A Fast Symbolic Computation Approach to Statistical Analysis of Mesh Networks with Multiple Sources *

Zhigang Hao and Guoyong Shi
School of Microelectronics
Shanghai Jiao Tong University, Shanghai, 200240, China
e-mail: {haozhigang, shiguoyong}@ic.sjtu.edu.cn

Abstract— Mesh circuits typically consist of many resistive links and many sources. Accurate analysis of massive mesh networks is demanding in the current integrated circuit design practice, yet their computation confronts numerous challenges. When variation is considered, mesh analysis becomes a much harder task. This paper proposes a symbolic computation technique that can be applied to the moment-based analysis of mesh networks with multiple sources. The variation issues are easily taken care of by a structured computation mechanism, which can naturally facilitate sensitivity based analysis. Applications are addressed by applying the computation technique to a set of mesh circuits with varying sizes.

I. INTRODUCTION

Interconnect has become the current design focus in the integrated circuit design practice. Tremendous research effort has been observed in the past twenty years; addressing the interconnect problems from the perspectives of modeling, analysis, physical design, manufacturing, and yield, etc. More recently, new research activities have been directed toward the process variation issues and the related problems such as statistical timing and signal integrity.

The moment-based analysis approaches initiated by Pillage et al. [1, 2] and the ensuing model order reduction techniques have been the dominating analysis techniques for interconnect related computation tasks. Along this line, recently we have observed that quite a number of researches have attempted to extend the model order reduction techniques to considering the variation issues. Typical approaches, among others, are 1) Incorporating interval analysis to matrix computation, resulting interval algebra for reduced-order modeling [3]; 2) Symbolic model order reduction techniques [4] which are far from mature; and 3) Other approximate or parametric approaches formulated under the name of *statistical timing analysis*, such as [5, 6].

Obviously, the model order reduction based approaches are confronting lots of difficulties as long as mesh networks are concerned. The major difficulties are: 1) massive links make those efficient methods such as RICE [2] not applicable (the efficiency of the interval methods of [3] also is limited without a tree structure.) 2) multiple sources create the hardest obstacle to reduced-order modeling, as the input-output multiplicity makes it hard to create compact macromodels.

Since the key computation load in model order reduction arises from matrix computations, such as solving direct current (DC) solutions repeatedly, extending such computation strategies directly to variational matrices in the form of symbolic model order reduction [4] or interval model order reduction [3] are not expected to be sustainable. However, many recent research works have suggested that low-order moments computed from the interconnect circuits can be very accurate and useful for physical design automation tasks, such as [7, 8, 9]. The usefulness of moments has motivated us to investigate the feasibility of symbolic computation of moments and its applications.

This line of research has gone through the following phases. It was discovered in [10] that the moment computation for tree-structured circuits takes an elegant recursive form [11], thus it can be programmed structurally or symbolically. Meanwhile the moment sensitivity can be computed symbolically. The work of [12] extends [10] to dealing with links, so that the symbolic computation of circuit moment can be applied to mesh networks as well, but at that time the mesh circuits were limited to having one *single source*. The key technique used in [12] is a successive link decomposition procedure initially proposed in [13, 14].

This work is a significant continuation of the previous work [10, 12] by lifting the *single* source assumption to consider multiple sources. The computation complexity remains polynomial. Multiple-source mesh networks arise frequently in the IC design practice, such as in the clock mesh design [15].

The rest of the paper starts from a background review presented in section II, where to help understanding we describe the symbolic moment computation procedure by walking through a simple mesh circuit with one *single* source. Then in section III the *multiple-source* problem is solved by slightly adapting the computation diagram developed for the single-source mesh circuits, without changing the underlying *branch tearing* based computation principle. An outline for the moment sensitivity computation is presented in section IV, which basically follows the same computation procedure and implementation strategy developed earlier in [10, 12]. The performance of the proposed computation methodology is evaluated in section V via applications to statistical timing analysis where several approximation strategies are investigated and compared. This paper concludes in section VI.

