Hierarchical Graph Reduction Approach to Symbolic Circuit Analysis with Data Sharing and Cancellation-Free Properties

Yang Song and Guoyong Shi
School of Microelectronics, Shanghai Jiao Tong University
Shanghai 200240, China
e-mail: shiguoyong@ic.sjtu.edu.cn

***Abstract*—Parallel to algebraic methods, graphical circuit analysis methods have the advantage of cancellation-free. This paper proposes a graph reduction method for hierarchical symbolic circuit analysis by applying a binary decision diagram (BDD) for data sharing. This method is extended from the Graph-Pair Decision Diagram (GPDD) method which was developed for two-port dependent sources. New graph construction rules for multiple-port dependent sources are introduced, with which large analog circuits can be analyzed hierarchically. The new hierarchical method guarantees the *cancellation-free* property at each layer of hierarchy. The BDD-based hierarchical analysis method can greatly reduce the analysis complexity of the entire circuit, while the software construction and circuit partition remain easy. The new method is compared to the algebraic hierarchical method based on DDD (Determinant Decision Diagram) which does not have the cancellation-free property. Comparable performance can be achieved with the new method which has the extra cancellation-free property.**

***Index Terms*—Analog integrated circuits, binary decision diagram (BDD), cancellation-free symbolic analysis, graph reduction, hierarchical analysis, multiple-port analysis.**

I. Introduction

Symbolic circuit analysis is mainly targeted at deriving analytical expressions of small-signal circuit behavior in terms of the circuit parameters. Because a symbolic expression is constructed only once and can be used repeatedly in design automation tasks such as sensitivity analysis, statistical analysis, ac response calculation for varying parameter values, etc., it is considered a helpful design tool in some computation-intensive analog design tasks. However, a main obstacle to developing such a tool is the computational complexity in solving large circuits. Because the majority of symbolic methods are based on enumeration, the computational complexity is exponential in nature. Even applying the most advanced data structure like Binary Decision Diagram (BDD) for data sharing [1], the nature of exponential complexity remains unchanged [2]. An effective approach to circumventing the difficulty is by hierarchical analysis, because a divider-and-conquer strategy can divide a large problem to subproblems of smaller scale, suppressing the analysis complexity of each subproblem.

The application of BDD for data sharing essentially makes an *explicit* enumeration process *implicit* [1]. The advantage seems significant, as having been demonstrated by the previous work of DDD (Determinant Decision Diagram) [3], which is an algebraic method, and the GPDD (Graph-Pair Decision Diagram) [4], which is a graphical method.

This paper is organized as follows. Section II addresses the cancellation-free problem in symbolic circuit analysis, where an example is given to show that DDD is not cancellation-free while GPDD is. The commonly used *Schur decomposition* method for hierarchical analysis is compared to the *symbolic stamp* method in Section III, where it is emphasized that the latter is more friendly to circuit partition. The main contribution of this work is presented in Section IV, where a graph transformation method is introduced to transform a multiport module to two-port dependent graph edges in order to apply the GPDD construction method. The transformation method forms the foundation for the hierarchical method described in Section V. Experiments and comparisons are presented in Section VI. Finally, Section VII concludes the paper.

II. The Issue of Term Cancellation

Advanced symbolic circuit analysis methods enumerate all signed product terms *implicitly* in that a sharing data structure BDD (binary decision diagram) is used [1]. Depending on the formulation, some symbolic techniques introduce intermediate symbols composed of the basic circuit elements. If the composite symbols in the product terms are expanded into the product terms of the basic circuit elements, *term-cancellation* would result. In contrast, some other symbolic techniques form the product terms directly from the circuit elements, the term-cancellation problem would not occur.

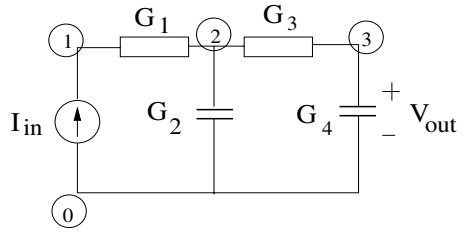

Fig. 1. A two-stage ladder circuit.

Consider the circuit shown in Fig. 1 which involves four conductance elements G_1, G_2, G_3, and G_4, with an in-

This research was supported by the National Natural Science Foundation of China (Grants No. 60876089 and No. 61176129).

978-1-4673-0772-7/12/$31.00 ©2012 IEEE

put current source I_{in} and an output voltage measured at $V_{out} = V_3$. We use this simple example to illustrate that the symbolic method DDD using the MNA formulation has the term-cancellation problem [3], while the graphical method GPDD does not [4].

