Realizable RLCK Circuit Crunching

Chirayu S. Amin
ECE Dept., Northwestern Univ.
Evanston, IL, USA
c-amin@northwestern.edu

Masud H. Chowdhury
ECE Dept., Northwestern Univ.
Evanston, IL, USA
masud@ece.northwestern.edu

Yehea I. Ismail
ECE Dept., Northwestern Univ.
Evanston, IL, USA
ismail@ece.northwestern.edu

ABSTRACT

Reduction of an extracted netlist is an important pre-processing step for techniques such as model order reduction in the design and analysis of VLSI circuits. This paper describes a method for realizable reduction of extracted *RLCK* netlists by node elimination. The method is much faster than model order reduction techniques and hence is appropriate as a pre-processing step. The proposed method eliminates nodes with time constants below a user specified time constant. By giving the freedom to the user to select a critical point in the spectrum of nodal time constants, this method provides an option to make a tradeoff between accuracy and reduction. The proposed method preserves the dc characteristics and the first two moments at all nodes. It also recognizes and eliminates all the redundant inductances generated by the extraction tools. The proposed method naturally reduces to TICER [13] in the absence of any inductances.

Categories and Subject Descriptors

J.6 [**Computer-Aided Engineering**]: *Computer-aided design (CAD)*, B.7.2 [**Integrated Circuits**]: Design Aids – *Simulation.*

General Terms

Algorithms, Performance, Design, Reliability, Verification.

Keywords

passive, realizable, model order reduction, simulation, interconnect, crunching.

1. INTRODUCTION

With increasing frequencies and faster signal transistion times, on chip inductive effects are becoming increasingly important [1]-[2]. Consequently, many commercial and proprietary extraction tools, such as [3], generate *RLC* circuits for high performance designs. These circuits together with the non-linear drivers are then analyzed by fast timing simulators such as [4] or by interconnect-centric tools such as RICE [5] using linearized models of the drivers. Due to the large amount of data typically generated by extraction tools, significant run-time and memory issues affect the analysis tools. Therefore, in the past decade there has been a significant focus on model order reduction methods which attempt to model the extracted netlist by a smaller model with minimal loss in accuracy.

Starting with AWE [6], methods such as [7], [8] use moment matching - either explicitly or implicitly - and projection techniques to generate a low order approximation of the original circuit. More recently, PRIMA [9] modified these techniques to guarantee the passivity of the reduced circuit. Reducing an extracted netlist is an important step before the netlist is fed into tools such as AWE and PRIMA so that they run faster. In order to take advantage of the model order reduction techniques, it is important to have a realizable netlist crunching method, i.e. *RLCK* in – *RLCK* out feature.

Realizability is important because this avoids the modification of mainstream analysis tools to handle reduced state-space or transfer function representations [10], since these tools are usually geared towards reading circuit netlists. Moreover, some analysis tools such as circuit checkers can only accept inputs in terms of *RLCK* circuits. In addition to realizability, however, maintaining sparsity is also important when reducing a circuit with a large number of ports. Even though the original circuit is large, it is very sparse since each node is only connected to a few nodes. It is important to maintain sparsity since model order reduction techniques perform better on sparse circuits.

Circuit reduction approaches based on Gaussian elimination include [11]-[12]. A different approach based on Gaussian elimination - called TICER - was presented in [13] for *RC* circuits. By selectively removing the non-port nodes of the original circuit, a smaller, realizable, passive *RC* circuit is produced. A method to reduce *RLC* netlists with coupling-inductors was presented in [14] but it requires matrix inversion and long run-time. In this paper, we extend the TICER approach to reduce general *RLC* circuits including coupling-inductors using run-time in the order of the number of nodes in netlist. We also present a heuristic to control the sparsity of the reduced circuit. This is similar to the technique presented for *RC* circuits in [11]. The presented reduction algorithm is much faster than model order reduction techniques and can be used as a pre-processing step for these techniques. This paper is organized as follows. The underlying theory behind the proposed reduction method is discussed in section 2. In section 3, a high level algorithm for reducing *RLC* circuits is presented. Experimental results are provided in section 4. Finally, section 5 concludes the paper. The appendix describes the procedure to reduce *RLC* circuits with coupling-inductors.

2. THEORY

Consider a source free *RLC* circuit consisting of n nodes. The nodal voltages must satisfy the following equation in the s-domain:

DAC 2003, June 2-6, 2003, Anaheim, California, USA.