II. ANALYZING A SIMPLE MESH WITH ONE SINGLE SOURCE – A REVIEW

Shown in Fig. 1(a) is a small mesh circuit with one source V_s. By viewing this circuit as a spanning tree rooted at V_s added with two link resistors G_1 and R_4, one can devise a

*This research was supported in part by Shanghai Pu Jiang Research Foundation (Grant No. 07pj14053) and the Initiative Research Fund for overseas returnees from the Ministry of Education of China (2008), and by the National Natural Science Foundation of China, Grant No. 60876089.

978-1-4244-5767-0/10/$26.00 2010 IEEE

structural computation procedure for computing the moments at all circuit nodes.

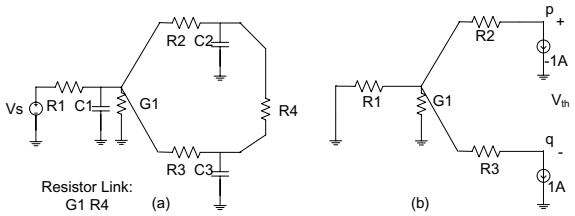

Fig. 1. (a) A circuit with two resistor links. (b) The Thevenin equivalent circuit for computing R_{th} for the link R_4.

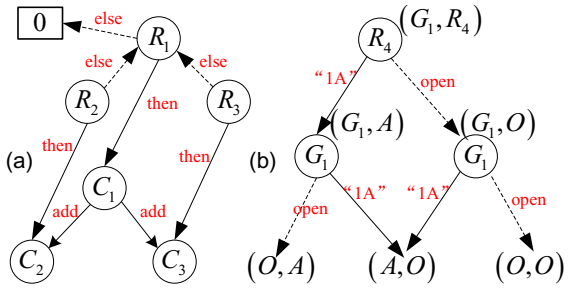

Fig. 2. (a) A tree-BDD. (b) A link-BDD.

First, the tree circuit with the resistive links G_1 and R_4 removed (only the capacitive loads remain at the non-root nodes) can be solved by an iterative tree traversal algorithm [2, 11]. Let $m_{i,k}$ be the kth order moment at node i, it holds that

$$m_{i,k} = \sum_{R_\ell \in P_i} R_\ell \sum_{j \in T_\ell} C_j \cdot m_{j,k-1} - \sum_{L_\ell \in P_i} L_\ell \sum_{j \in T_\ell} C_j \cdot m_{j,k-2}, \quad (1)$$

where P_i represents the path from the full-tree root to node i, the summation index $R_\ell \in P_i$ means summing over all the resistors R_ℓ on the path P_i, T_ℓ denotes the subtree rooted at node ℓ, and the summation index $j \in T_\ell$ indicates the summation over all nodes belonging to the subtree T_ℓ (including the root of T_ℓ).

Let $p(j)$ be the parent node of the node j in a tree circuit. The recursion formula (1) can be rewritten as

$$m_{i,k} = m_{p(i),k} + R_i \sum_{j \in T_i} C_j \cdot m_{j,k-1}, \quad (2)$$

which implies that the kth order moments can be computed recursively along the path from the tree-root toward the target node where a moment is requested. We observe that only *addition* and *multiplication* are involved in the expression (2). Therefore, it is very easy to program the computation of (2) in a binary decision diagram (BDD) as it is used for logical expressions. The summation in (2) is over a tree, which also can be structurally programmed with a BDD [10].

The main data structures used in the implementation resemble the circuit tree structure (see Fig. 2(a)), where all the R nodes and the associated arrows (solid and dashed) form a BDD while the C nodes and the (solid) arrows form a tree. The BDD nodes and the tree nodes are linked as shown in Fig. 2(a) which implements the computation of (2); each R node has a solid arrow pointing at a respective C node, where the *solid* ('then') arrows of R's stand for the *multiplication* in the expression (2) and the *dashed* ('else') arrows of R's stand for the *addition* in the expression (2). The tree consisting of the C nodes in Fig. 2(a) is for calculating the summation in (2). We call the computation diagram shown in Fig. 2(a) a *tree-BDD*, to differentiate it from the other BDD to be discussed next.

The difficulty arises from the resistive links existing between nodes of a tree circuit (or between a node to the ground), which must be handled differently from a capacitive coupling [10]. A good approach that maintains the merit of structural computation is by apply successive branch tearing to all resistive links. It turns out that by integrating the previous tree-BDD computation structure with the branch tearing process to be described next, the moment analysis of a whole mesh with multiple links and multiple sources can be done symbolically with only a polynomial complexity.