The MNA formulation of the circuit is

$$\begin{bmatrix} G_1 & -G_1 & 0 \\ -G_1 & G_1+G_2+G_3 & -G_3 \\ 0 & -G_3 & G_3+G_4 \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \\ V_3 \end{bmatrix} = \begin{bmatrix} I_{in} \\ 0 \\ 0 \end{bmatrix}. \quad (1)$$

DDD [3] treats the coefficient matrix as a symbolic function as follows

$$A = \begin{bmatrix} a & b & 0 \\ c & d & e \\ 0 & f & g \end{bmatrix}, \quad (2)$$

where the composite symbols are listed here: $a = G_1$, $b = c = -G_1$, $d = G_1 + G_2 + G_3$, $e = f = -G_3$, and $g = G_3 + G_4$. The determinant $\det(A)$ expanded in product terms reads $\det(A) = adg - aef - bcg$. When the composite symbols are substituted with the circuit parameters, some product terms would cancel each other as seen below:

$$\begin{aligned} \det(A) &= adg - aef - bcg \\ &= G_1(G_1+G_2+G_3)(G_3+G_4) - G_1G_3^2 \\ &\quad -G_1^2(G_3+G_4) \\ &= G_1G_2G_3 + G_1G_2G_4 + G_1G_3G_4. \end{aligned}$$

DDD applies the Cramer's rule to get the symbolic transfer function from I_{in} to V_{out}, which can be expressed by

$$H(s) = \frac{cf}{adg - aef - bcg} \quad (3)$$

in terms of the composite symbols. The above expression is equivalent to

$$H(s) = \frac{G_1G_3}{G_1G_2G_3 + G_1G_2G_4 + G_1G_3G_4}, \quad (4)$$

if written in terms of the original circuit parameters.

With the double precision computation in contemporary computers, whether or not the expressions are cancellation free is not a serious problem for the computation accuracy. However, in certain applications such as in variational circuit analysis, the cancellation problem becomes an issue. One of such example is addressed in the recent work [5] where a symbolic method is developed to estimate the frequency-domain performance bound of an analog circuit subject to parameter variations. That work proposed to use the *affine interval analysis* technique for the performance bound estimation, in which the existence of term-cancellation would largely affect the accuracy of the estimated performance bound.

We are therefore motivated to develop a cancellation-free symbolic method that is favored in variational circuit analysis problems. The graph-based symbolic method GPDD is guaranteed to be cancellation-free [4] without the need of *decancelation* in DDD [6].

The GPDD method analyzes the circuit given in Fig. 1 by constructing a pair of graphs consisting of the circuit elements as the graph edges. The graph-pair is reduced according to a set of edge operation rules with the intermediate subgraph pairs shared in a BDD. After a GPDD is constructed, those reduction paths corresponding to certain spanning tree pairs generate the product terms satisfying the following homogeneous equation

$$-G_1G_3X + G_1G_2G_3 + G_1G_2G_4 + G_1G_3G_4 = 0, \quad (5)$$

where $X = I_{in}/V_{out}$ models the input-output as a voltage-controlled current source (VCCS) [4]. The symbolic transfer function derived from (5) is exactly identical to that in (4) but without performing any term cancellation.

In addition to being cancellation-free, another feature of GPDD is that the symbolic expression is directly composed of the circuit parameters, rather than intermediate symbols, which can be advantageous in sensitivity analysis.

For example, we would like to compute the sensitivity of the transfer function with respect to (w.r.t.) a circuit parameter, say, G_3. DDD would have to apply the chain rule to the symbolic expression in the following form

$$\begin{aligned} \frac{\partial \det(A)}{\partial G_3} &= a\frac{\partial d}{\partial G_3}g + ad\frac{\partial g}{\partial G_3} - a\frac{\partial e}{\partial G_3}f \\ &\quad -ae\frac{\partial f}{\partial G_3} - bc\frac{\partial g}{\partial G_3} \\ &= G_1G_2 + G_1G_4, \end{aligned}$$

where the last expression is again the result of several term cancelations. The existence of canceling terms in the original symbolic expression causes further term cancellation in the sensitivity expressions, which is wasteful in computation. Hence, without incurring too much overhead, one should choose a cancellation-free algorithm like GPDD to simplify the implementation meanwhile improve the efficiency [7].

For the same reason, cancellation-free should also be enforced in hierarchical symbolic analysis, which is addressed in this work.

