Balanced Multi-Level Multi-Way Partitioning of Large Analog Circuits for Hierarchical Symbolic Analysis *

Xiang-Dong Tan and C.-J. Richard Shi
Department of Electrical Engineering
University of Washington
Seattle, WA 98195, USA

Abstract: Symbolic analsis of analog circuits is important in analog design automation. However, it is limited to the analysis of small analog circuits where exact symbolic expressions are required. In this paper, we present an efficient algorithm for partitioning large general analog circuits into smaller subcircuits so that symbolic analysis can be performed hierarachically. Experimental results have demontrated that our method ourperforms the best partitioning-based symbolic analyzer SCAPP.

1. Introduction

Symbolic analysis calculates the behavior or the characteristics of a circuit in terms of symbolic parameters. But symbolic analysis generally suffers the circuit-size limitation problem due to the exponential growth of the symbolic terms with the increase of circuit size. One way to cope with the size limitation problem in symbolic analysis is by means of hierarchical decomposition.

Hierarchical decomposition generates symbolic expressions in a nested form [4, 8]. However, the effectiveness of hierarchical decomposition depends crucially on how a circuit is partitioned. So far no effective partitioning algorithm for symbolic analysis has been reported on such less regular-structured circuits as $\mu A741$ Opamp [3].

In this paper, we present a balanced multi-level multi-way partitioning heuristic for hierarchical symbolic analysis of large analog circuits [8]. It takes advantage of both hierarchical decomposition and a recently introduced graph-based representation, called Determinant Decision Diagrams (DDDs), for symbolic determinants [6].

The rest of the paper is organized as follows: Section 2 provides an overview of DDD-based hierarchical symbolic analysis. Section 3 formulates the partitioning problem for symbolic analysis. Section 4 presents our partitioning heuristic. Section 5 presents experimental results. Section 6 concludes the paper.

2. DDD-based Hierarchical Symbolic Analysis

For a linear(ized) time-invariant analog circuit, it is well-known that its system equation can be formulated by modified nodal analysis in the general matrix form: $\mathbf{Ax} = \mathbf{b}$, where $\mathbf{x}$ is a vector of the node voltage variables and branch current variables, $\mathbf{A}$ is the modified nodal admittance matrix and $\mathbf{b}$ represents the independent sources.

If we partition the circuit into two device-disjointed parts, the variables $\mathbf{x}$ will be divided into three disjoint groups: $\mathbf{x}^I$, $\mathbf{x}^B$, and $\mathbf{x}^R$, where the sup-scripts *I, B, R* stand for, respectively, internal variables, boundary variables and the *rest* of variables. Then the system-equation set can be rewritten in the following form:

$$\begin{bmatrix} \mathbf{A}^{II} & \mathbf{A}^{IB} & \\ \mathbf{A}^{BI} & \mathbf{A}^{BB} & \mathbf{A}^{BR} \\ & \mathbf{A}^{RB} & \mathbf{A}^{RR} \end{bmatrix} \begin{bmatrix} \mathbf{x}^I \\ \mathbf{x}^B \\ \mathbf{x}^R \end{bmatrix} = \begin{bmatrix} \mathbf{B}^I \\ \mathbf{B}^B \\ \mathbf{B}^R \end{bmatrix}. \quad (1)$$

The basic idea that underlines all the hierarchical analysis methods is to eliminate some equations from the equation-set above until the remaining set of equations involves only the desired variables. The resulting set of equations (1) by eliminating $\mathbf{x}^I$ can be written as follows:

$$\begin{bmatrix} \mathbf{A}^{BB*} & \mathbf{A}^{BR} \\ \mathbf{A}^{RB} & \mathbf{A}^{RR} \end{bmatrix} \begin{bmatrix} \mathbf{x}^B \\ \mathbf{x}^R \end{bmatrix} = \begin{bmatrix} \mathbf{B}^{B*} \\ \mathbf{B}^R \end{bmatrix}, \quad (2)$$

where

$$\mathbf{A}^{BB*} = \mathbf{A}^{BB} - \mathbf{A}^{BI}(\mathbf{A}^{II})^{-1}\mathbf{A}^{IB}, \quad (3)$$

and

$$\mathbf{B}^{B*} = \mathbf{B}^{B} - \mathbf{A}^{BI}(\mathbf{A}^{II})^{-1}\mathbf{B}^{I}. \quad (4)$$