The Kron's branch tearing method [16] works as follows. For each selected resistive branch R, a circuit is decomposed into two: one (denoted *circuit-O*) with the branch R removed (open), the other (denoted *circuit-A*) with the branch R replaced by a current source I_R,

$$I_R = \frac{V_{oc}}{R_{link} + R_{th}}, \quad (3)$$

where R_{link} is the impedance of the removed branch, R_{th} is the Thevenin equivalent impedance seen by R_{link}, and V_{oc} is the open circuit voltage across the removed R. By a basic circuit analysis [16], the nodal voltages (or moments) of the original circuit are obtained by superposition of the nodal voltages (or moments) of *circuit-O* and *circuit-A*. For mesh circuits, replacing a resistive link by a current source is equivalent to removing the link while placing two opposite but equal valued current sources at the two branch ends (see Fig. 1(b)).

A good consequence of the above decomposition is that we have removed one resistive link and just solve two new circuits, one added with current sources at the certain nodes. By applying a repeated branch tearing process to all resistive links in a mesh, we end up with a set of tree circuits, all driven by (possibly different) pure current sources at the tree nodes, which can be solved by a tree-BDD discussed earlier.

The link-tearing process can be represented by a binary decision diagram (BDD) again (called a *link-BDD*) as shown in Fig. 2(b) for the working example. By Kron's tearing and the superposition principle, each node in the link-BDD performs the following computation for all nodal moments

$$m_{i,k} = m_{i,k}^{(O)} - I_R \cdot m_{i,k}^{(A)}, \quad (4)$$

where $m_{i,k}$ is the kth order moment of node i, the superscript 'O' stands for the *circuit-O* while the superscript 'A' stands for the *circuit-A* mentioned above. One should notice the sharing of one node in Fig. 2(b) which is a natural consequence of the link-decomposition process.

The link-decomposition method for mesh analysis was first proposed by Lee et al. [13, 14] where the Kron's tearing was applied. Lee et al. also introduced a BDD based computation scheme for managing the sharing during decomposition. But they did not propose a solution to meshes with multiple-source.

Also they did not treat the moment computation in a symbolic formulation; hence, they did not address the sensitivity computation issue.

III. Mesh with Multiple Sources

It would be nontrivial to extend the single-source computation introduced above to multiple-source cases if the source superposition principle is used directly. In that way one would have to construct a set of computation diagrams for computing tree circuits with the roots at the different sources. This approach is in principle feasible, but would end up with an inefficient implementation. An alternative approach is to keep the single spanning tree formulation, while taking care of the multiple sources once again by adopting a decomposition process as we have developed for the resistive links. Our earlier implementation for single-source mesh circuits can be reused to the maximum extent by taking the latter approach.

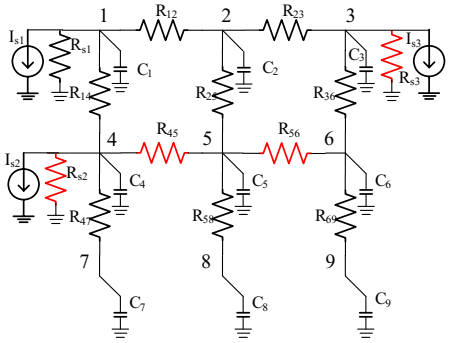

Fig. 3. A mesh driven by multiple sources.

We assume without loss of generality that all driving sources (either voltage or current) have inherent driving resistances. For the example shown in Fig. 3, three driving sources are placed in the mesh, all in the current form. For a consistent development of a structured computation process, we select an arbitrary one source (say, source I_{s1}) for the tree root and identify an arbitrary spanning tree as the underlying tree circuit to start the analysis.

The next step is to introduce another link decomposition process but this time to the (non-root) source resistors. The differences from the resistive links are that the links now are connected to the ground and meanwhile there exist current sources in parallel to the grounding resistor links. As the grounding resistive links are progressively decomposed, the remaining pure current sources are easily taken care of by a *tree-BDD* that symbolically solves the spanning tree circuit only driven by pure current sources at the tree nodes (as discussed in section II.)

