a function of Q_{max}, for the standard cell design and the extraction of poly, active layer and well resistance.

Figure 2 shows the elimination degree frequency, the number of times that a node with a certain degree is eliminated, with Q_{max} as a parameter, for the case of N-well resistance extraction. This figure can help to explain the speed-improvement, when considering that the cost of eliminating a node with degree d_k is $O(d_k^2)$.

Figure 3 shows the effect of varying Q_{max}. It displays normalized computation times and memory requirements as a function of Q_{max}. Normalization is with respect to the case $Q_{max} = 0$. The curves are not smooth because the minimum-degree method is a *heuristic* method. Thus, although elimination orders for a larger Q_{max} are actually 'not less minimum degree' than those for a smaller Q_{max}, they can be less optimal. Figure 3 also shows that, for this example, a Q_{max} around $1000 \sim 5000$ does not significantly increase the memory requirements but already provides a 10 times speedup.

5 Conclusion and Additional Remarks

We have shown that the frontal method for Gaussian elimination is inefficient when used with a scanline ordering of the elements. As an improvement, we have introduced the *Delayed Frontal Solution* algorithm, that allows to use the minimum-degree ordering heuristic. When combined with this heuristic, Delayed Frontal Solution can easily provide one or two orders of magnitude speed-up. The running times become much more linear with the problem size and will not depend strongly on the geometric structure and the discretization of the problem.

Delayed frontal solution needs extra storage for the nodes, and it can require extra total memory. The amount of extra memory depends on a parameter Q_{max} of the method. With small values of Q_{max}, the total amount of memory required can actually be less than what would be required with frontal solution while a significant speed-up is still achieved. In practice, a good default value for Q_{max} is around $1000 \sim 5000$. This provides a significant speedup at a small cost.

We can make the following additional remarks:

• For the accurate extraction of RC models that include coupling capacitances, [17] has introduced a method where the capacitances 'go along' in the Gaussian elimination process and the resulting reduced RC models have a correct first order time constant. In this method, the elimination degrees of the nodes will be much higher than in the case of only resistance extraction. Hence, the algorithm of this paper will have an even greater effect on the performance.

• Although the discussion of Delayed Frontal Solution took place in the context of a graph formulation, the ideas are also valid for a matrix formulation of the problem.

• Instead of the minimum-degree heuristic, other heuristics such as the minimum local fill-in heuristic (see, e.g. [5]), can also be used.

References

[1] T. Mitsuhashi and K. Yoshida, "A Resistance Calculation Algorithm and its Application to Circuit Extraction," *IEEE Trans. CAD*, vol. CAD-6, no. 3, pp. 337–345, 1987.

[2] A. J. van Genderen and N. P. van der Meijs, "Extracting Simple But Accurate RC Models for VLSI Interconnect," in *Proc. ISCAS-88*, (Helsinki, Finland), pp. 2351–2354, June 1988.

[3] D. Stark, "Analysis of Power Supply Networks in VLSI Circuits," Tech. Rep. WRL-TR-91-3, Digital Equipment Corporation, Western Research Laboratory, 1991.

[4] S.P. McCormick, "EXCL: A circuit extractor for IC designs," in *Proc. 21st Design Automation Conference*, pp. 616–623, 1984.

[5] I.S. Duff, A.M. Erisman, and J.K. Reid, *Direct Methods for Sparse Matrices*. Oxford Science Publications, 1986.

[6] M. Horowitz and R. W. Dutton, "Resistance extraction from mask layout data," *IEEE Trans. on CAD*, vol. CAD-2, pp. 145–150, July 1983.

[7] P.M. Hall, "Resistance Calculation for Thin Film Patterns," *Thin Solid Films*, vol. 1, pp. 277–295, 1967/1968.

[8] B.R. Chawla and H.K. Gummel, "A Boundary Technique for Calculation of Distributed Resistance," *IEEE Trans. Electron Devices*, vol. ED-17, pp. 915–25, 1970.

[9] M. G. Harbour and J. M. Drake, "Calculation of Signal Delay in Integrated Interconnections," *IEEE Trans. on Circuits and Systems*, vol. CAS-36, pp. 272–276, Feb. 1989.

[10] W.F. Tinney and J.W. Walker, "Direct Solutions of Sparse Network Equations by Optimally Ordered Triangular Factorization," *Proc. IEEE*, vol. 55, pp. 1801–1809, 1967.

[11] H.M. Markowitz, "The Elimination Form of the Inverse and its Application to Linear Programming," *Managment Sci.*, vol. 3, pp. 255–269, 1957.

[12] A. J. van Genderen, *Reduced Models for the Behavior of VLSI Circuits*. PhD thesis, Delft University of Technology, Delft, the Netherlands, Oct. 1991.

[13] B.M. Irons, "A Frontal Solution Scheme for Finite Element Analysis," *Numer. Meth. Engng*, vol. 2, pp. 5–32, 1970.

[14] M. Yannakis, "Computing the Minimum Fill-In is NP-Complete," *Siam J. Alg. Disc. Meth.*, vol. 2, pp. 77–79, 1981.

[15] N. P. van der Meijs and A. J. van Genderen, "Space User's Manual," Tech. Rep. ET-NT 92.21, Delft University of Technology, Dept of EE, Delft, NL, Apr. 1992.

[16] N. P. van der Meijs and A. J. van Genderen, "Space-Efficient Extraction Algorithms," in *Proc. IEEE 3rd European Design Automation Conference*, (Brussels, Belgium), pp. 520–524, Mar. 1992.

[17] A. J. van Genderen and N. P. van der Meijs, "Reduced RC Models for IC Interconnections with Coupling Capacitances," in *Proc. IEEE 3rd European Design Automation Conference*, (Brussels, Belgium), pp. 132–136, Mar. 1992.