Acknowledgements: The authors thank Chandramouli Kashyap and Byron Krauter from IBM Corp., Austin, TX, for their valuable feedback and help in this research work.
This research has been supported in part by National Science Foundation CAREER Award #CCR-0237822, Intel gifts and grants, and the Semiconductor Research Corporation (SRC).

$$Y(s)V(s) = 0\ , \tag{1}$$

where $Y(s)$ is the n x n admittance matrix. Consider the i^{th} node of the circuit and its k neighbors as shown in Figure 1(a). The i^{th} row of (1) is given by:

$$Y_i V_i - y_1 V_1 - y_2 V_2 - \ldots - y_k V_k = 0 \tag{2}$$

$$\text{where } Y_i = \sum_{j=1}^{k} y_j\ .$$

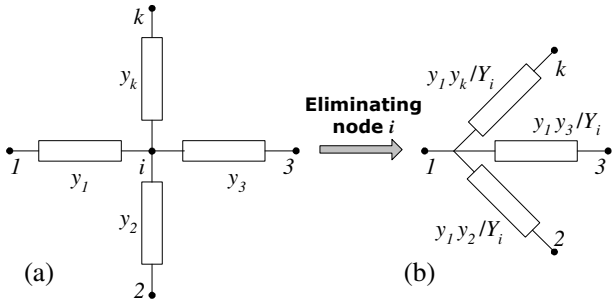

Figure 1. A general node in an *RLC* circuit. Admittances are added between node 1 and other neighbors of node *i* due to elimination of node *i*.

In order to eliminate V_i from the system (which is equivalent to eliminating the i^{th} node), we solve for V_i using (2):

$$V_i = \left(\sum_{j=1}^{k} y_j V_j \right) \Big/ Y_i \tag{3}$$

and substitute for V_i in the k equations where V_i occurs. Consider the first neighbor of i. Its equation is now given by:

$$(\hat{Y}_1 + y_1 - y_1^2 / Y_i) V_1 - \left(\sum_{j=2}^{k} y_1 y_j V_j \right) \Big/ Y_i - \sum_{\substack{r=1 \\ r \neq i}}^{k1} y_r V_r = 0 \tag{4}$$

where $\hat{Y}_1 = \sum_{\substack{r=1 \\ r \neq i}}^{k1} y_r$

where $\hat{Y}_1$ is the sum of all admittances from node 1 except to node i and k_1 is the number of nodes connected to node 1. The above equation can be simplified to

$$\left(\hat{Y}_1 + \left(\sum_{j=2}^{k} y_1 y_j \right) \Big/ Y_i \right) V_1 - \left(\sum_{j=2}^{k} y_1 y_j V_j \right) \Big/ Y_i - \sum_{\substack{r=1 \\ r \neq i}}^{k1} y_r V_r = 0 \tag{5}$$

Note that this is equivalent to adding k - 1 *new* elements between node 1 and the k -1 former neighbors of node i, (see Figure 1(b)). Specifically, for any two neighbors of node i, say m and n, the elimination of node i results in the addition of a new element between nodes m and n, whose admittance is given by:

$$y_{mn} = (y_m y_n)/Y_i\ . \tag{6}$$

Thus, repeating this process for all the k neighbors of i will result in the addition of $k(k\text{-}1)/2$ new elements. Note that the elimination of node i may introduce a *fill-in* in the original Y matrix of (1). For instance, referring to (6), if the $(m,n)^{th}$ entry in Y was a zero, *i.e.*, no element existed connecting m and n, elimination of node i would produce a fill-in in the $(m,n)^{th}$ entry of Y. In general, the formula for computing the fill-in produced by eliminating node i, ϕ_i, with k neighbors is given by

$$\phi_i = k(k-1)/2 - k - p\ . \tag{7}$$

where p is the number of elements connecting the neighbors of i in the original system.

Note that the new admittance given by (6) will be a polynomial in s. Without loss of generality, we assume that each admittance connecting a pair of nodes in the original system in Figure 1 consists of R, L, and C connected as shown in Figure 2.

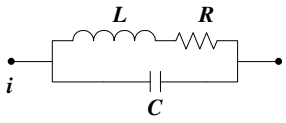

Figure 2. An admittance branch connected to node *i*

In most practical circuits, one or more of these elements will be zero. Note that this topology is general enough to handle any *RLC* circuit including coupling capacitances. If no inductance is incident on the node, then (6) reduces to the TICER case and we proceed as outlined in [13]. In that case (6) can be expressed by (8).

$$y_{mn} = (g_m + sc_m)(g_n + sc_n)/(G_i + sC_i) \tag{8}$$

where $G_i = \sum_{j=1}^{k} g_j$ is the sum of all conductances to node i, and $C_i = \sum_{j=1}^{k} c_j$ is the sum of all capacitances to node i. For pure LC case (6) can be expressed as

$$y_{mn} = \left(\frac{b_m}{s} + sc_m \right) \left(\frac{b_n}{s} + sc_n \right) / \left(\frac{B_i}{s} + sC_i \right) \tag{9}$$

where $B_i = \sum_{j=1}^{k} b_j$ is the sum of all susceptances (reciprocal inductances) connected to node i.

