Computational Complexity Analysis of Determinant Decision Diagram

Guoyong Shi, *Member, IEEE*

***Abstract*—A determinant decision diagram (DDD) uses a binary decision diagram (BDD) to calculate a determinant symbolically, which is then applied for symbolic circuit analysis. The efficiency of such a technique is determined mainly by a symbol ordering scheme. Finding an optimal symbol order is an non-deterministic polynomial-time hard problem in the practice of BDD. So far, it is unknown what an optimal order is for a general sparse matrix. This brief shows that a row-wise (or column-wise) order is an optimal BDD order for *full* matrices in the sense that the DDD graph constructed has the minimum number of vertices (i.e., the DDD size). The optimal DDD size is proven to be $(n \cdot 2^{n-1})$ for an $n \times n$ full matrix. This size provides a DDD complexity measure that has rarely been investigated in the literature.**

***Index Terms*—Binary decision diagram, computational complexity, determinant decision diagram, symbolic circuit analysis.**

I. Introduction

SYMBOLIC circuit analysis derives analytically the characterizations of a circuit in terms of the circuit parameters. Design tools with such symbolic characterization are considered helpful for analog design automation. For example, the sensitivity of a design metric (e.g., the dc gain or the unity-gain frequency) with respect to a device parameter can be calculated analytically for design decision making [1].

All symbolic analysis tools proposed so far run into the curse of exponential complexity in one way or another. This difficulty had existed for many decades until the proposal of the determinant decision diagram (DDD) approach [2], which effectively tapered the exponential complexity growth to a much milder level. The key mechanism that achieves the complexity attenuation is the enforcement of substructure sharing as in all binary decision diagrams (BDDs) [3]. In the context of DDD, the shared substructures are the minors of different dimensions. Many applications and extensions of DDD have appeared in the literature [4]–[7] and most recently in [8].

A symbolic analysis technique with a lower complexity means a higher capacity for analyzing larger circuits. Since the DDD technique is inherently exponential as well, a sensible question to ask is how low the exponential complexity can be for a set of problems. Here, the complexity mainly refers to the memory required for storing a full DDD, which in general is proportional to the time required for constructing such a DDD. A natural complexity measure is by the number of vertices created for a DDD (known as the DDD size and denoted by $|DDD|$).

A matrix without any zero element is called a *full matrix*. Since a full matrix is the most regular, a discussion on the DDD complexity for such a class of matrices is tractable. It is proven in this brief that a row-based or column-based symbol ordering can produce a minimal DDD for any full matrix, and the minimum DDD size is $(n \cdot 2^{n-1})$ for an $n \times n$ full matrix.

The complexity number $(n \cdot 2^{n-1})$ reveals that the DDD complexity grows exponentially with an asymptotic growth rate of two by using an optimum order. We know that a direct enumeration of the determinant of a $n \times n$ full matrix would produce $T_n := n!$ product terms, whose growth rate is by n, without counting the growing length of each term. Therefore, a large degree of complexity compression can be achieved by using a DDD.

Any $n \times n$ sparse matrix is a special case of an $n \times n$ full matrix by setting a portion of the elements to zero. While the DDD size for a full matrix can be greatly different from that for a sparse matrix of the same dimension, one may devise a problem-specific symbol ordering scheme to achieve a surprising complexity reduction for DDD construction.

A discussion on the DDD optimality must be based on a DDD with *canonicity*. It is justified in Section II that the row and column indexes of a minor are sufficient for identifying a minor sharing (i.e., minor hashing). The optimality results on the symbol order and the DDD size for full matrices are established in Section III with the help of a layered expansion directed graph (digraph). A discussion is given in Section IV, where the complexity of row (or column) ordering is compared to the existing Greedy-Labeling scheme proposed in [2]. A conclusion is made in Section V.

II. Canonicity by Minor Hashing

The Laplace expansion of a determinant $\det A$ along the ith row can be written as

$$\det A = \sum_{j=1}^{n} (-1)^{i+j} a_{i,j} M_{i,j} \tag{1}$$

where $a_{i,j}$ is the element of matrix A at the ith row and the jth column, and $M_{i,j}$ denotes the minor of matrix A after its ith row and jth column are deleted. Since each $M_{i,j}$ is again a determinant of reduced dimension, it can be expanded further by a selected row or column. In general, the expansion does not have to follow a row or a column.

