RC-In-RC-Out Model Order Reduction Via Node Merging

Manar Abdel-Galil [1] , Hazem Hegazy [2] , Yehea Ismail [1] , *Fellow*, *IEEE*
1. Center for Nano-electronics & Devices, American University in Cairo &Zewail City for Science and Technology, Cairo, Egypt
2. Mentor Graphics, Inc., Cairo, Egypt
Email: manar.mansour@aucegypt.edu , y.ismail@aucegypt.edu

***Abstract*— In this paper, we introduce a method for realizable reduction of extracted *RC* netlists by merging nodes. This method can achieve high reduction (reaching 96%) with high accuracy and can be used to complement existing techniques of realizable reduction such as TICER [6]. The method preserves sparsity; has controllable accuracy and can result in lossless reduction (exact reduction) for certain circuits. The node merging translates to a simple matrix operation and thus can be easily adopted commercially and realized in CAD tools.**

Keywords*— *realizable reduction; model order reduction; RC circuit reduction; node merging; passive reduction ; sparse reduction ; sparsity preserving; lossless reduction .

I. INTRODUCTION

Following Moore's Law, the complexity of integrated circuits has nearly doubled every year. This trend requires engineers in industry to produce new sophisticated CAD tools keeping up with the increased complexity of circuits. A particular bottleneck in the verification flow of VLSI circuits is the simulation of extracted *RC* netlists which is memory and time consuming. As a result of this problem, since 1990, there has been many works introducing model order reduction (MOR) methods that could reduce the large extracted netlists into smaller ones with acceptable accuracy. However, the problem with the majority of those methods (e.g., [1] [2] [3] [4] [5]) is that they are not realizable .i.e. their output is a mathematical state space representation instead of a real *RC* circuit. This represents an issue when it comes to mainstream CAD tools in industry that are designed to deal with *RC* netlists and it is usually hard to attempt to modify the analysis flow of these tools. However, some realizable methods were introduced (e.g., [6] - [11]). The most important realizable method of reduction, which was introduced in 1999 and is still used up till now in industrial mainstream CAD tools, is TICER [6]. The advantage of TICER is that it could produce a realizable reduced circuit with good accuracy. However, it reduces the sparsity of the original circuit resulting in denser matrices which could reduce the benefit of the reduction.

In this paper, we present a realizable reduction method that is sparsity preserving and complementary to TICER. The method merges nodes and this corresponds to simple matrix operations as compared to the Y- Δ transformations used by TICER. Moreover, in some *RC* structures, Node Merging can achieve the same reduction as TICER but with less average error. Another advantage of Node Merging is that it can result in exact (lossless) reduction for circuits with high symmetry, *i.e.,* reduction without any sacrifice in accuracy. This option is not available in TICER where for any reduction made, no matter how small it might be, some accuracy has to be sacrificed. However, the Node Merging method described in this paper is not intended to replace TICER but rather to complement it. As will be shown in the experimental results, when both methods are integrated, better results are achieved as compared to using any of the two methods individually.

The organization of this paper is as follows. The Node Merging method is presented in section II. The experimental results are given in section III. and the paper is concluded in section IV.

II. NODE MERGING METHOD

A. Mathematical Formulation

Any *RC* circuit can be described by the state space equations given by (1) which follow from the general nodal admittance analysis.

$$G\,\underline{V} + C\,\underline{\dot{V}} = \underline{b}\,V_{in}, \qquad (1)$$

where *"G"* is an nxn admittance matrix, *"C"* is the nxn capacitance matrix, *"V"* is the $nx1$ voltage matrix of the internal nodes of the circuit, *"b"* is an nxp matrix that defines p inputs to the circuit and n is the number of internal nodes in the circuit excluding the ground node. To illustrate the basic idea, and without loss of generality, we will consider the example of a simple three node *RC* circuit (Fig. 1) which could be a part of any large scale *RC* network.