For every node in the circuit two time constants are defined: the *RC* time constant given by $\tau_{RCi} = C_i / G_i$, and the *LC* time constant given by $\tau_{LCi} = \sqrt{C_i / B_i}$. The *nodal time cosntant* at a node i is given by:

$$\tau_i = \max(\tau_{RCi}, \tau_{LCi})\ . \tag{10}$$

A node i is said to be a *quick node* if:

$$\tau_i < \tau_{\min} = 2\pi / \omega_{\max}\ , \tag{11}$$

where τ_{min} is a user defined time constant below which a node is considered quick and depends on the maximum frequency of interest in the circuit, ω_{max}, as given above. The selection of τ_{min} is circuit specific and is discussed in sections 3 and 4. According to (10) and (11), both time constants of a quick node must be less than τ_{min}. Note that τ_{min} is proportional to $1/s_{max}$ as given by (11). Hence a quick node satisfies the following approximations: $sC_i < G_i$, $G_i < B_i/s$, and $sC_i < B_i/s$.

To eliminate a quick node with the above general *RLC* branch (Figure 2) two cases can be considered. In the first case, τ_{RCi} is much larger than τ_{LCi} and (8) can be used as in TICER [13]. With the quick node approximation ($sC_i < G_i$,), (8) can be expanded into Taylor series upto first order:

$$y_{mn} = \frac{g_m g_n}{G_i} + s \frac{c_m g_n + c_n g_m}{G_i} + \ldots\ldots\ldots\ . \tag{12}$$

Here the constant coefficient of (12) gives the required conductance to be inserted between node m and n to eliminate node i. The coefficient of s gives the capacitance value to be inserted. In the second case, τ_{LCi} is much larger than τ_{Rci}, (9) is used. Again with the quick node approximation ($sC_i < B_i/s$), (9) can be expanded into Taylor series:

$$y_{mn} = \frac{1}{s}\frac{b_m b_n}{B_i} + s\frac{c_m b_n + c_n b_m}{B_i} + \text{.....} \quad \text{....} \qquad (13)$$

The coefficient of *1/s* in (13) gives the required susceptance (reciprocal to inductance) to be inserted between nodes *m* and *n* to eliminate node *i*. The coefficient of *s* gives the capacitance value to be inserted. For the general case, when both *R* and *L* are present with *C* as in Figure 2, the rules for eliminating node *i* based on the equations (12) and (13) are derived in Figure 3. The values for the conductance and the susceptance come directly from the equations (12) and (13) respectively. But two expressions for capacitance are obtained from (12) and (13). If τ_{RCi} is larger than τ_{LCi} the expression for capacitance is taken from (12) and when τ_{LCi} is larger than τ_{RCi} the expression for capacitance is taken from (13). Therefore, all but one of the rules shown in Figure 3 for merging any two branches connected to a particular node *i* can be easily derived from (12) and (13). The exception is for the case when nodes *m* and *n* are connected to node *i* through capacitances. In that case the value of the capacitance to be inserted is $c_m c_n/C_i$, which is the series combination of the capacitances. Since only positive valued *RLC* elements are added during node elimination, the passivity of the reduced circuit is guaranteed by construction. In contrast to TICER, we do not consider the notion of *slow nodes.* This is because in practice the lowest frequency of interest is zero, *i.e.*, the DC operation of the circuit is also required. In the following section, a high level algorithm implementing these ideas is presented. The appendix describes a method to reduce *RLC* netlists with coupling-inductors.

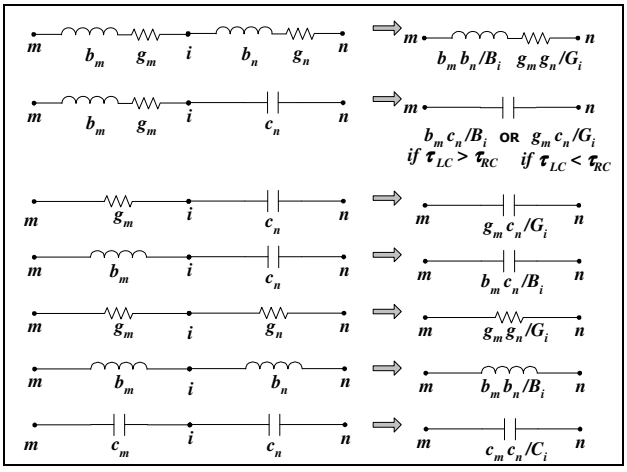

Figure 3. Rules for eliminating node *i*

It should be noted that the derivation of the above method of node elimination is based on two approximations, namely dominant τ_{RCi} and dominant τ_{LCi} approximations. Therefore, the correctness of the method is dependent on the validity of these two approximations. If for any node the two time constants (τ_{RCi} and τ_{LCi}) are far from each other the proposed method will work very effectively. If the two time constants are comparable (rarely happens in practice) then the reduction will introduce larger error. In such situation it is suggested not to eliminate that node if very low error is required. Therefore, to eliminate a node from the net list with less error two criteria have to be considered. First the node has to be a quick node, and second the two time constants should not be comparable. Tuning the threshold for these two conditions produces a tradeoff between accuracy and reduction.