An illustration of this computation process is given in Fig. 4, where the links R_{45} and R_{56} are decomposed together with the source resistors R_{s2} and R_{s3} in the same *link-BDD*. The current sources I_{s2} and I_{s3} that carry through the diagram top-down is due to superposition. Reading from the root node downward, the tuple $(R_{45}, R_{56}, R_{s2}+I_{s2}, R_{s3}+I_{s3})$ refers to the original circuit with two non-root sources; the tuple $(R_{45}, R_{56}, R_{s2}, A)$ refers to the circuit with the link R_{s3} replaced by a unity current source with all other independent sources switched off; the tuple (O, O, I_{s2}, I_{s3}) refers to the circuit with the all the (intra

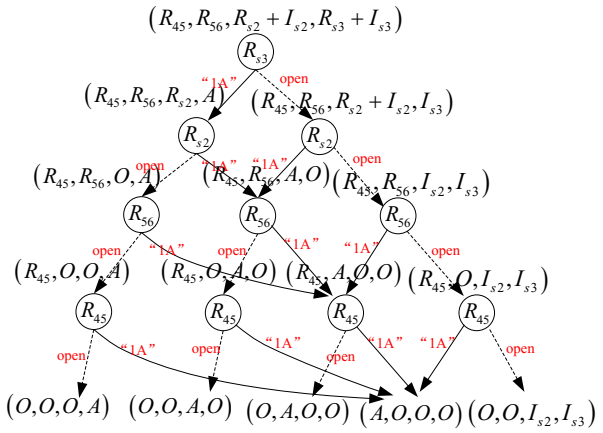

Fig. 4. Multi-source-link-BDD for the example circuit.

and grounding) link resistors removed except for the two pure current sources I_{s2} and I_{s3}; and all other notations in Fig. 4 are supposed self-evident. We stress that, whenever computing the Thevenin equivalent impedance for one specific link (whether grounding or not), only the unity current source(s) corresponding to the chosen link is applied while all other (independent and equivalent) sources must be turned *off*. Therefore, only one 'A' symbol is allowed to appear in all tuples involving an 'A' at the leaf nodes in Fig. 4. Since the underlying branch-tearing principle remains unchanged, the moment computation performed by each BDD vertex in the link-BDD for the multiple-source circuits remains the same as given by (4).

The symbolic moment computation procedure for multiple-source meshes is summarized below.

Moment Computation Algorithm for Multiple-Source Mesh Circuits

Step 1. Select one source as the primary source. Find a spanning tree of the original circuit rooted at the primary source. Construct a tree-BDD for the spanning tree and construct a link-BDD by decomposing all the inter-node links and grounding source links.

Step 2. Evaluate the tree circuits at the leaf vertices of the link-BDD using the tree-BDD by substituting the C-values in the tree-BDD by the appropriate current values resulting from branch tearing.

Step 3. Evaluate the moments of the link-BDD from bottom-up.

Step 4. Repeat Steps 2 and 3 to get the nodal moments of the succeeding order.

Note that the independent sources are used only in the computation for the 0th order moments and are set to zero for computing the higher-order moments.

The computation complexity for the nodal moments is estimated similarly to [12]. Suppose a mesh has N nodes, S sources, and L links. Then we have $(L+S-1)$ links to decompose. According to the triangular computation diagram of the link-BDD shown in Fig. 4, there are $O((L+S)^2)$ link-BDD vertices in total. Since at each vertex all (N) nodal moments are computed once, the computation complexity of the link-BDD is proportional to $O(N(L+S)^2)$. On the other hand, one

round of the tree-BDD computation is linear in the number of the tree nodes, i.e., $O(N)$. Since we have $(L+S)$ leaf vertices in the link-BDD, the computation cost for all leaf vertices of a link-BDD is $O(N(L+S))$.

Clearly, if the number of links L and the number of sources S are much smaller than the number of nodes N, i.e., $\max\{L,\, S\} \ll N$, then the computation complexity is almost linear in the number of mesh nodes N. However, the computation cost increases with the numbers of links and sources. In the worst case, the number of mesh links could reach the order of $O(N)$ and each mesh intersection could be driven by a current source, it could result in the worst computation complexity of $O(N^3)$, but still polynomial. It is worth stressing that a *symbolic* computation algorithm of polynomial complexity for moment computation is considered acceptable; as we know, a symbolic matrix inversion algorithm would of the exponential complexity if not implemented properly. The link-decomposition process developed so far fully takes the advantage of the circuit structure and its inherent superposition properties; while a mathematical symbolic matrix inversion procedure pays no attention to the underlying circuit structure and properties.