Manuscript received March 29, 2010; revised June 3, 2010; accepted July 14, 2010. Date of publication September 16, 2010; date of current version October 15, 2010. This research was supported by the National Natural Science Foundation of China (Grant No. 60876089). This paper was recommended by Associate Editor P. Li.

The author is with the School of Microelectronics, Shanghai Jiao Tong University, Shanghai 200240, China (e-mail: shiguoyong@ic.sjtu.edu.cn).

Digital Object Identifier 10.1109/TCSII.2010.2067791

By defining two basic operations "Minor" and "Remainder" [2], any determinant can be expanded according to any selected order of the nonzero elements (called the *symbol order* in DDD). Specifically, given any minor, the "Minor" operation on an element $x_{i,j}$ existing in the minor refers to the operation of deleting the ith row and the jth column from the minor, while the "Remainder" operation on an element $x_{i,j}$ refers to setting the element $x_{i,j}$ to zero. Clearly, the "Minor" operation reduces the dimension of the minor under operation by one, while the "Remainder" operation does not change the dimension of the minor.

Basically, the expansion of a determinant can be expressed by a sequence of the binary operations according to a preselected order of the nonzero elements. During the expansion, the intermediate minors produced by the "Minor" and "Remainder" operations are preserved in the memory for the succeeding operations.

The main contribution of the original DDD work [2] is to model the sequence of binary operations using a BDD [3], with which all the identical intermediate minors are shared (the so-called *canonicity* in BDD parlance). In [2], the sharing was implemented by invoking the routines offered by a zero-suppressed binary decision diagram (ZBDD) package [9].

In a DDD, each vertex is associated with a symbol and a minor generated by the preceding operations. For example, the root vertex is associated with the first symbol and a minor, which is the original determinant. At each nonterminal vertex, one of the operations, "Minor" or "Remainder," is applied to the associated minor by selecting one nonzero element out of the minor according to a given symbol order. Note that multiple vertices of the same symbol name could exist in one DDD because such vertices are associated with different minors.

Proposition 1: Given a symbol order for a determinant to be expanded in a DDD. If in the middle of DDD construction, two minors associated with two DDD vertices of the same symbol name have identical row and column indexes, then the two minors must be identical, i.e., their elements coincide exactly.

It is important to note that during the DDD construction, there exist minors of the same row and column indexes but different minor entries because the "Remainder" operations could have set some nonzero entries to zero. Therefore, this proposition is not self-evident.

Proof: The proof is based on a given symbol order. Without loss of generality, we consider the two 3×3 minors M_1 and M_2

$$M_1 = \begin{vmatrix} \times & \alpha & \times \\ \beta & \times & \times \\ \times & 0 & 0 \end{vmatrix}, \quad M_2 = \begin{vmatrix} \times & 0 & \times \\ \beta & \times & \times \\ \times & 0 & 0 \end{vmatrix}, \tag{2}$$

where α, β, and $\times$ indicate the nonzero elements remaining. Suppose the two minors have identical row and column indexes and are associated to two DDD vertices of the same symbol 'β' being processed. If the two minors have at least one element (symbol) different, for instance, symbol 'α,' which exists in minor M_1 but not in minor M_2 [see (2)], then a simple argument leads to a contradiction. The vanished α in minor M_2 indicates that the symbol α must precede symbol β. However, the symbol α still existing in the minor M_1 indicates that symbol β precedes symbol α, which is contradictory. ■

Proposition 1 plays the key role in designing a minor sharing mechanism for DDD construction. It suffices to maintain a minor hash table to store only the minor row and column indexes. When creating a DDD vertex as the result of a "Minor" or "Remainder" operation, look up the minor hash table to see whether another DDD vertex has a minor with the identical row and column indexes. If yes, the existing DDD vertex is shared. This minor sharing mechanism ensures the DDD canonicity by construction. In contrast, the authors of [2] used a ZBDD package for DDD construction, and introduced a minor cache table for efficiency. However, the overall efficiency could be jeopardized by cache collisions because some minors expanded before might have to be reexpanded later. Moreover, a final DDD reduction phase must be performed to maintain canonicity.