III. Schur Decomposition versus Symbolic Stamp

A brief survey of the existing hierarchical methods is presented in [8]. Those previously developed hierarchical methods such as in [9] construct nested expressions without using a BDD-based sharing data structure. Hence, it would be difficult to use such methods for solving large analog circuit blocks. The DDD-based hierarchical strategy developed recently [10], [6], [11] basically takes the approach of Schur decomposition. After an MNA matrix is set up for a large circuit, the matrix is partitioned based on the circuit interconnection and Schur-decomposed into lower-dimensional submatrices, which are then solved by the DDD tool. Since DDD is not cancellation-free, every layer of the hierarchy has the term cancellation problem.

The Schur decomposition technique used in DDD can be reformulated in terms of *symbolic stamp* [8], because eliminating a submatrix block is equivalent to calculating a symbolic stamp for a selected subcircuit. The symbolic stamp concept is more circuit-oriented and generic for implementation. Both algebraic and graphical methods can be used for deriving symbolic stamps, which are studied and compared in this work.

A multiport subcircuit module can be described by a symbolic stamp in the transadmittance form (i.e., voltage-controlled currents) or in the m-port Y-matrix form given by

$$\begin{pmatrix} i_1 \\ \vdots \\ i_2 \end{pmatrix} = \begin{pmatrix} y_{1,1} & \cdots & y_{1,m} \\ \vdots & \ddots & \vdots \\ y_{m,1} & \cdots & y_{m,m} \end{pmatrix} \begin{pmatrix} v_1 \\ \vdots \\ v_m \end{pmatrix}. \quad (6)$$

The m^2 transadmittances, $y_{i,j}$, $i, j = 1, \cdots, m$, are the intermediate symbols in hierarchical symbolic analysis, which are manipulated by the circuit modules at higher levels.

The multiport symbolic stamps can either be assembled in an MNA matrix and analyzed by a DDD (as done in the work [8]) or to be processed by the graph reduction rules addressed in the next section.

IV. Graph Formulation for Multiport Module

Our strategy is to transform a multiple-branch controlled dependent source into a set of multiple one-to-one controlled sources satisfying the superposition law. Then the resulting graph can be passed to the previously developed GPDD tool [4], [12] for symbolic analysis.

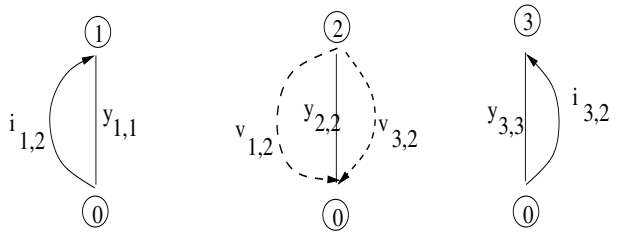

Fig. 2. Illustration of edge-pairs for a three-port module. Two voltage edges $v_{1,2}$ and $v_{3,2}$ are added at node '2'.

Without loss of generality, we describe the graph construction rule for a three-port network, whose Y-matrix is defined by

$$\begin{pmatrix} i_1 \\ i_2 \\ i_3 \end{pmatrix} = \begin{pmatrix} y_{1,1} & y_{1,2} & y_{1,3} \\ y_{2,1} & y_{2,2} & y_{2,3} \\ y_{3,1} & y_{3,2} & y_{3,3} \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix}. \quad (7)$$

The above equation describes three port currents $i_k = y_{k,1}v_1 + y_{k,2}v_2 + y_{k,3}v_3$ for $k = 1, 2, 3$. Each port current i_k is composed of three partial currents, denoted separately by $i_{k,j} := y_{k,j}v_j$ for $j = 1, 2, 3$. It is clear that each v_j controls three partial currents, $i_{1,j}$, $i_{2,j}$, and $i_{3,j}$, at the three ports. In graph, they are represented by three VCCS edge-pairs. Since the graph edges in GPDD must be named distinctly, we judiciously introduce three alternative names for the voltage variable v_2, namely, $v_{1,2}$, $v_{2,2}$, and $v_{3,2}$ (all equal to v_2). We then obtain the following three VCCS pairs: $i_{1,2} = y_{1,2}v_{1,2} = y_{1,2}v_2$, $i_{2,2} = y_{2,2}v_{2,2} = y_{2,2}v_2$, and $i_{3,2} = y_{3,2}v_{3,2} = y_{3,2}v_2$. Among them, $y_{2,2}$ is the coefficient of a self-controlling VCCS, which is simply an admittance element. In graph representation, a single edge for $y_{2,2}$ is sufficient. The other two, $y_{1,2}$ and $y_{3,2}$, are the coefficients of two cross controlling VCCS's, each corresponding to a pair of VCCS edges, i.e., $i_{1,2} = y_{1,2}v_{1,2}$ and $i_{3,2} = y_{3,2}v_{3,2}$. As illustrated in Fig. 2, the edge $v_{1,2}$ across the element $y_{2,2}$ controls the current $i_{1,2}$ entering node '1', while the edge $v_{3,2}$ across the element $y_{2,2}$ controls the current $i_{3,2}$ entering node '3'.