IV. Sensitivity Computation

The gradient of moment with respect to the resistors and the capacitors are defined by:

$$\nabla_{\vec{R}} m_{i,k} := [\partial m_{i,k}/\partial R_1, \cdots, \partial m_{i,k}/\partial R_q]^T,$$
$$\nabla_{\vec{C}} m_{i,k} := [\partial m_{i,k}/\partial C_1, \cdots, \partial m_{i,k}/\partial C_q]^T,$$

where $\vec{R}$ and $\vec{C}$ are the vectors containing the selected R_i's and C_i's from the spanning tree circuit, q is the total number of R_i's or C_i's selected for sensitivity analysis. Taking the gradient with respect to $\vec{R}$ or $\vec{C}$ in the equation (2) gives rise to

$$\nabla_{\vec{R}} m_{i,k} = \nabla_{\vec{R}} m_{p(i),k} + \vec{e}_i \sum_{j\in T_i} C_j m_{j,k-1} + R_i \sum_{j\in T_i} C_j \nabla_{\vec{R}} m_{j,k-1}, \tag{5}$$

$$\nabla_{\vec{C}} m_{i,k} = \nabla_{\vec{C}} m_{p(i),k} + R_i \sum_{j\in T_i} \vec{e}_j m_{j,k-1} + R_i \sum_{j\in T_i} C_j \nabla_{\vec{C}} m_{j,k-1}, \tag{6}$$

where $\vec{e}_i$ is the ith basis vector in the q-dimensional space. Equations (5) and (6) are the basic equations for the *resistive* sensitivity and the *capacitive* sensitivity, respectively. There is no difficulty in defining the higher order derivatives. Inductors in serious with the tree resistors and the respective inductor sensitivity also can be considered; but are omitted for the sake of brevity.

The moment gradient expressions given in (5) and (6) indicate clearly that the gradient computation can easily be embedded in the structured moment computation process developed in the previous section. What we need to do in implementation is to add an extra moment gradient vector to each BDD vertex participating in the moment computation. The gradient vector propagates through the BDD data structure bottom-up and the arithmetic derivative operations given in (5) and (6) are performed successively by the BDD vertices.

If links are present as in mesh circuits, the moment sensitivity (or gradient vector) would have to propagate through the link-BDD vertexes as well (see Fig. 4). Knowing that the BDD vertices and arrows basically implement the functional composition, we can implement the sensitivity propagation by following the differentiation chain rules for composite functions. Since the link-BDD vertexes perform the moment calculation in the form of (4), taking the gradient operation w.r.t. any single parameter p leads to

$$\nabla_p m_{i,k} = \nabla_p m_{i,k}^{(O)} - \nabla_p I_R \cdot m_{i,k}^{(A)} - I_R \cdot \nabla_p m_{i,k}^{(A)}. \tag{7}$$

Since

$$I_R = \frac{m_{p,k}^{(O)} - m_{q,k}^{(O)}}{R_{link} + m_{p,k}^{(A)} - m_{q,k}^{(A)}}, \tag{8}$$

$\nabla_p I_R$ is computed in terms of $\nabla_p m_{i,k}^{(O)}$ and $\nabla_p m_{i,k}^{(A)}$. Equation (7) shows the computation being performed by each vertex in the link-BDD during the gradient computation.

From the above discussion we know that there is no change of the computation data structure and data flow during the sensitivity computation except that additional memory has to be allocated for the storage of the intermediate gradient vectors. Therefore, the sensitivity computation complexity remains polynomial. The detailed complexity analysis is omitted to save space. But we point out that the moment sensitivity computation can be carried out together with the moment computation during the bottom-up traversal of the tree-BDD and the link-BDD.