3. REDUCTION ALGORITHM

The high level algorithm is shown in Figure 4. All the nodes in the original circuit are stored in a priority queue with the node with the smallest nodal time constant at the head of the queue. In addition to checking for the time constant of the node, we also check if the fill-in due to this node is less than some user specified threshold ϕ_{max}. If unspecified this threshold defaults to zero. This heuristic is added to ensure that the admittance matrix of the reduced circuit remains sparse. Of course, this heuristic may cause the algorithm to get stuck in a local optimum where even though a node's time constant may be less than $\tau_{min,}$ it may fail the fill-in check. However, for most interconnect circuits, such as those with a tree like topology, this algorithm will produce reasonably sparse reduced circuits. On the other hand, for very dense topologies where each node has several neighbors, the algorithm will perform poorly. While not explicitly shown in Figure 4, certain non-port nodes may also be marked as required by the user in which case these nodes cannot be eliminated. A check for this can be easily added to the algorithm.

CRUNCH_NETLIST (τ_{min}, ϕ_{max})

1. Place all non-port nodes of the circuit in a priority queue sorted by local nodal time-constants.
2. *i* = head of the priority queue
3. while (queue is not empty AND $\tau_i < \tau_{min}$ AND $\phi_i < \phi_{max}$)
4. set S = {neighbors of *i*}
5. if ($1/B_i == 0$)
6. eliminate *i* according to the TICER quick-node rules in [13]
7. else
8. eliminate *i* according to rules in Figure 3
9. update time-constants of nodes in set *S* (neighbors of *i*)
10. *i* = head of the priority queue

Figure 4. Circuit crunching algorithm

One natural question arises regarding the choice of τ_{min}. While signal transition times and operating frequencies do play a part in the choice, a histogram showing the distribution of the time constants can be very helpful. As we show in the next section, a large number of nodes typically have very small time constants. By choosing τ_{min} appropriately, these nodes can be eliminated with almost no loss in accuracy. A histogram helps in placing the time constants of all the nodes in the circuit in perspective as described in the next section.

4. RESULTS

We implemented the reduction algorithm in C++. Specifically, *p* was assumed to be zero in (7). We also required $\phi_{max} = 0$, which guarantees no refills by the reduction algorithm and maintains the sparsity of the *Y* matrix. Therefore, only nodes with a degree less than or equal to three were candidates for elimination. By applying the methods on some smaller industrial circuits (Table 1 and Figure 5) with less than 500 hundred nodes an average 50% reduction of circuit elements and nodes is obtained with less than 1% error in rise time and delay calculation. If the allowable margin of error is around 3% an average reduction of 55% can be achieved by selecting a higher τ_{min}. Higher error tolerance (around 5% and higher) will give even higher reduction in the range of 55 to 70%.

For symmetric and uniformly distributed *RLC* networks this method will give very high reduction. For a balanced and uniform H-tree network a reduction of about 90% is obtained with almost 0% error (Table 2 and Figure 6). With this amount of reduction a twenty-fold decrease in simulation time is obtained. Such a high reduction of the simulation time is due to the symmetric and uniform nature of the original and resultant reduced circuits.

Table 1: Small and Medium Industrial Circuits

	Small industrial circuit (less than 100 nodes)				Medium industrial circuit (several hundred nodes)			
	Original	Reduced circuit for different tau_{min} (ps)			Original	Reduced circuit for different tau_{min} (ps)		
tau_{min} (ps)	-	0.002	20	200	-	80	180	192
Nodes	81	55	43	29	227	158	147	127
Total Elements	152	105	79	54	526	290	271	241
% Reduction of nodes	-	32	47	64	-	30.39	35.24	44.1
% Reduction of elements	-	31	48	64.5	-	44.87	48.47	54.2
% Error in Rise Time	-	0.02	0.8	0.7	-	0.09	3.1	13.7
% Error in Delay	-	0.36	0.36	11	-	0.34	0.97	7.5

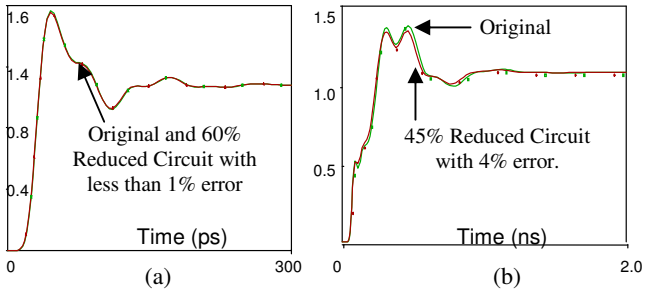

Figure 5. (a) Small circuit (b) Medium circuit

Table 2: Balanced and uniform H-tree networks.