It is easily justified that the placement of voltage and current edges at the port terminals as described above is a direct consequence of the current superposition law and voltage parallelism. After all ports of a circuit module are processed by placing appropriate edges according to the principle described above, the GPDD tool [4] can be applied directly for graph reduction.

V. The Proposed Hierarchical Scheme

Many methods can be used to obtain the symbolic expressions for the m^2 transadmittances in the Y-matrix of a circuit module. It is important to observe that the symbolic transadmittances $y_{i,j}$'s are derived from the same circuit module, hence they must have lots of sharable subexpressions. This basic property can be explored by a BDD-based symbolic method for a highly compact representation of all m^2 expressions in a single data structure. DDD is of course a valid candidate for such a construction, but one should be aware of its term-cancellation problem. To avoid the cancellation issue, GPDD becomes a better candidate for the symbolic stamp construction.

In the GPDD-based analysis, all $y_{i,j}$'s for one circuit module are computed by a shared *multi-root* GPDD, with each $y_{i,j}$ computed at one of the GPDD roots. Given a set of symbol values, one evaluation of the GPDD produces all the transadmittance values in the stamp. The technical details on the multi-root GPDD construction have been discussed in [8].

The previous work [8] has demonstrated the feasibility of applying DDD in the top layer for solving an MNA matrix assembling all symbolic stamps in multi-root GPDD. But this strategy has the restriction of hierarchical analysis in only two layers. A more flexible strategy addressed below is to allow multi-level hierarchical analysis, for which a multiport circuit module is still treated as a graphical element. Hence, no matrix is ever formed in the whole hierarchical procedure. This methodology provides a more flexible means for arbitrary circuit partitioning and arbitrary hierarchical interconnection of the subcircuit blocks. Even a large circuit with cross-level interconnection as shown in Fig. 3 can be treated easily by the proposed hierarchical scheme. Such a cross-level multi-hierarchy is especially suited for analyzing analog integrated circuits, where certain circuit sub-blocks are intentionally designed with a nested hierarchy.

The Hierarchical GPDD Procedure:

Step 1. Partition the circuit into a multi-level hierarchy, each subcircuit module in the hierarchy should have limited complexity.

Step 2. Run GPDD construction for the subcircuit modules from bottom up to build a hierarchical decision diagram. The transadmittances of a multiport module appear as the intermediate symbols in the circuit modules above the hierachy. Create only one GPDD

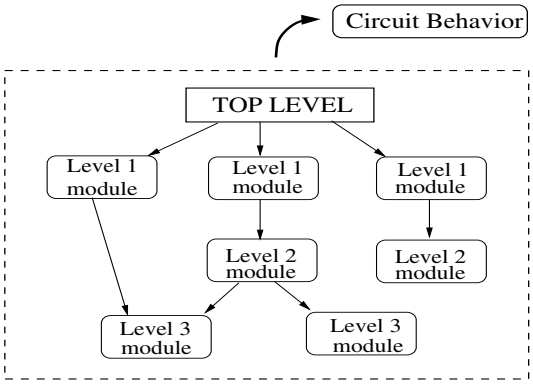

Fig. 3. Multi-level hierarchical circuit structure.

for those modules having the identical subcircuit topology.

Step 3. Run numerical evaluations in the hierarchical GPDD.

VI. Experimental Results

The main purpose of this section is to compare two BDD-based hierarchical implementations for symbolic circuit analysis. The *Hier-GPDD* program implements the hierarchical GPDD method proposed in this paper, while the *Hier-DDD* program implements the hierarchical DDD method, which is also based on the *symbolic stamp* concept rather than the *de-cancellation* strategy implemented in [6], [11]. A recently proposed DDD algorithm [13] is implemented in the *Hier-DDD* program for DDD construction which is believed to be more efficient than the traditional implementation. All programs were implemented in C++ and executed on an Intel Core2 Duo T7100 1.80GHz processor with 2GB memory.