Remark 1 *We are aware of the existence of the automatic differentiation technique that can compile a C language program to generate routines that perform the differentiation of a programmed function. We are not taking this approach because our moment computation diagram already has a very regular data structure that can easily incorporate the differentiation computation. Be aware that the main operations involved in the moment computation are the very basic arithmetic operations (additions, multiplications, and divisions), the corresponding differentiations can easily be programmed.*

V. Performance Evaluation and Applications

We have implemented the proposed symbolic moment and sensitivity computation method in the form of a circuit simulator using C++. This simulator is designed for statistical interconnect analysis, thus code named *Statistical Interconnect Simulator* (SIS). The interconnect simulator was tested on an Intel 2.83G CPU with 4GB memory running a Redhat Enterprise Linux 4 operating system. Its performance was compared to HSPICE (version 2005.03) and Matlab 2007a running on the same machine.

A. Performance of the Interconnect Simulator

The per-unit-length (PUL) resistor and capacitor values were set to 0.1Ω and $4.11E-16F$, respectively, according to a $0.18\mu m$ process technology. Shown in Table I are five industrial level clock mesh designs used for testing. The first four columns list the circuit details. For a comprehensive comparison, we have evaluated a number of running times. the columns

TABLE I
STATISTICAL INTERCONNECT SIMULATOR PERFORMANCE

Nodes	Mesh Type	Sources	R-Links	BDD Building Time(s)	4th Order Moment Evaluation Time(s)	4th Order Moment Sensitivity Time(s)	Obtain Reduced Order Model Time(s)	Speedup over Superposition
576	5x5	30	54	0.11	0.03	0.23	0.01	29.8
1296	7x7	56	104	0.89	0.09	1.53	0.02	55.5
1720	9x9	63	143	2.28	0.16	3.63	0.02	63.2
3838	13x13	130	298	20.97	0.75	32.88	0.04	130.2
6976	15x15	180	404	70.31	1.83	106.61	0.07	179.8

5-8 list the time splits of the SIS simulation. As expected, the construction times for two BDDs dominate the whole simulation. As soon as the BDDs are constructed, the times spent on evaluation are less prominent for moment evaluations (up to the 4th order) and reduced-order model evaluations. We also compared the SIS to another simulation method by taking the multiple sources one after another by superposition. The speedup of SIS over superposition is remarkable as listed in the last column.

To have a feeling of how slow a symbolic matrix inversion package would be, we tested a case by using Matlab for inverting the matrix arising from the smallest mesh with 576 nodes. Matlab took over eight hours to compute the symbolic matrix inversion. Because the symbolic expressions derived are so complicated, the numerical evaluation phase would also be time-consuming. Therefore, it is in general not recommended to use a general purpose symbolic matrix inversion package to do symbolic moment computation.

The performance merit of SIS is more appealing than HSPICE as far as the sensitivity computation is concerned. SIS computes the sensitivity analytically, thus each round of numerical evaluation is extremely fast, while HSPICE has to run for multiple times to calculate the numerical sensitivities by finite differences.

B. Applications of Moment Sensitivity

The symbolically computed moment sensitivity has many applications. One application of the moment sensitivity is to derive approximate moments by Taylor expansion, with which repeated moment computation can be avoided.

The Taylor expansion of moment w.r.t. parameters up to the second order can be written as

$$m_k(\delta_1, \delta_2, \ldots, \delta_p) = m_k(0) + \sum_{r=1}^{p} \delta_r \frac{\partial m_k(0)}{\partial \delta_r} + \frac{1}{2} \sum_{u=1}^{p} \sum_{v=1}^{p} \delta_u \delta_v \frac{\partial^2 m_k(0)}{\partial \delta_u \partial \delta_v} + \cdots \quad (9)$$

Thus, if the first and second order sensitivities are computed at a set of nominal parameters, then the new moments for the perturbed parameter values can be approximately computed using (9) without running the moment computation engine again.

We tested this idea for two interconnect trees, one involving RCL elements, the other with RC elements. We assume that all the R, C and L values are subject Gaussian statistical variation with 3σ at the level of 30% variation. Table II shows that the relative errors for the different orders of moments are mild while the computation speedups over the exact computations are remarkable. Therefore, for some large-scale applications, taking the Taylor approximation approach is a favorable choice.