	Original	Reduced circuit for different tau_{min} (ns)		
tau_{min} (ns)	-	0.1	0.5	1
Nodes	2502	666	416	128
Total Elements (R: 1250, L: 1250, C: 1250)	3750	997	617	190
% Reduction of nodes	-	73.4	83.4	94.9
% Reduction of elements	-	73.4	83.5	94.9
% Error in Rise Time	-	0.00	0.01	0.03
% Error in Delay	-	0.00	0.00	0.00

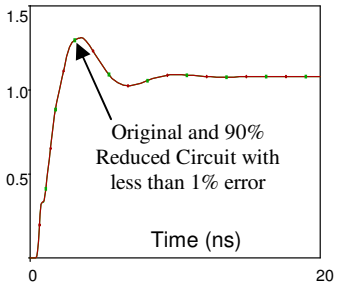

Figure 6. Sample signals from balanced and uniform H-tree network

The above small and medium sized circuits do not give clear idea of speed up (reduction of actual simulation time) due to nodes and elements reduction. With these objectives in mind the method is applied to an extracted clock distribution network from a commercial high-performance microprocessor. The extracted circuit contained over 678,608 elements and more than 10,000 sinks. A histogram of the distribution of the nodal time constants of this circuit up to a maximum of 100ps is shown in Figure 7. Based on this histogram, we made three choices of τ_{min} for circuit reduction: τ_{min}=15 ps, τ_{min}=25 ps, and τ_{min}=35ps. The results of running RICE (v5) on the original as well as the three reduced circuits are shown in Table 3 for an input rise time of 50ps. A fourth order approximation was computed at every sink. Clearly τ_{min}=15 ps offers the best choice resulting in over three times decrease in analysis time with almost no loss in accuracy. These reductions in run-time are expected to be much higher when nonlinear elements are combined with the interconnects. Note that the reduced *RLC* circuit can be readily inserted in any simulator such as SPICE or AS/X. As expected, choosing larger values of τ_{min} resulted in a loss of accuracy. Besides improving the analysis time, the reduction algorithm also reduced the storage requirements by one-third.

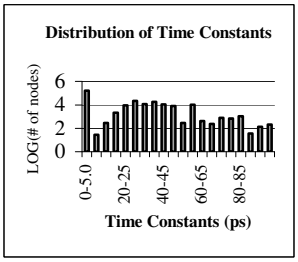

Figure 7. Distribution of time-constants for different nodes

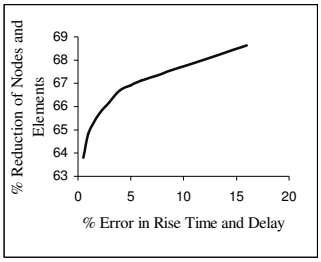

Figure 8. % Error vs. Reduction

Table 3. Reduced commercial circuit statistics

	Original	Reduced circuit for different tau_{min} (ps)		
tau_{min} (ps)	-	15	25	35
Nodes	380K	148K	141K	113K
Total Elements (R: 368K, L: 72K, C: 239K)	679K	285K	277K	239K
% Reduction of nodes	-	60.8	62.8	70.1
% Reduction of elements	-	57.9	59.2	64.6
% Error in Rise Time	-	0.05	-1.80	-11.1
% Error in Delay	-	0.25	10.5	21.7
Speedup	-	3.23	3.26	3.54

The average reduction at different levels of error in delay and rise time calculation is shown in Figure 8 from the reduction statistics of all the circuits presented above. It is observed that if the error is limited to 1% an average reduction of 60% is obtained. If the error is allowed to go up to 3% an average reduction of 66% is obtained. At 5% error, average 67% reduction can be achieved. It is observed from Figure 8 that beyond 67% reduction point, for a smaller gain in reduction a very high error is introduced. This is because at these amounts of reduction many critical circuit nodes and elements are thrown out of the netlists. Consequently the circuit behavior changes sufficiently and very high error in performance evaluation occurs.