III. Results on Optimality

A discussion on the optimal DDD size need not only the canonicity of a DDD, but also a regular organization of a DDD.

Recall that a "Minor" operation reduces the minor under operation by one dimension, but the "Remainder" operation keeps the dimension of the minor unchanged. An $n \times n$ minor is reduced successively to a scalar (1×1) minor after applying $(n-1)$ "Minor" operations in addition to a number of "Remainder" operations in the middle. For an easy derivation of the DDD size, we convert a DDD in one-to-one correspondence to a new graph called *layered expansion digraph*.

Since the "Remainder" operations produce minors of the same dimension, we place all the DDD vertices created by the "Remainder" operations in the same layer. Since the "Minor" operations reduce the minor dimensions by one, we place all the DDD vertices created by the "Minor" operations in the succeeding layers.

Take the 3×3 full determinant

$$\begin{vmatrix} a^{(1)} & b^{(2)} & c^{(3)} \\ d^{(4)} & e^{(5)} & f^{(6)} \\ g^{(7)} & h^{(8)} & i^{(9)} \end{vmatrix} \tag{3}$$

for an example, where the numbers in the superscripts indicate the order of the symbol in the ordered symbol list. For instance, $a^{(1)}$ means that the symbol a is the first element to be expanded. The order given in (3) is a row-wise order.

The DDD created for the 3×3 determinant with the given order is shown in Fig. 1(a), where the DDD vertices associated with the minors of the same dimension are placed in the same row (called *layer*). The horizontally drawn arrows are all dashed in the figure; they correspond to the "Remainder" operations. The solid arrows connecting vertices in the neighboring layers correspond to the "Minor" operations.

For an $n \times n$ determinant in general, after n layers of expansion, the first layer consists of all DDD vertices associated with $n \times n$ minors, the second layer consists of all DDD vertices associated with $(n-1) \times (n-1)$ minors, and so on. The bottom layer consists of all DDD vertices associated with 1×1 minors (i.e., scalars).

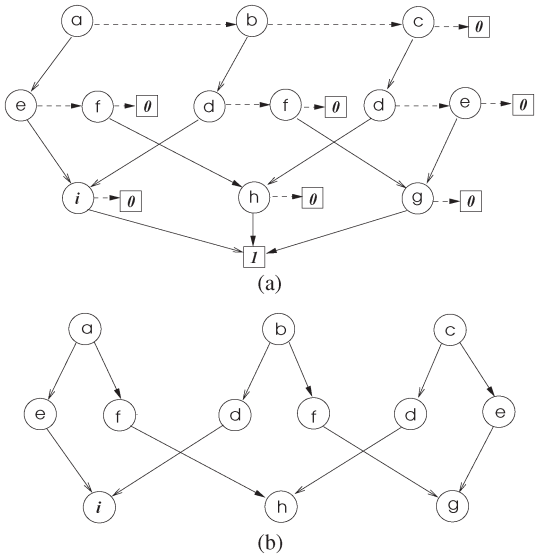

Fig. 1. (a) DDD for the 3 × 3 full matrix. (b) Corresponding layered expansion digraph.

In the middle of DDD construction, some expansions should be terminated because the newly generated minors become singular. The singularity test is implementation-specific. It is not discussed further in this brief.

By definition, a *one*-path (or term-path) refers to a path in a DDD graph that starts from the DDD root and ends at the terminal vertex one. Along any path, all vertices lying at the tails of the solid arrows form a product term of the original determinant [2]. Since the dashed arrows are not significant for identifying a product term, we decide to convert a DDD to another form of digraph in which the term-paths are more easily identified.