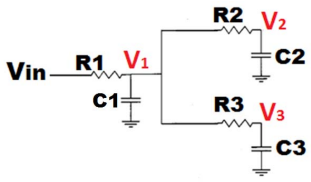

Fig. 1. *Three node RC circuit*

The nodal admittance representation for this circuit is given by:

$$G\begin{bmatrix}V_1\\V_2\\V_3\end{bmatrix} + C\begin{bmatrix}\dot{V}_1\\\dot{V}_2\\\dot{V}_3\end{bmatrix} = \begin{bmatrix}g_1V_{in}\\0\\0\end{bmatrix}, \qquad (2)$$

where $G = \begin{bmatrix} g_1 + g_2 + g_3 & -g_2 & -g_3 \\ -g_2 & g_2 & 0 \\ -g_3 & 0 & g_3 \end{bmatrix}$, $C = \begin{bmatrix} c_1 & 0 & 0 \\ 0 & c_2 & 0 \\ 0 & 0 & c_3 \end{bmatrix}$,

$g_1=1/R_1$, $g_2=1/R_2$ *and* $g_3=1/R_3$. (3)

B. Node Merging Idea

For the same three node *RC* circuit (in Fig. 1), if $R_2=R_3=R$ and $C_2=C_3=C$, then the circuit is symmetric in V_2 *and* V_3. Consequently, the response, the delay, and all the moments at node 2 are exactly the same as those at node 3. Therefore, we can short circuit node 2 and node 3 without changing the circuit. In this case, the two resistors are parallel and can be replaced by their equivalent resistor *R/2* and also the two capacitors are parallel and can be replaced by their equivalent capacitor *2C*. Thus, we arrive to the two node circuit as depicted in Fig. 2.

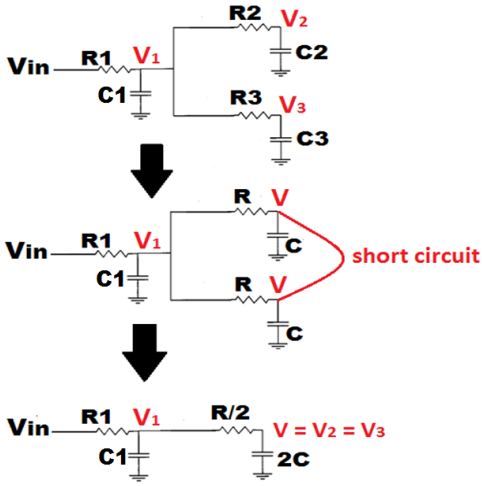

Fig. 2. *Merging two nodes*

Those intuitive steps in Fig. 2 correspond to the equivalent simple matrix operation of adding the two rows of the two merged nodes and placing the result in one of them while eliminating the other and the same is done for the two columns of the two merged nodes. In other words, to obtain the *"G"* and *"C"* matrices of the reduced two node circuit in Fig. 2, we apply the following operations on the matrices of the original three node circuit.

$$row_2 \leftarrow row_2 + row_3 \quad (4)$$

Next, we eliminate row_3 and after that, we apply:

$$column_2 \leftarrow column_2 + column_3 \quad (5)$$

Then we eliminate $column_3$.

By performing the previous operations on (2) and taking into account that $R_2=R_3=R$ and $C_2=C_3=C$, we arrive at the following matrices of the reduced circuit:

$$Gnew = \begin{bmatrix} g_1 + 2\,g & -2g \\ -2g & 2g \end{bmatrix}, \quad Cnew = \begin{bmatrix} c_1 & 0 \\ 0 & 2c \end{bmatrix} \quad (6)$$

If we realize the elements of the new matrices, we get the reduced two node circuit in Fig. 2. Hence, there is an equivalent simple matrix operation corresponding to the Node Merging method which facilitates the adoption and implementation of this method in the main stream CAD tools. In addition, the resulting circuit is always realizable because the merging corresponds to a short circuit between two nodes.

In the previous example, two nodes were merged resulting in one node that preserves the exact response of the original two nodes. This operation can be called "**Exact Node Merging**" or "**Lossless Node Merging**" which is a special case of the general Node Merging method. Lossless reduction gives this method a great advantage for certain types of circuits over TICER and other reduction techniques that don't exploit the presence of symmetries in the circuit.