Another application of the sensitivity is to derive rapidly the statistical timing distribution at some or all mesh nodes for timing related mesh design tasks. For efficiency reasons, using certain empirical timing metrics is more competitive than using dynamical models. Among the moment-based metrics proposed recently, we find that the D2M metric [7] is of high fidelity for 50% delay calculation for mesh networks. Other metrics such as the S2M [8] can be used for statistical skew analysis.

The sensitivity function can be used for mapping between the probability density functions (PDFs), the so-called *direct mapping* method proposed in [10, 12].

Let X be a real random variable with the probability density function (PDF) $f_X(x)$. Let $y = g(x)$ be a first-order differentiable function of x. In Probability Theory, the PDF of the random variable Y satisfies

$$f_Y(y) = \sum_k \frac{f_X(x_k)}{|dg(x_k)/dx|}, \quad (10)$$

where x_k, $k = 1, 2, \cdots$, are all points satisfying $g(x_k) = y$.

The D2M delay metric [7] is calculated using the first and second-order moments

$$D2M(i) = ln2 \frac{m_{i,1}^2}{\sqrt{m_{i,2}}}. \quad (11)$$

Once the moment vector and moment sensitivity vectors have been calculated, the timing PDFs can easily be evaluated using equation (10). The direct mapping method has the advantage of directly obtaining the mapped PDF function without running the time-consuming Monte Carlo simulation.

For example, we can use the stated method for examining the timing distribution at any mesh nodes when the mesh geometry is subject to statistical global variation (such as in yield analysis.) We assume that all the mesh R and C values are subject Gaussian statistical variation with 3σ at the level of 30% perturbation. The direct mapping method generates a timing PDF plot by sampling the parameter space. Shown in Fig. 5 is the PDF generated for the 7x7 mesh listed in Table I. A similar PDF was also generated by running HSPICE Monte Carlo (5000 times) and the marvelous speedups are observed in Table III.

TABLE II
RELATIVE ERROR AND TIME SPEEDUP USING TAYLOR APPROXIMATION OVER EXACT MOMENT CALCULATION

			1st order moment		2nd order moment		3rd order moment	
		speedup	mean	std	mean	std	mean	std
RCL	1st order Taylor	96	3.04%	1.56%	8.29%	2.99%	16.23%	2.90%
	2nd order Taylor	77	0.92%	0.08%	3.22%	0.36%	7.76%	1.53%
RC	1st order Taylor	99	3.71%	1.47%	9.73%	2.42%	18.56%	1.45%
	2nd order Taylor	80	0.88%	0.26%	3.27%	0.31%	8.12%	0.93%

TABLE III
SPEEDUP OF DIRECT MAPPING OVER HSPICE FOR PDF PROFILING.

Mesh Type	All Nodes Average Error		Speedup to
	mean	std	Monte Carlo
5x5	0.21%	1.27%	1121
7x7	0.09%	1.12%	843
9x9	0.23%	0.09%	658
13x13	0.17%	0.75%	356
15x15	0.13%	0.62%	340

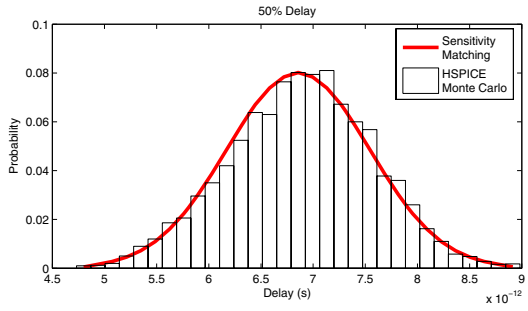

Fig. 5. 50% delay distribution of one node of mesh 7x7.

VI. CONCLUSION

Symbolically analyzing a mesh network driven by multiple sources has never been addressed in the literature. It is traditionally believed that symbolic reduced-order modeling requires the symbolic computation of matrix inversions. By this work, we have demonstrated a fundamental fact that circuits in the form of mesh with multiple driving sources can be *symbolically* analyzed by a successive link decomposition process with *polynomial* memory and time complexity, not the *exponential* complexity as commonly believed. To some degree, this work also solves the model order reduction problem with multiple sources.