It is noted that a fair comparison between two BDD-based computation methods requires caution, because the efficiency of any BDD implementation is known to be highly dependent on the variable ordering [14]. No optimal ordering is known for almost all practical problems. Since the two BDD programs deal with different hash objects (minors or subgraphs) in construction, the symbol orders or even the ordering heuristics used are not comparable. While ordering is a critical issue in comparison, a hierarchical scheme in general weakens the issue of ordering because the divided sub-problems are much smaller and hence the BDD construction complexity becomes less dependent on the symbol ordering. Therefore, in this experiment we only compare the overall efficiency of the two programs, without making any effort on selecting good symbol orders.

Three large op-amp circuits are used as benchmarks for test in this work:

- Benchmark 1: The $\mu A725$ bipolar operational amplifier containing 26 transistors (Fig. 4).
- Benchmark 2: A folded-cascode two-stage MOSFET Miller amplifier containing 24 transistors (Fig. 5).
- Benchmark 3: A MOSFET operational amplifier containing 44 transistors (Fig. 6) [15].

A $0.18\mu m$ model library was used for dc analysis. The level-3 small-signal model shown in Fig. 7 was used for both MOS circuits. The bipolar small-signal model is standard [4].

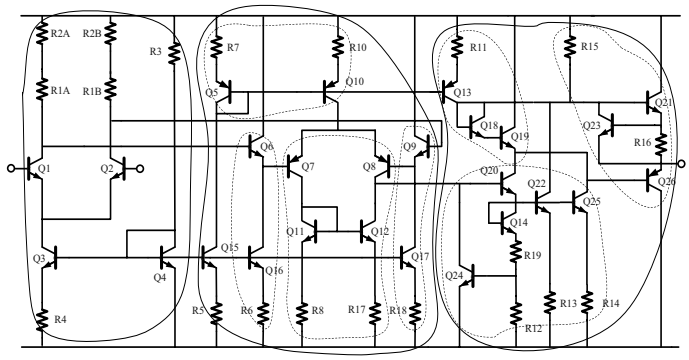

Fig. 4. Benchmark 1: Partitioned $\mu A725$.

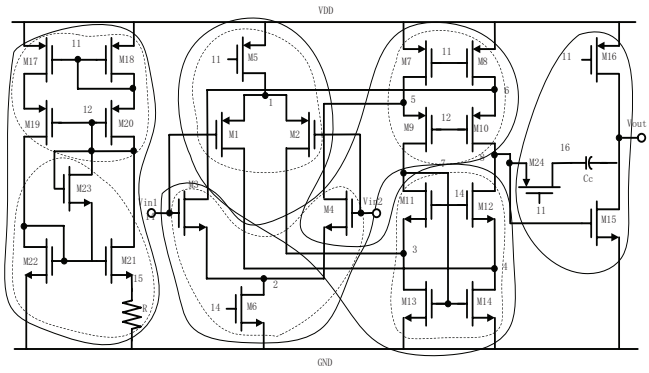

Fig. 5. Benchmark 2: A partitioned rail-to-rail Miller amplifier.

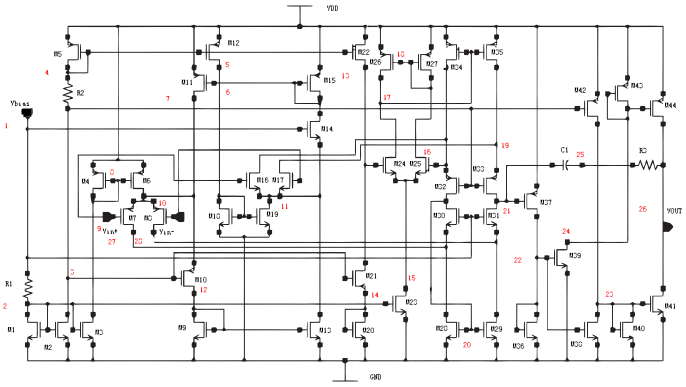

Fig. 6. Benchmark 3: A MOS op-amp containing 44 transistors [15].

The $\mu A725$ op-amp was considered hard for non-hierarchical analysis even using BDD. It was solved exactly by the de-cancellation method in [6] or by GPDD [4]. We attempted two partitions for $\mu A725$, one in two levels listed in Table I and the other in three levels listed in Table II. In the tables the module index $L_{m,n}$ refers to the nth circuit module at the partition level m. Hence, $L_{1,1}$ always refers to the main circuit.