In the general Node Merging method presented in the next section, we merge the nodes that have equal or approximately equal first moment m_1 (Elmore delay). The acceptable tolerance to compare m_1 of the merged nodes is set according to the user preferences.

C. The General Node Merging Algorithm

The algorithm takes as input the *"G"* and *"C"* matrices of the circuit to be reduced. It also takes a user selectable tolerance value corresponding to the maximum acceptable percentage error in the first moment m_1 .i.e. it gives the user the choice to determine how accurate he wants the reduction to be. The algorithm starts by sorting the first moment m_1 of all the internal nodes in the circuit. Then the nodes that differ from each other by the specified tolerance are grouped together and then merged. Next, for the resulting circuit, we calculate the first moment of the internal nodes and sorting is done again. The algorithm repeats and continues to reduce the circuit until there are no longer any two nodes that differ from each other by the specified tolerance.

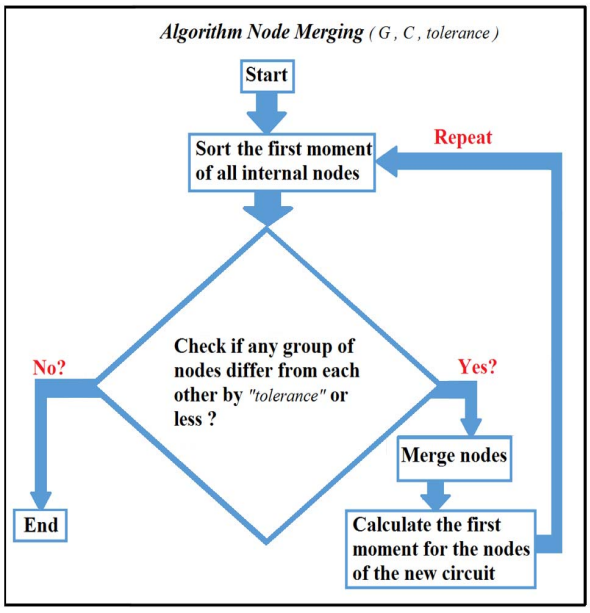

Fig. 3. *Node Merging Algorithm*

D. Qualitative Comparison to TICER

We applied both methods (Node Merging and TICER) to different *RC* networks and the following can be noted:

- For certain *RC* networks, Node Merging can reach higher reduction with little or no loss in accuracy as compared to TICER. This behavior is because Node Merging exploits the existence of symmetry in any *RC* network, while TICER does not. For example, TICER cannot discriminate the symmetry in Fig. 2. However, in some *RC* structures like *RC* lines, TICER achieves higher reduction with less error than Node Merging.
- Node Merging preserves the sparsity of the original matrices where the admittance and capacitance matrices of the resulting reduced circuit are also sparse. This occurs because merging nodes is always accompanied by eliminating elements, *i.e.*, Node Merging leads to decreasing the number of nodes and elements at the same time. On the other hand, TICER reduces the sparsity of the original circuit since it produces matrices that are dense compared to the sparse original matrices. This is evident because TICER uses the general Y- Δ transformations to eliminate nodes from the original circuit. Those transformations in their general form work by eliminating nodes at the expense of increasing the number of elements causing TICER to produce a reduced circuit that is relatively dense.

III. EXPERIMENTAL RESULTS

A. Node Merging Versus TICER

We implemented both Node Merging and TICER algorithms and applied them to the same *RC* circuits .The nodes retained by Node Merging after reduction are different from the nodes retained by TICER. For a fair comparison, we calculated the average error for all the nodes.

A comparison between the two methods for several test circuits having different numbers of nodes is given in Table I. We notice that for the same number of nodes, more reduction is achieved in case of Node Merging by increasing the tolerance used to compare m_1 of the merged nodes, while in TICER, the reduction increases by increasing the time constant used to decide which nodes to be eliminated by TICER algorithm. There is a clear tradeoff between the reduction achieved and the percentage error in the delay. Note that in case of Node Merging, the reduction in the number of elements is either equal to (e.g. in binary trees and the *RC* line) or greater than (e.g. in the *RC* mesh) the reduction in nodes, while in case of TICER, the reduction in elements is always less than the reduction in nodes except for the case of the *RC* line. This is expected since TICER reduces the sparsity of the original circuit, while Node Merging preserves it.