The applications of the proposed approach are not limited to statistical timing analysis and variational reduced-order modeling. Other potential applications lie in the physical synthesis of clock meshes. Clock mesh analysis and synthesis have been the subject of many research efforts recently. Some techniques developed in the literature [17, 18] have obvious limitations. An efficient mesh analysis scheme is crucial for clock mesh synthesis. As a continuing research topic, we are going to apply the symbolic mesh analysis methodology to clock mesh synthesis including mesh placement and sizing.

REFERENCES

[1] L. Pillage and R. Rohrer, "Asymptotic waveform evaluation for timing analysis," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 9, pp. 352–366, April 1990.

[2] C. Ratzlaff and L. Pillage, "RICE: Rapid interconnect circuit evaluation using AWE," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 13, no. 6, pp. 763–776, June 1994.

[3] J. D. Ma and R. A. Rutenbar, "Fast interval-valued statistical modeling of interconnect and effective capacitance," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 25, no. 4, pp. 710–724, April 2006.

[4] G. Shi, B. Hu, and C. J. R. Shi, "On symbolic model order reduction," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 25, no. 7, pp. 1257–1272, July, 2006.

[5] K. Agarwal, M. Agarwal, D. Sylvester and D. Blaauw, "Statistical interconnect metrics for physical-design optimization," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 25, no. 7, pp. 1273–1288, July 2006.

[6] J. K. Zeng and C. P. Chen, "Deep submicron interconnect timing model with quadratic random variable analysis," *Proc. Design, Automation and Test in Europe*, pp.1091-1094, March 2008

[7] C. J. Alpert, A. Devgan, and C. Kashyap, "A two moment RC delay metric for performance optimization," in *Proc. International Symposium on Physical Design (ISPD)*, pp. 73–78, San Diego, CA, 2000.

[8] K. Agarwal, D. Sylvester, and D. Blaauw, "A simple metric for slew rate of RC circuits based on two circuit moments," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 23, no. 9, pp. 1346–1354, Sept. 2004.

[9] C. Kashyap, C. Alpert, F. Liu, and A. Devgan, "Closed-form expressions for extending step delay and slew metrics to ramp inputs for RC trees," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 23, no. 4, pp. 509–516, April 2004.

[10] Z. Hao and G. Shi, "Sensitivity approach to statistical signal integrity analysis of coupled interconnect trees," in *Proc. IEEE Midwest Symp. Circuits Syst.*, vol. 1, pp. 212–215, Aug. 2009.

[11] Q. Yu and E. S. Kuh, "Exact moment matching model of transmission lines and application to interconnect delay estimation," *IEEE Trans. on Very Large Scale Integration Systems*, vol. 3, no. 2, pp. 195–206, April 2002.

[12] Z. Hao and G. Shi, "Symbolic techniuqes for statistical timing analysis of RCL mesh networks with resistor loops," *to appear in IEEE International Symposium on Integrated Circuits*, Singapore, Dec. 2009.

[13] H. J. Lee, M. H. Lai, C. C. Chu, and W. S. Feng, "Moment computations for R(L)C interconnects with multiple resistor loops using ROBDD techniques," in *Proc. Asia Pacific Circuits Syst. Conf.*, vol. 1, pp. 525–528, Dec. 2004.

[14] H. J. Lee, M. H. Lai, C. C. Chu, and W. S. Feng, "Applications of tree/link partitioning for moment computations of general lumped RLC networks with resistor loops," in *Proc. IEEE Int. Symp. Circuits Syst.*, vol. 1, pp. 713–716, May. 2004.

[15] P. Restle et al, "A clock distribution network for microprocessors," *IEEE J. Solid-State Circuits*, vol. 36, no. 5, pp. 792–799, May 2001.

[16] R. Rohrer, "Circuit partitioning simplified," *IEEE Trans. Circuits Syst.*, vol. 35, no. 1, pp. 2–5, Jan 1988.

[17] H. Chen et al, "A sliding window scheme for accurate clock mesh analysis," in *Proc. Int. Conf. Computer-Aided Design*, pp. 939–946, Nov. 2005.

[18] X. Ye, P. Li, M. Zhao, R. Panda, and J. Hu, "Analysis of large clock meshes via harmonic-weighted model order reduction and port sliding," in *Proc. Int. Conf. Computer-Aided Design*, pp. 627–631, Nov. 2007.