The *RLCK* node elimination scheme (see Appendix) was applied to a 16-bit coupled bus with 18,000 elements (3200 *R*, 3200 *L*,

3200 *C*, 8400 *K*) including coupling-inductors and 9617 nodes. The circuit was reduced down by 93% with the new circuit having 189*R*, 205*C*, 189*L*, and 916*K*. A sample output for a middle bit is shown in Figure 9. The output signal for the reduced is almost identical to the signal in original circuit with 18,000 elements. Error in rise-time and delay was almost 0%.

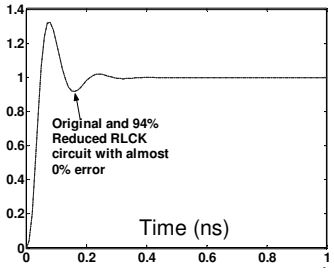

Figure 9. 16-bit coupled *RLC* bus with coupling-inductors

5. CONCLUSION

This paper presented a realizable reduction method of *RLC* circuits (with coupling-inductors) by nodal elimination. This method is useful in analyzing and verifying large *RLC* networks. If model order reduction is not realizable it produces reduced mathematical model of transfer function or reduced state equations. Hence all downstream circuit simulation and associated tests have to be modified to handle these reduced mathematical representations of the circuits. Since the proposed realizable reduction method is an *RLCK-in* to *RLCK-out*, all standard simulators can handle the reduced circuits without any modification. Reduced netlist also guarantees faster simulation, lower memory and storage requirements. It is shown that for an average of 40 to 50% reduction of nodes and elements error in waveform shape calculation is less than 1%. If 3% error is allowed a reduction of about 60 to 70% is obtained. Higher allowance of error results in an even higher reduction. Therefore, the user has the freedom to make the trade-off between accuracy and speed. Circuits with coupling-inductors are also reducible with the presented algorithm (see Appendix).

APPENDIX: *RLCK* NETLIST CRUNCHING

The circuit reduction scheme presented in section 2 can be extended easily to include *RLC* circuits with coupling-inductors (represented by *K*). Figure 10 shows the general node *i* of an *RLC* circuit with coupling-inductors. In Figure 10, each admittance has been decomposed into the generalized *RLC* branch form of Figure 2 and coupling-inductors are added to the *RL* branch.

The i^{th} row of equation (1) is now given by:

$$\begin{aligned} &Y_iV_i + sC_iV_i - y_1\left(V_1 - \sum_{x=1}^{1_a} sM_{1,x}I_{1,x}\right) - \\ &y_2\left(V_2 - \sum_{x=1}^{2_a} sM_{2,x}I_{2,x}\right) - \cdots - y_k\left(V_k - \sum_{x=1}^{k_a} sM_{k,x}I_{k,x}\right) - \\ &sc_1V_1 - sc_2V_2 - \cdots sc_kV_k = 0 \\ &where\ Y_i = \sum_{j=1}^{k} y_j \ and\ C_i = \sum_{j=1}^{k} c_j. \end{aligned} \tag{14}$$

Unlike equation (2), *y* here represents the *RL* branch only and *c* represents the capacitive branch of the generalized admittance. Coupling-inductors are represented by $M_{j,x}$ ($j=1..k,\ x=a,b,..$) in equation (14) and they induce a voltage drop on the *RL* branch proportional to $sM_{j,x}I_{j,x}$ where $I_{j,x}$ is the current passing through an inductor which is coupled with inductor L_x. For generalized admittance *j*, there are j_a coupling-inductors.

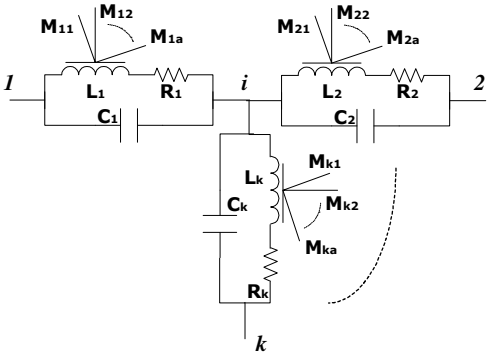

Figure 10. A general node in *RLC* circuit with coupling-inductors

In order to eliminate V_i from the system, we solve for V_i using (14) and obtain (15). Equation (16) describes the row corresponding to the first neighbor of node *i* where k_1 is the number of nodes connected to node 1.