For a 63 node binary tree, Node Merging could achieve a reduction of 90.47% in nodes and elements with no error (lossless reduction) , while TICER achieves a lower reduction of 85.71% in nodes and 76.19% in elements with higher average and maximum errors of 3.81% and 25.32% respectively. For the same reduction of 92.06% in nodes , the average and maximum errors in case of Node Merging are 1.5% and 6.33% respectively, whereas in TICER, the average and maximum errors are 24.39% and 103.8% respectively .

For a 49 node *RC* mesh and for the same reduction of 73.46% in nodes, Node Merging achieves a higher reduction of 76.11% in elements as compared to 31.34% in case of TICER. Also, in this case, the average and maximum errors in Node Merging are 0.41% and 1.11% as compared to 2.6% and 30.54% in TICER.

However, for the 20 node *RC* line, TICER achieves a higher reduction of 45% in nodes and elements with less average and maximum errors of 1.09% and 7.69% respectively, whereas Node Merging achieves a lower reduction of 35% in nodes and elements with higher average and maximum errors of 4.35% and 10.2% respectively.

Therefore, Node Merging outperforms TICER in case of *RC* trees and meshes, while in case of *RC* lines, TICER is better.

TABLE I. NODE MERGING VERSUS TICER FOR DIFFERENT *RC* STRUCTURES

Circuit Type	#Nodes	#Elements	Node Merging					TICER				
			Tolerance (%)	Nodes Reduction (%)	Elements reduction (%)	Average error (%)	Max. error (%)	Time constant	Nodes Reduction (%)	Elements reduction (%)	Average error (%)	Max. error (%)
Binary tree	31	62	2	87.09	87.09	2.11	6.56	5	83.87	80.64	10.09	39.34
Binary tree	63	126	0.1	90.4	90.47	0	0	7	85.71	76.19	3.81	25.32
Binary tree	63	126	1.5	92.06	92.06	1.5	6.33	8	92.06	90.47	24.39	103.8
Binary tree	127	254	0.1	94.48	94.48	0	0	8	92.91	88.18	10.26	78
RC mesh	49	134	1	73.46	76.11	0.41	1.11	1	73.46	31.34	2.6	30.54
RC line	20	40	4	35	35	4.35	10.2	1	45	45	1.09	7.69

For the 63 node binary tree (in Table I) and at the same reduction of 92.0635% in nodes, the step response of both methods as compared to the original circuit is shown in Fig. 4. We notice that the step response of Node Merging coincides with that of the original circuit, while the step response of TICER differs slightly from the original response especially at the 50% delay point.

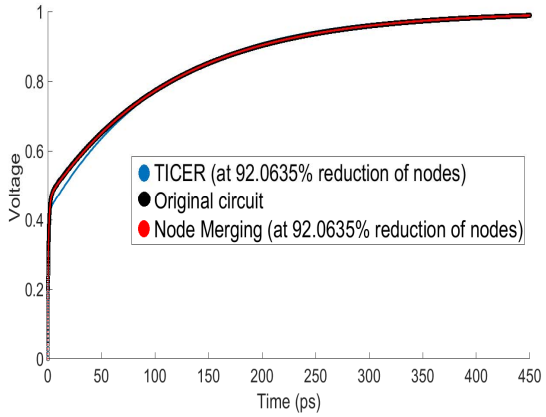

Fig. 4. *Node Merging versus TICER for a 63 node binary tree at the same reduction of 92.0635% in nodes.*

B. Applying Node Merging and TICER Simultaneously

So far we encountered cases where Node Merging outperforms TICER and other cases where TICER was better. This suggests the idea that if both methods are applied together, a much better performance can be reached as compared to the performance attained in case of using either method alone. This experiment was performed on a 63 node binary tree by applying Node Merging first and then applying TICER to further reduce the circuit obtained after Node Merging. Comparisons are given in Table II. It's obvious that using Node Merging and TICER together could achieve higher reduction of 96.82% with less average and maximum errors as compared to the situation of using TICER alone. Also we did the same experiment on an *RC* line of 20 nodes where Node Merging was first applied individually, and then Node Merging and TICER were applied together successively. The results in Table III show that using Node Merging and TICER together leads to a higher reduction of 50% in nodes with less average and maximum errors of 0.89% and 5.13% respectively as compared to the case of using Node Merging alone.