In our application, $\mathbf{b}^I$ is a zero vector, we rewrite (3) in the following expanded form:

$$a^{BB*}_{u,v} = a^{BB}_{u,v} - \frac{1}{det(\mathbf{A}^{II})} \sum_{k_1,k_2=1}^{l} a^{BI}_{u,k_1} \Delta^{II}_{k_2,k_1} a^{IB}_{k_2,v}. \quad (5)$$

where $u, v = 1, ..., k$, k is size of $\mathbf{A}^{BB*}$, l is the size of $\mathbf{A}^{II}$, $det(A^{II})$ is the determinant of matrix A^{II}, $a^{BB*}_{u,v}$ is the entry at row u and column v in $\mathbf{A}^{BB*}$, $\Delta^{II}_{k_2,k_1}$ is the first order cofactor of A^{II} defined as $(-1)^{(k_2+k_1)}det(A^{II}_{k_2,k_1})$ and $A^{II}_{k_2,k_1}$ is obtained by eliminating row k_2 and column k_1 in A^{II}.

Note that we need first-order cofactors $\Delta^{II}_{k_2,k_1}$ only when a_{u,k_1} and $a_{k_2,v}$ are both non-zeros. In practice a_{u,k_1} and $a_{k_2,v}$ are zero for most time due to the sparsity of A^{BI} and A^{IB}, provided a good partitioning is given. The determinant $det(\mathbf{A}^{II})$ and a few of its required first-order cofactors can be efficiently represented by a newly proposed special graph, Determinant Decision Diagrams, described as follows.

A DDD example for a determinant is shown in Fig. 1. Its determinant is shown in the left-hand side. Except for two special

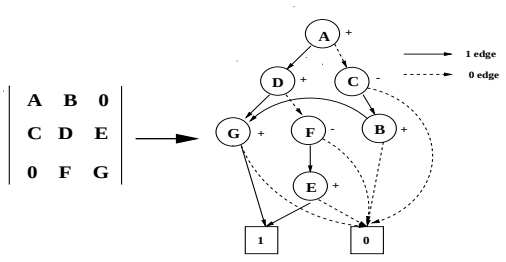


Figure 1: A matrix determinant and its DDD.

terminal vertices, namely the *0-terminal* vertex and the *1-terminal* vertex, each non-terminal vertex is labeled by a symbol in the determinant denoted by a_i, and a sign denoted by $s(a_i)$. It *originates* two outgoing edges, called *1-edge* and *0-edge*. Each vertex a_i represents a symbolic expression $D(a_i)$ defined recursively as follows: $D(a_i) = a_i\, s(a_i)\, D_{a_i} + D_{\overline{a}_i}$. where D_{a_i} and $D_{\overline{a}_i}$ represent, respectively, the vertices pointed by the 1-edge and 0-edge of a_i. A *1-path*

*This work was sponsored by U.S. Defense Advanced Research Projects Agency (DARPA) under grant number F33615-96-1-5601 from the United States Air Force, Wright Laboratory, Manufacturing Technology Directorate, by Rockwell Semiconductor Systems, and by NSF/Industry Center for Design of Analog-Digital Integrated Circuits (CDADIC).

in a DDD corresponds a product term in the original DDD, which is defined as a path from the root vertex (A in our example) to the 1-terminal including all symbolic symbols and signs of the vertices that originate all the 1-edges along the 1-path. In our example, there exist three 1-paths representing three product terms: ADG, $-AFE$ and $-CBG$. The root vertex represents the sum of these product terms.