$$V_i = \frac{\left[\begin{array}{l} y_1\left(V_1 - \sum_{x=1}^{1_a} sM_{1,x}I_{1,x}\right) + y_2\left(V_2 - \sum_{x=1}^{2_a} sM_{2,x}I_{2,x}\right) + \cdots \\ + y_k\left(V_k - \sum_{x=1}^{k_a} sM_{k,x}I_{k,x}\right) + sc_1V_1 + sc_2V_2 + \cdots + sc_kV_k \end{array}\right]}{Y_i + sC_i} \tag{15}$$

$$\begin{aligned} &Y_1V_1 + sC_1V_1 - y_1\left(V_i + \sum_{x=1}^{1_a} sM_{1,x}I_{1,x}\right) - \\ &y_2\left(V_2 - \sum_{x=1}^{2_a} sM_{2,x}I_{2,x}\right) - \cdots - y_{k1}\left(V_{k1} - \sum_{x=1}^{k1_a} sM_{k1,x}I_{k1,x}\right) - \\ &sc_1V_i - sc_2V_2 - \cdots sc_{k1}V_{k1} = 0 \end{aligned} \tag{16}$$

To eliminate node *i*, we substitute value of V_i as from (2) in equation (3) and obtain:

$$\begin{aligned} &\left(\hat{Y}_1 + s\hat{C}_1 + y_1 + sc_1 - \frac{(y_1 + sc_1)^2}{Y_i + sC_i}\right)V_1 - \\ &y_1\sum_{x=1}^{1_a} sM_{1,x}I_{1,x} - \frac{\sum_{j=2}^{k}(y_1 + sc_1)(y_j + sc_j)V_j}{Y_i + sC_i} + \\ &\frac{\sum_{j=1}^{k}(y_1 + sc_1)y_j\sum_{x=1}^{j_a} sM_{j,x}I_{j,x}}{Y_i + sC_i} - \\ &\sum_{\substack{r=1 \\ r\neq i}}^{k1} y_r\left(V_r - \sum_{x=1}^{r_a} sM_{r,x}I_{r,x}\right) - \sum_{\substack{r=1 \\ r\neq i}}^{k1} sc_rV_r = 0 \\ &where\ \hat{Y}_1 = \sum_{r=1,r\neq i}^{k1} y_r \ \text{and}\ \hat{C}_1 = \sum_{r=1,r\neq i}^{k1} c_r \end{aligned} \tag{17}$$

where $\hat{Y}_1$ is the sum of all admittances (*RL*) from node 1 except to node *i* and $\hat{C}_1$ is the sum of all capacitances from node 1 except to node *i*. Equation (17) can be simplified to:

$$\left(\hat{Y}_1 + s\hat{C}_1 + \frac{\sum_{j=2}^{k}(y_1+sc_1)(y_j+sc_j)}{Y_i+sC_i}\right)V_1 - \frac{\sum_{j=2}^{k}(y_1+sc_1)(y_j+sc_j)V_j}{Y_i+sC_i} +$$
$$\frac{-\sum_{j=2}^{k}(y_j+sc_j)y_1\sum_{x=1}^{1_a}sM_{1,x}I_{1,x} + \sum_{j=2}^{k}(y_1+sc_1)y_j\sum_{x=1}^{j_a}sM_{j,x}I_{j,x}}{Y_i+sC_i} - \qquad (18)$$
$$\sum_{\substack{r=1\\r\neq i}}^{k1} y_r\left(V_r - \sum_{x=1}^{r_a}sM_{r,x}I_{r,x}\right) - \sum_{\substack{r=1\\r\neq i}}^{k1}sc_rV_r = 0$$

Note that this is equivalent to adding *k-1 new* elements between node 1 and the *k-1* former neighbors of node *i*. Equation (18) is very similar to equation (5) except that the admittances are split between *RL* and *C* branch and there are additional terms representing coupling-inductors. Specifically, for any two neighbors of node *i*, say *m* and *n*, the elimination of node *i* results in the addition of a new generalized admittance between nodes *m* and *n* whose equivalence is given by:

$$(y_{mn}+sc_{mn}) + y_{mn}\sum_{x=1}^{mn_a}sM_{mn,x}I_{mn,x} \equiv$$
$$\frac{\left[\begin{array}{l}(y_m+sc_m)(y_n+sc_n) - \\ (y_n+sc_n)y_m\sum_{x=1}^{m_a}sM_{m,x}I_{m,x} + (y_m+sc_m)y_n\sum_{x=1}^{n_a}sM_{n,x}I_{n,x}\end{array}\right]}{Y_i+sC_i} \qquad (19)$$

where it is assumed that current flows from *m* to *i* and from *n* to *i* before eliminating node *i* and it flows from *m* to *n* after eliminating node *i*. For low frequency approximation, s^2 terms in (19) are ignored. Thus, when node *i* is eliminated, the coupling-inductors present on the *RL* branch between node *m* and node *i* are copied over to the new *RL* branch between nodes *m* and *n*. Similarly, all the coupling-inductors present on the *RL* branch between node *i* and node *n* are also copied over to the new *RL* branch between nodes *m* and *n*. The inductance, resistance, and capacitance of the new admittance between nodes *m* and *n* are obtained the same way as for circuits without coupling-inductors. Thus, for circuits without coupling-inductors, the elimination procedure falls back to the scheme presented in section 2. Figure 11 shows the rules for eliminating node *i* for a general *RLC* circuit with coupling-inductors.