TABLE II. USING NODE MERGING AND TICER TOGETHER VERSUS USING TICER ALONE FOR A 63 NODE BINARY TREE

	TICER alone	**Node Merging & TICER**
Nodes reduction (%)	92.06	96.82
Average error (%)	24.39	13.22
Max. error (%)	103.8	25.32

TABLE III. USING NODE MERGING AND TICER TOGETHER VERSUS USING NODE MERGING ALONE FOR A 20 NODE *RC* LINE

	Node Merging alone	**Node Merging & TICER**
Nodes reduction (%)	35	50
Average error (%)	4.35	0.89
Max. error (%)	10.2	5.13

IV. CONCLUSION

A realizable sparsity preserving reduction method of *RC* circuits was introduced. This method works by merging nodes differing from each other in the first moment by a certain tolerance. It provides the options of controllable accuracy and lossless reduction. It also preserves the sparsity of the original circuit. For some networks, it could achieve higher reduction with less error compared to TICER. When it is integrated with TICER, we could attain better results than using TICER alone or Node Merging alone. Thus, the Node Merging method could be complementary to TICER.

REFERENCES

[1] Miguel Silveira, L.; Kamon, M.; Elfadel, I.; White, J., "A coordinate-transformed Arnoldi algorithm for generating guaranteed stable reduced-order models of RLC circuits," *IEEE/ACM International Conference on Computer-Aided Design ICCAD*, Nov. 1996, pp.288-294.

[2] Odabasioglu, A.; Celik, M.; Pileggi, L.T., "PRIMA: passive reduced-order interconnect macromodeling algorithm," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol.17, no.8, pp.645-654, Aug 1998.

[3] Zuochang Ye; Vasilyev, D.; Zhenhai Zhu; Phillips, J.R., "Sparse Implicit Projection (SIP) for reduction of general many-terminal networks," *IEEE/ACM International Conference on Computer-Aided Design ICCAD*, Nov. 2008, pp.736-743.

[4] Dumitriu, L.; Iordache, M.; Mandache, L., "Model order reduction by a projection technique implemented on state equations," *International Symposium on Signals, Circuits and Systems ISSCS*, July 2009, pp.1-4.

[5] Ismail, Y.I.; Friedman, E.G., "DTT: direct truncation of the transfer function - an alternative to moment matching for tree structured interconnect," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* vol.21, no.2, pp.131-144, Feb 2002.

[6] Sheehan, B.N., "TICER: Realizable reduction of extracted RC circuits," *IEEE/ACM International Conference on Computer-Aided Design,* Nov. 1999, pp.200-203.

[7] Ganesh, P.; Chen, C.C., "RC-in RC-out model order reduction accurate up to second order moments," *Proc. International Conference on Computer Design ICCD,* 2001, pp.505-506.

[8] Amin, C.S.; Chowdhury, M.H.; Ismail, Y.I., "Realizable reduction of interconnect circuits including self and mutual inductances," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol.24, no.2, pp.271-277, Feb. 2005.

[9] Miettinen, P.; Honkala, M.; Roos, J.; Valtonen, Martti, "PartMOR: Partitioning-Based Realizable Model-Order Reduction Method for RLC Circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol.30, no.3, pp.374-387, March 2011.

[10] Miettinen, P.; Honkala, M.; Valtonen, M.; Roos, J., "Realizable reduction of interconnect models with dense coupling," *European Conference on Circuit Theory and Design ECCTD,* Sept. 2013, pp.1-4.

[11] C. Amin, M. Chowdhury, and Y. Ismail, "Realizable RLCK circuit crunching," in Proc. Des. Autom. Conf., 2003, pp. 226–231.