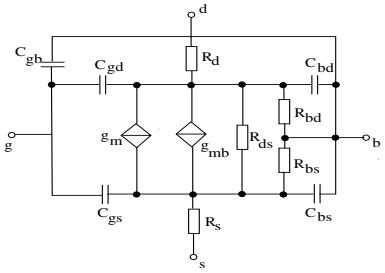

Fig. 7. MOS level-3 small signal model.

The construction times of Hier-GPDD and Hier-DDD for the two partitions of $\mu A725$ are listed in Tables III and IV, respectively, together with some information on the partitioned circuit modules. It seems that a finer partition leads to faster analysis using less memory. The notations $|GPDD|$ and $|DDD|$ refer to the numbers of vertices (i.e., sizes) of the relative BDD constructions.

TABLE I
PARTITION LIST FOR BENCHMARK 1 ($\mu A725$) WITH 2 LEVELS.

Module Index	Components
$L_{1,1}$	$L_{2,1}, L_{2,2}, L_{2,3}$
$L_{2,1}$	$Q_1, Q_2, Q_3, Q_4, R_1, R_2, R_3, R_4$
$L_{2,2}$	$Q_5, Q_{10}, R_7, R_{10}, Q_6, Q_{16}, R_6, Q_7, Q_8, Q_{11}, Q_{12}, R_8, R_{17}, Q_9, Q_{17}, R_{18}, Q_{15}, R_5$
$L_{2,3}$	$Q_{13}, Q_{18}, Q_{19}, R_{11}, Q_{21}, Q_{23}, Q_{26}, R_{15}, R_{16}, Q_{14}, Q_{20}, Q_{22}, Q_{24}, Q_{25}, R_{12}, R_{13}, R_{14}, R_{19}$

TABLE II
PARTITION LIST FOR BENCHMARK 1 ($\mu A725$) WITH 3 LEVELS.

Module Index	Components
$L_{1,1}$	$L_{2,1}$,$L_{2,2}$,$L_{2,3}$
$L_{2,1}$	Q_1,Q_2,Q_3,Q_4,R_1,R_2,R_3,R_4
$L_{2,2}$	$L_{3,1}$,$L_{3,2}$,$L_{3,3}$,$L_{3,4}$,Q_{15},R_5
$L_{2,3}$	$L_{3,5}$,$L_{3,6}$,$L_{3,7}$
$L_{3,1}$	Q_5,Q_{10},R_7,R_{10}
$L_{3,2}$	Q_6,Q_{16},R_6
$L_{3,3}$	Q_7,Q_8,Q_{11},Q_{12},R_8,R_{17}
$L_{3,4}$	Q_9,Q_{17},R_{18}
$L_{3,5}$	Q_{13},Q_{18},Q_{19},R_{11}
$L_{3,6}$	Q_{21},Q_{23},Q_{26},R_{15},R_{16}
$L_{3,7}$	Q_{14},Q_{20},Q_{22},Q_{24},Q_{25},R_{12},R_{13},R_{14},R_{19}

Listed in Tables V and VII are the circuit partition and the comparison of analysis performances, respectively, for Benchmark 2. The result indicates that a larger GPDD construction (larger size) does not necessarily mean a lower speed, which is worth attention in BDD-based applications.

The largest op-amp circuit tested in this work is Benchmark 3, which was once solved exactly by the hierarchical method in [8]. A five-level partition is given in Table VI. The resulting runtime performance is listed in Table VIII together with the performances of the other two benchmarks for comparison. Both hierarchical methods GPDD and DDD were able to solve the three large op-amp circuits in only a few seconds by the chosen hierarchical partitions.

It is interesting to observe that although the Hier-GPDD might construct a larger hierarchical GPDD than the Hier-DDD (because Hier-GPDD involves more symbols in general), it does not mean that its overall construction speed is slower.

TABLE V
PARTITION LIST FOR BENCHMARK 2.

Module Index	Components
$L_{1,1}$	$L_{2,1}$,$L_{2,2}$,$L_{2,3}$,$L_{2,4}$
$L_{2,1}$	$L_{3,1}$,$L_{3,2}$
$L_{2,2}$	$L_{3,3}$,$L_{3,6}$
$L_{2,3}$	$L_{3,4}$,$L_{3,5}$
$L_{2,4}$	M_{15},M_{16},M_{24},C_c
$L_{3,1}$	M_{17},M_{18},M_{19},M_{20}
$L_{3,2}$	M_{21},M_{22},M_{23},R
$L_{3,3}$	M_1,M_2,M_5
$L_{3,4}$	M_3,M_4,M_6
$L_{3,5}$	M_7,M_8,M_9,M_{10}
$L_{3,6}$	M_{11},M_{12},M_{13},M_{14}