Note that the vertices successively connected by the dashed arrows [e.g., vertices e and f in the second layer of Fig. 1(a)] are to be multiplied by the same vertex in the preceding layer [vertex a in Fig. 1(a)], which has a solid arrow connecting to the leading vertex [vertex e in Fig. 1(a)] of the successively dash-connected vertices. In the converted digraph, solid arrows are added explicitly to manifest the multiplications. For instance, a solid arrow is added to connect vertex a to vertex f in the converted digraph. Shown in Fig. 1(b) is the layered digraph converted from the DDD given in Fig. 1(a), where all the dashed arrows have been removed and some solid arrows have been added. It is clearly seen that all the paths from the vertices in the top layer to the vertices in the bottom layer produce all the product terms of the determinant. The six paths in Fig. 1(b) represent the six product terms of the 3 × 3 full determinant in (3).

The converted layered digraph is instrumental to the development of an optimality argument. We shall use the two features that come with a layered digraph. First, regardless of the symbol order, the number of paths (i.e., terms) in any digraph for a given determinant is invariant. Second, the number of vertices in a digraph is equal to the size of the DDD from which the digraph is converted. In this brief, the DDD size (i.e., $|DDD|$) refers to the number of vertices in a DDD excluding the two terminal vertices 'one' and 'zero.'

Let $C(n,k)$ denote the so-called n *choose* k function in combinatorics, i.e., $C(n,k) = n!/(k!(n-k)!)$.

Theorem 1: By a row-wise (or column-wise) order, the DDD size of an $n \times n$ full matrix is $(n \cdot 2^{n-1})$.

Proof: Without loss of generality, we prove the theorem for a natural row-wise order going from the first row (left to right) successively down to the last row.

According to Proposition 1, a DDD vertex is uniquely determined by its symbol name and the row and column indexes of the associated minor where the symbol lies. When an intermediate $k \times k$ minor is expanded, there exist k elements in the first row of this minor. The k elements have their own symbol names, but they share the identical row and column indexes of the same underlying minor. Hence, k different DDD vertices must be created for the k elements in the first row.

On the other hand, all the minors generated in the $(n-k+1)$th layer of the digraph are of dimension $k \times k$, which means that $(n-k)$ "Minor" operations have been applied to the original determinant. Since the expansion is in the natural row-wise order, all the $k \times k$ minors are actually the results of selecting k columns out of the $k \times n$ submatrix formed by the bottom k rows of the original determinant, which implies that there are $C(n,k)$ such $k \times k$ minors. (Note that any such $k \times k$ minor does not become singular until all of its first row elements are set to zero after applying k "Remainder" operations.)

Any two $k \times k$ minors so selected may have some common symbols in their first rows, but must not have the identical set of column indexes, although their row indexes are all identical. Therefore, the DDD vertices created for the first-row elements of any two different $k \times k$ minors are all distinct (i.e., cannot be shared).

Since there are $C(n,k)$ minors of size $k \times k$ and each minor has k first-row elements (each element creating one unique DDD vertex), the total number of DDD vertices to be created for all first-row elements of all $k \times k$ minors is $k \cdot C(n,k)$, which is the total number of DDD vertices created for the $(n-k+1)$th layer of the digraph.

Summing over all the layers for $k = 1, 2, \ldots, n$, we obtain that the total number of DDD vertices to be constructed for an $n \times n$ full matrix in the row-wise order is (using the basic identities in combinatorics)

$$\sum_{k=1}^{n} k \cdot C(n,k) = n \sum_{k=1}^{n} C(n-1, k-1) = n \cdot 2^{n-1} \qquad (4)$$

which completes the proof. ■

The next theorem states that the DDD size obtained for a full matrix with a row (column) order is actually optimal. The proof is based on an argument which shows that the number of vertices created in each layer of the layered digraph is minimum by a row-wise (or column-wise) ordering. For this purpose, the notion of path count is used.

Theorem 2: The row-wise (or column-wise) order is optimal for any full matrix in the sense that the resulting DDD size is minimum.

We reiterate that for an $n \times n$ full matrix with any symbol order, all the minors in the $(n-k+1)$th layer of the digraph are of size $k \times k$, where $k = n, n-1, \ldots, 2, 1$, from the top

layer down to the bottom layer. The lemma next provides a technical preparation.