3. Problem Formulation

An analog circuit can be modeled as a hypergraph $H(V, E)$ with vertex set $V = \{v_1, v_2, ..., v_m\}$ and edge set $E = \{e_1, e_2, ..., e_n\}$. Each vertex corresponds to a circuit device and each edge represents a node or net in the circuit. An edge e_j also defines a subset of V with the vertices incident on the net and $|e_j| \geq 2$. The k-way partitioning problem is to assign $v_i, i = 1, ..., m$ into k disjoint subsets $V_1, V_2, ..., V_k$. If $k > 2$, it is a *multi-way* partitioning. If $V_j, j = 1, ..., k$ are further decomposed into smaller subcircuits, it is called *multi-level* or *tree* partitioning, otherwise it is called *two-level* partitioning. The subcircuit without further subcircuit definitions is a *leaf* subcircuit, otherwise it is a *middle* subcircuit. We further define the *span* of a net e_j, denoted as $span(e_j)$, as the number of the subcircuits e_j connects. If $span(e_j) > 1$, e_j is called *cut* net, otherwise it is an *internal* net.

The most important performance criterion in hierarchical symbolic analysis is how compact the generated symbolic expressions are. In our case, such requirement amounts to reducing DDD vertices used. From equation (5), we observe that the problem of reducing the DDD vertices in partitioning actually centers around reducing the number of the required first-order cofactors $\Delta^{II}_{k_2,k_1}$ and eventually the nonzero elements in the remaining circuit. This implies that nonzero entries $a^{BI}_{x,y}$ and $a^{IB}_{x,y}$ due to *boundary* variables $\mathbf{x}^B$ should be as few as possible. Boundary variables essentially are *cut nets* in nodal analysis formulation. So the total number of the cut nets in all the subcircuits should to be minimized. The partitioning objective can then can be expressed as:

$$min \sum_{e_j \in E} span(e_j) \tag{6}$$

To further reduce the DDD size, we also need to balance the numbers of internal nets among different subcircuits. This is due to the fact that min-cut type partitioning always gives rise to very dense interconnection within the leaf subcircuits and therefore dense circuit matrices. For a dense or full matrix, DDD representation is exponential. Suppose that the DDD size only depends on the size of determinant, then the minimal DDD size is achieved if all the determinants of subcircuits have the same size. Therefore, it is desirable to balance the number of the internal nets. For large analog circuits, each subcircuit size should also be bounded to efficiently reduce the overall DDD size. Therefore the multi-level partitioning becomes a must. Let S denote the set of all cut nets e_j, i.e. $span(e_j) > 1$. The balance requirements can be expressed as following constraints:

$$\text{both:} \qquad \frac{|E| - |S|}{k} - \tau \leq \mathrm{I}(V_i) \leq \frac{|E| - |S|}{k} + \tau \tag{7}$$

$$\text{tree:} \qquad \mathrm{I}(V_i) \leq \alpha |E| \tag{8}$$

for $i = 1, ..., k$. $\mathrm{I}(V_i)$ is the set of internal nets in V_i, i.e. $\mathrm{I}(V_i) = \{e_j | e_j \subseteq V_i \text{ and } span(e_j) = 2\}$, τ is the measure of the offset from its balanced size (referred to as $deviationfactor$), and α is a positive constant and $0 < \alpha < 1$.

4. Partitioning Algorithm

4.1. Multi-Way Balanced Partitioning

In this section, we describe an extention of the vertex moving partitioning heuristic due to Fiducciai and Mattheyses(FM) [2] for multi-way balanced partitioning.

The FM's algorithm begins with two initial partitions and proceeds in a series of *passes*. In each pass, it keeps moving vertices between two partitions until each vertex has been moved exactly once. After each pass, the best solution observed during the pass becomes the initial solution for a new pass. During each pass, the moved vertices are *locked* from further exchanging. The unmoved vertices are called *free* vertices. The pass terminates when a pass does not improve upon the most recent solution.