TABLE VI
PARTITION LIST FOR BENCHMARK 3

Module Index	Components
$L_{1,1}$	$L_{2,1}, L_{2,2}$
$L_{2,1}$	$L_{3,1}, L_{3,2}$
$L_{2,2}$	$R_3, C_1, L_{3,3}, L_{3,4}, M_{36}, M_{37}$
$L_{3,1}$	$L_{4,1}, M_{32}, M_{33}$
$L_{3,2}$	$L_{4,2}, L_{4,3}, M_{30}, M_{31}$
$L_{3,3}$	M_{39}, M_{43}, M_{44}
$L_{3,4}$	$M_{38}, M_{40}, M_{41}, M_{42}$
$L_{4,1}$	$L_{5,1}, L_{5,2}, L_{5,3}, L_{5,4}, L_{5,5}$
$L_{4,2}$	M_{28}, M_{29}
$L_{4,3}$	$L_{5,6}, L_{5,7}, L_{5,8}$
$L_{5,1}$	R_2, M_5, M_{12}, M_{22}
$L_{5,2}$	$M_{16}, M_{17}, M_{18}, M_{19}$
$L_{5,3}$	$R_1, M_1, M_2, M_3, M_{23}$
$L_{5,4}$	M_{20}, M_{21}
$L_{5,5}$	$M_{24}, M_{25}, M_{26}, M_{27}, M_{34}, M_{35}$
$L_{5,6}$	M_{11}, M_{14}, M_{15}
$L_{5,7}$	M_4, M_6, M_7, M_8
$L_{5,8}$	M_9, M_{10}, M_{13}

VII. CONCLUSION

A graphical hierarchical symbolic circuit analysis method is presented, which has extended the previously developed GPDD algorithm to multiport circuit modules. By the graphical method, the sum of product terms in each hiearchy are guaranteed cancellation-free and the sum of product terms at the bottom-level are cancellation-free expressions of the circuit element parameters. It is demonstrated that both hierarchical strategies (based on DDD or GPDD) formulated in the framework of symbolic stamps can greatly suppress the BDD construction complexity by reducing the sub-problem sizes. Although both methods are valid for most applications, whenever the property of cancellation-free is required in certain application, the hierarchical GPDD is recommended.

REFERENCES

[1] G. Shi, "A survey on binary decision diagram approaches to symbolic analysis of analog integrated circuits," *Analog Integrated Circuits and Signal Processing*, 2011, Springer online first.

[2] ——, "Computational complexity analysis of determinant decision diagram," *IEEE Trans. on Circuits and Systems - II: Express Briefs*, vol. 57, no. 10, pp. 828–832, 2010.

[3] C. J. R. Shi and X. D. Tan, "Canonical symbolic analysis of large analog circuits with determinant decision diagrams," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 19, no. 1, pp. 1–18, January 2000.

TABLE III
PERFORMANCE COMPARISON FOR BENCHMARK 1 ($\mu A725$) WITH 2 LEVELS.

Module Index	#Edges	#Nodes	#Ports	$\|GPDD\|$	GPDD Time (sec)	$\|DDD\|$	DDD Time (sec.)
$L_{1,1}$	34	7	–	773	0.016	130	0.013
$L_{2,1}$	26	8	4	548	0.046	341	0.063
$L_{2,2}$	62	16	4	135,785	3.015	11870	1.999
$L_{2,3}$	62	13	3	91,682	2.067	12022	0.687
Total	–	–	–	228,788	5.165	24,363	2.781

TABLE IV
PERFORMANCE COMPARISON FOR BENCHMARK 1 ($\mu A725$) WITH 3 LEVELS.

Module Index	#Edges	#Nodes	#Ports	$\|GPDD\|$	GPDD Time (sec)	$\|DDD\|$	DDD Time (sec)
$L_{1,1}$	34	7	–	773	0.016	130	0.014
$L_{2,1}$	26	8	4	548	0.046	341	0.062
$L_{2,2}$	44	16	4	2,987	0.099	1,667	0.238
$L_{2,3}$	27	13	3	700	0.066	193	0.027
$L_{3,1}$	12	4	2	99	0.038	52	0.007
$L_{3,2}$	11	4	3	103	0.043	79	0.012
$L_{3,3}$	22	7	4	476	0.045	366	0.062
$L_{3,4}$	11	4	3	126	0.038	79	0.012
$L_{3,5}$	16	5	3	133	0.05	99	0.017
$L_{3,6}$	17	4	3	131	0.05	79	0.014
$L_{3,7}$	29	8	3	4,822	0.13	494	0.074
Total	–	–	–	10,898	0.67	3,579	0.586