Lemma 1: The following properties hold for the vertices in a layered digraph constructed for an $n \times n$ full matrix:

(a) For any symbol order, the maximum number of paths arriving at any vertex in the $(n-k+1)$th layer is $(n-k)!$ and the maximum number of paths leaving from any vertex in the $(n-k+1)$th layer is $(k-1)!$ for $k = n, n-1, \ldots, 2, 1$.
(b) All digraph vertices resulting from the row-wise order have the maximum number of arriving paths and the maximum number of leaving paths.

Proof: When a path starting from a vertex at the top (first) layer of digraph reaches a vertex, say, vertex x in the $(n-k+1)$th layer, it has gone through $(n-k)$ "Minor" operations. Given any symbol order, the total number of partial paths that arrive at vertex x cannot exceed the total number of terms expanded from a full $(n-k) \times (n-k)$ minor (without zero elements). Hence, the maximal number of possible partial paths that can arrive at vertex x in the $(n-k+1)$th layer is $(n-k)!$.

Analogously, originating from the same vertex x at the $(n-k+1)$th layer and ending at one terminating vertex at layer-n (the bottom layer) of the digraph, there could be no more than $(k-1)!$ partial paths, which are all the terms obtained by expanding a remaining full $(k-1) \times (k-1)$ minor. This proves part (a) of the lemma.

With the row-wise ordering, the total number of vertices in the $(n-k+1)$th layer is $k \cdot C(n,k)$ (see the proof of Theorem 1). Since for a full $n \times n$ matrix there must be total $(n)!$ paths passing all the vertices of any given layer, the average number of paths passing each vertex in the $(n-k+1)$th layer is

$$\frac{(n)!}{k \cdot C(n,k)} = \frac{(n)!(k)!(n-k)!}{k \cdot (n)!} = (k-1)!(n-k)!. \quad (5)$$

The result of part (a) implies that there are maximum $(k-1)!(n-k)!$ paths passing each vertex in the $(n-k+1)$th layer of the digraph. Consequently, the row-wise order has realized (by equal distribution) that the maximum number of paths arrive at and leave from every vertex in each layer of the digraph, which proves part (b) of the lemma. ■

Proof of Theorem 2: If neither a row-wise order nor a column-wise order is used, then some vertices in a certain layer of the digraph might have the number of arriving paths or the number of leaving paths less than the corresponding maximum numbers stated in Lemma 1 because singular minors might have been encountered in the middle.

However, the total number of paths passing all the vertices in any digraph layer must be $(n)!$ for an $n \times n$ full matrix. Therefore, more DDD vertices than the minimum must be created in that layer to accommodate the total number of $(n)!$ paths. Consequently, the total number of digraph vertices, which is equal to the DDD size, must be greater than the minimum size $(n \cdot 2^{n-1})$.

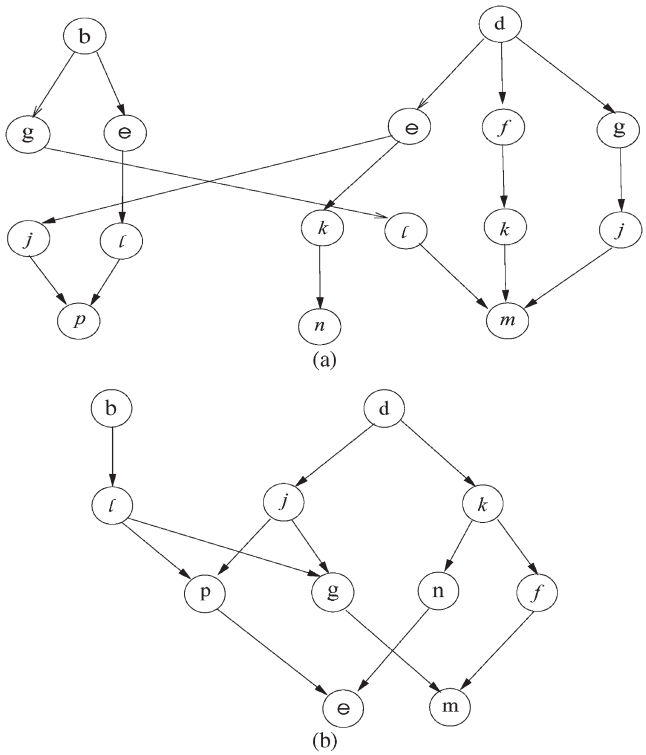

Fig. 2. (a) Layered digraph of the 4×4 sparse matrix with a column-wise order given in (6). (b) Optimal layered digraph with the order given in (7).