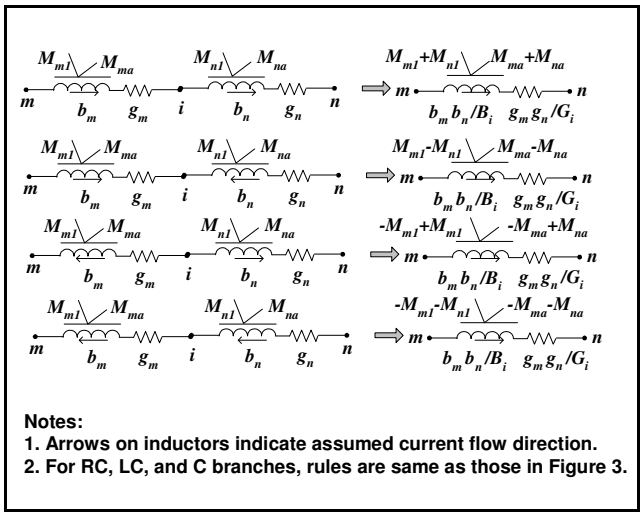

Figure 11. Rules for eliminating node *i* in RLCK circuit

6. REFERENCES

[1] A. Deutsch, P.W. Coteus, G.V. Kopcsay, H.H. Smith, C.W. Surovic, B. L. Krauter. D. C. Edelstein, and P. J. Restle, "On-Chip Wiring Design Challenges for Gigahertz Operation," *Proc. of the IEEE*, Vol. 89, No. 4, 2001.

[2] Y.I. Ismail, E.G. Friedman, and J. L. Neves, "Figures of Merit to Characterize the Importance of On-Chip Inductance," *IEEE Trans.on VLSI Systems*, Vol. 7, No. 4, pp. 442-449, 1999.

[3] xCalibre® Enabling Accurate Design Analysis, IC Station, Mentor Graphics.

[4] A. Devgan, R.A. Rohrer, "Efficient simulation of interconnect and mixed analog-digital circuits in ACES," *IEEE Proc. of the 8th Int.Con.on VLSI Design*, 1995.

[5] C.L. Ratzlaff, N. Gopal, and L. T. Pillage, "RICE: Rapid Interconnect Circuit Evaluator," *Proc. of IEEE DAC*, pp. 555-560, 1991.

[6] L.T. Pillage, R.A. Rohrer, "Asymptotic Waveform Evaluation for Timing Analysis," *IEEE Tran. on CAD*, Vol. 9, No. 4, pp. 352-366, 1990.

[7] Feldmann and R.W. Freund, "Efficient Linear Circuit Analysis by Pade Approximation via the Lanczos Process," *IEEE Trans. on CAD*, vol. CAD-14, pp. 639-649, 1995.

[8] M. Silveria, M. Kamon, I. Elfadel, and J. White, "A coordinate transformed Arnoldi algorithm for generating guaranteed stable reduced-order models of arbitrary RLC circuits", *IEEE/ACM Proc. of ICCAD*, pp. 288-294, 1996.

[9] A. Odabasioglu, M. Celik, and L.T. Pileggi, "PRIMA: Passive Reduced-order Interconnect Macromodeling Algorithm," *IEEE Trans. on CAD*, pp. 645-654, 1998.

[10] A. Devgan, P.R. O'Brien, "Realizable Reduction for RC Interconnect Circuits," *Digest of Technical Papers, IEEE/ACM Prof. of ICCAD*, pp. 204-207, 1999.

[11] P.J.H. Elias and N.P. van der Meijs, "Extracting Circuit Models for Large RC Interconnections that are Accurate up to a Predefined Signal Frequency," *IEEE/ACM Proc. of DAC*, pp. 764-769, 1996.

[12] A.J. van Genderen and N.P. van der Meijs, "Extracting Simple But Accurate RC Models for VLSI Interconnect", *Proc. of ISCAS*, pp. 2351-2354, 1988.

[13] B.N. Sheehan, "TICER: Realizable Reduction of Extracted RC Circuits," Digest of Technical Papers, *IEEE/ACM Proc. of ICCAD*, pp. 200-203, 1999.

[14] P.J.H. Elias and N.P. van der Meijs, "Efficient Moments Extraction of Large Inductively Coupled Interconnection Networks," *Proc. of ISCAS*, pp. IV 540-543, 1996.