TABLE VII
PERFORMANCE COMPARISON FOR BENCHMARK 2 WITH 3 LEVELS

Module Index	#Edges	#Nodes	#Ports	$\|GPDD\|$	GPDD Time (sec)	$\|DDD\|$	DDD Time(sec)
$L_{1,1}$	63	6	–	5,111	0.117	142	0.014
$L_{2,1}$	13	3	2	68	0.039	41	0.005
$L_{2,2}$	32	6	5	26	0.04	474	0.098
$L_{2,3}$	32	6	5	26	0.044	474	0.099
$L_{2,4}$	28	7	3	780	0.054	660	0.107
$L_{3,1}$	36	12	3	1,846	0.074	1,006	0.165
$L_{3,2}$	28	8	2	895	0.054	357	0.052
$L_{3,3}$	27	11	4	1,343	0.06	1,437	0.239
$L_{3,4}$	27	11	4	1,343	0.061	1,437	0.237
$L_{3,5}$	36	14	4	3,563	0.108	3,341	0.602
$L_{3,6}$	36	13	4	2,487	0.092	2,137	0.372
Total	–	–	–	17,488	0.793	11,506	2.042

TABLE VIII
PERFORMANCE COMPARISON FOR ALL THREE BENCHMARKS.

Circuit	#Edges	#Nodes	$\|GPDD\|$	GPDD Time (sec)	$\|DDD\|$	DDD Time (sec)
Benchmark 1 (3 levels)	166	31	10,432	0.682	3,579	0.586
Benchmark 2 (3 levels)	218	66	17,488	0.793	11,506	2.042
Benchmark 3 (5 levels)	399	114	197,274	6.771	62,794	10.359

[4] G. Shi, W. Chen, and C. J. R. Shi, "A graph reduction approach to symbolic circuit analysis," in *Proc. Asia South-Pacific Design Automation Conference (ASPDAC)*, Yokohama, Japan, Jan. 2007, pp. 197–202.

[5] Z. Hao, S. X. D. Tan, R. Shen, and G. Shi, "Performance bound analysis of analog circuits considering process variations," in *Proc. IEEE/ACM Design Automation Conference (DAC)*, CA, USA, June. 2011, (accepted for publication).

[6] S. X. D. Tan, W. Guo, and Z. Qi, "Hierarchical approach to exact symbolic analysis of large analog circuits," in *Proc. Design Automation Conference*, 2004, pp. 860–863.

[7] G. Shi and X. Meng, "Variational analog integrated circuit design by symbolic sensitivity analysis," in *Proc. International Symposium on Circuits and Systems (ISCAS)*, Taiwan, China, May 2009, pp. 3002–3005.

[8] H. Xu, G. Shi, and X. Li, "Hierarchical exact symbolic analysis of large analog integrated circuits by symbolic stamps," in *Proc. Asia South-Pacific Design Automation Conference (ASPDAC)*, Yokohama, Japan, Jan. 2011, pp. 19–24.

[9] M. M. Hassoun and P. M. Lin, "A hierarchical network approach to symbolic analysis of large-scale networks," *IEEE Trans. on Circuits and Systems – I: Fundamental Theory and Applications*, vol. 42, no. 2, pp. 201–211, 1995.

[10] X. D. Tan and C. J. R. Shi, "Hierarchical symbolic analysis of analog integrated circuits via determinant decision diagrams," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 19, no. 4, pp. 401–412, April 2000.

[11] S. X. D. Tan, "A general hierarchical circuit modeling and simulation algorithm," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, no. 3, pp. 418–434, March 2005.

[12] W. Chen and G. Shi, "Implementation of a symbolic circuit simulator for topological network analysis," in *Proc. Asia Pacific Conference on Circuits and Systems (APCCAS)*, Singapore, Dec. 2006, pp. 1327–1331.

[13] G. Shi, "A simple implementation of determinant decision diagram," in *Proc. International Conf. on Computer-Aided Design (ICCAD)*, San Jose, CA, USA, Nov. 2010, pp. 70–76.

[14] R. E. Bryant, "Graph-based algorithms for boolean function manipulation," *IEEE Trans. on Computers*, vol. C-35, no. 8, pp. 677–691, 1986.

[15] T. McConaghy and G. G. E. Gielen, "Globally reliable variation-aware sizing of analog integrated circuits via response surfaces and structural homotopy," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 28, no. 11, pp. 1627–1640, Nov. 2009.