IV. Discussion

The DDD complexity measure established in the preceding section for full matrices can be used for several purposes. For example, it helps us gain knowledge of how much storage is compressed by using the DDD technique instead of direct enumeration. It also helps us quantize the complexity difference between using an optimal order for full matrices and using a nonoptimal heuristic such as the Greedy-Labeling order proposed in [2] for general sparse matrices.

First, we point out via an example that neither a row-wise nor a column-wise order is necessarily optimal for a sparse matrix. The 4×4 matrix

$$\begin{vmatrix} 0 & e^{(3)} & 0 & m^{(9)} \\ b^{(1)} & f^{(4)} & j^{(6)} & n^{(10)} \\ 0 & g^{(5)} & k^{(7)} & p^{(11)} \\ d^{(2)} & 0 & \ell^{(8)} & 0 \end{vmatrix} \quad (6)$$

is ordered column-wise. Shown in Fig. 2(a) is the DDD digraph constructed with the column-wise symbol order. The digraph has 16 vertices. However, another order given below

$$\begin{vmatrix} 0 & e^{(10)} & 0 & m^{(11)} \\ b^{(1)} & f^{(8)} & j^{(5)} & n^{(9)} \\ 0 & g^{(6)} & k^{(4)} & p^{(7)} \\ d^{(2)} & 0 & \ell^{(3)} & 0 \end{vmatrix} \quad (7)$$

gives rise to the digraph shown in Fig. 2(b) where the number of vertices is 11, which is optimal because the number of symbols in the matrix is also 11. However, the order given in (7) is neither row-wise nor column-wise.

TABLE I
COMPARISON OF THE DDD SIZES PRODUCED BY THE *ROW-WISE* ORDER AND BY THE *GREEDY-LABELING* FOR FULL MATRICES

Matrix size	2	3	4	5	6	7	8
$\|DDD\|$ (Rowwise)	4	12	32	80	192	448	1,024
$\|DDD\|$ (Greedy)	4	13	40	118	340	965	2,708
Matrix size	9	10	11	12	13	14	15
$\|DDD\|$ (Rowwise)	2,304	5,120	11,264	24,576	53,248	114,688	245,760
$\|DDD\|$ (Greedy)	7,535	20,828	57,266	156,764	427,571	1,162,580	3,152,681
Matrix size	16	17	18				
$\|DDD\|$ (Rowwise)	524,288	1,114,112	2,359,296				
$\|DDD\|$ (Greedy)	8,529,668	23,030,492	62,072,002				

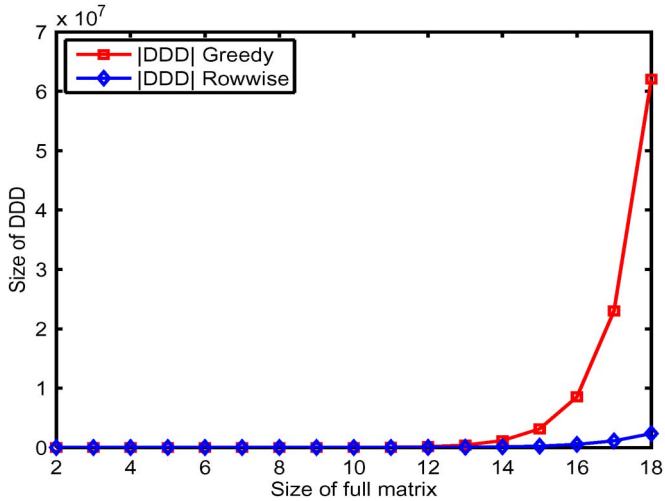

Fig. 3. Comparison of the DDD sizes by the row-wise order and by the greedy order for full matrices.

An optimal order for a general sparse matrix is unknown. Shi and Tan [2] proposed a Greedy-Labeling heuristic for sparse matrices. This ordering heuristic is a dynamic ordering scheme in that the symbol order is determined in the process of determinant expansion according to the minimum degree rule: by counting the row and column degrees (i.e., the number of nonzeros) of an intermediate minor, a minimum-degree row (respectively column, with priority upon a tie) is selected and the in-row (respectively in-column) elements are sorted in the increasing column (respectively row) degrees.

The order produced by the Greedy-Labeling for the 3×3 full determinant is given by

$$\begin{vmatrix} a^{(1)} & d^{(4)} & g^{(5)} \\ b^{(2)} & e^{(6)} & h^{(8)} \\ c^{(3)} & f^{(7)} & i^{(9)} \end{vmatrix} \tag{8}$$

which is neither row-wise nor column-wise. The DDD constructed with this order is of size 13, while the optimal DDD size in row-wise order is $n \cdot 2^{n-1} = 3 \times 2^2 = 12$.

The DDD size given by the Greedy-Labeling for the 4×4 sparse matrix in (6) is 12, but the optimal DDD size for this matrix is known to be 11. Hence, the Greedy-Labeling heuristic is not necessarily optimal for general sparse matrices.

Listed in Table I are the DDD sizes for a sequence of full matrices, which are counted by a DDD calculator program. The DDD calculator implements both the row-wise (column-wise) ordering and the Greedy-Labeling. The sizes marked "Row-wise" in Table I are exactly equal to that predicted by the complexity result of Theorem 1. The sizes marked by "Greedy" are counted by the program. The rapidly growing complexity difference is clearly seen from the data shown in Fig. 3.

Although the Greedy-Labeling is nonoptimal for both full matrices and sparse matrices in general, it was found to be a good heuristic for practical circuit problems in our experimental tests.

V. CONCLUSION

An optimal symbol order for the DDD construction of full matrices is discovered, and the optimal DDD size is derived to be $(n \cdot 2^{n-1})$ for an n-dimensional full matrix. An optimal order and the optimal DDD size for a general sparse matrix is not available at the moment. It is expected that the complexity growth rate would be much lower than two for most sparse matrices arising from circuit problems, while a true growth factor must be dependent on the matrix sparsity pattern. The results developed in this brief also reveal that a symbol order is the key factor that determines the complexity compression ratio achievable by using a DDD. In this spirit, although the Greedy order is an empirically good heuristic for practical circuits, it does not preclude the existence of other better ordering mechanisms for general circuit matrices.

REFERENCES

[1] H. Yang, A. Agarwal, and R. Vemuri, "Fast analog circuit synthesis using multi-parameter sensitivity analysis based on element-coefficient diagrams," in *Proc. IEEE Comput. Soc. Annu. Symp. VLSI*, Tampa, FL, 2005, pp. 71–76.

[2] C. J. R. Shi and X. D. Tan, "Canonical symbolic analysis of large analog circuits with determinant decision diagrams," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 19, no. 1, pp. 1–18, Jan. 2000.

[3] R. E. Bryant, "Graph-based algorithms for Boolean function manipulation," *IEEE Trans. Comput.*, vol. C-35, no. 8, pp. 677–691, Aug. 1986.

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

[5] W. Verhaegen and G. E. Gielen, "Efficient DDD-based symbolic analysis of linear analog circuits," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 49, no. 7, pp. 474–487, Jul. 2002.

[6] S. X. D. Tan and C. J. R. Shi, "Efficient approximation of symbolic expressions for analog behavioral modeling and analysis," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 23, no. 6, pp. 907–918, Jun. 2004.

[7] S. X. D. Tan, "Symbolic analysis of analog integrated circuits by Boolean logic operations," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 53, no. 11, pp. 1313–1317, Nov. 2006.

[8] E. Tlelo-Cuautle, C. Sánchez-López, E. Martínez-Romero, and S. X. D. Tan, "Symbolic analysis of analog circuits containing voltage mirrors and current mirrors," *Analog Integr. Circuits Signal Process.*, Feb. 4, 2010. [Online]. Available: www.springerlink.com

[9] S. Minato, "Zero-suppressed BDD's for set manipulation in combinatorial problems," in *Proc. 30th IEEE/ACM Des. Autom. Conf.*, Dallas, TX, 1993, pp. 272–277.