For the convenience, subcircuit, partition and subset are used interchangeablely in the sequel. We first focus on the two-level, multi-way balanced partitioning. Then we extend it to tree partitioning.

4.2. Computation of Gain and Potential Gain

Unless otherwise specified, we assume that a vertex c is moved from subset A to subset B, A and B are two subsets mong k subsets and $c \in A$. We further define *incident number* of a net e with respect to a set of vertices A denoted by $\alpha_A(e)$ as:

$$\alpha_A(e) = |\{c | c \in A \text{ and } c \in e\}| \tag{9}$$

A *binding force* of a net with respect to the set A, denoted by $\beta(e)$, is defined as:

$$\beta_A(e) = \begin{cases} \alpha_{A_F}(e) & \text{if } \alpha_{A_L}(e) = 0 \\ \infty & \text{if } \alpha_{A_L}(e) > 0 \end{cases} \tag{10}$$

where A_F and A_L denote the subsets that contain all the free vertices and locked vertices in A, respectively.

In order to efficiently calculate the $span(e_j)$ of a net e_j, we divide the move operation of c into two steps:

1. Vertex c is selected from A and put into $\overline{A}$ ($\overline{A}$ is complement of A, $\overline{A} = V - A$). The corresponding net cut gain, denoted by $G^A_{get}(c)$, can be written as:

$$\begin{aligned} G^A_{get}(c) \quad = \quad & |\{e \in E_c | \beta_A(e) = 1 \text{ and } \beta_{\overline{A}}(e) > 0\} \\ & -\{e \in E_c | \beta_A(e) > 0 \text{ and } \beta_{\overline{A}}(e) = 0\} \end{aligned} \tag{11}$$

   where E_c denotes the set of nets incident on vertex c.
2. Vertex is moved from $\overline{B}$ to B. Let $G^B_{put}(c)$ denote the gain of this step. Then

$$\begin{aligned} G^B_{put}(c) \quad = \quad & |\{e \in E_c | \beta_{\overline{B}}(e) = 1 \text{ and } \beta_B(e) > 0\} \\ & -\{e \in E_c | \beta_{\overline{B}}(e) > 0 \text{ and } \beta_B(e) = 0\} \end{aligned} \tag{12}$$

The total gain of the move operation of c from A to B is given by

$$G_{span}(c) = G^A_{get}(c) + G^B_{put}(c) \tag{13}$$

$G_{span}(c)$ acutally is the total decrease in spans due to the move operation of the c.

To ensure that constraint (7) is always satisfied after a moving operation for both subset A and B, we need to check the following constraint at each moving step:

$$\begin{aligned} & \frac{|E| - |S| - G_{cut}(c)}{k} - \tau \leq |\mathrm{I}(A)| - |I_A(c)| \\ & \qquad \leq \frac{|E| - |S| - G_{cut}(c)}{k} + \tau \text{ for set A} \end{aligned} \tag{14}$$

$$\begin{aligned} & \frac{|E| - |S| - G_{cut}(c)}{k} - \tau \leq |\mathrm{I}(B)| + |S_B(c)| \\ & \qquad \leq \frac{|E| - |S| - G_{cut}(c)}{k} + \tau \text{ for set B} \end{aligned} \tag{15}$$

where $I_A(c)$ is the set of all the internal nets of A in E_c,i.e.,

$$I_A(c) = \{e \in E_c | \beta_A(e) > 0 \text{ and } \beta_{\overline{A}} = 0\} \tag{16}$$

and $S_B(c)$ is the set of all the cut nets in E_c, and it becomes an internal net of B after the move,i.e,

$$S_B(c) = \{e \in E_c | \beta_{\overline{B}}(e) = 1 \text{ and } \beta_B > 0\} \tag{17}$$

One way to improve the partitioning quality is to break the tie situation where two vertices have same gain [5]. In our method, we introduce a new penalty function, which is devised for multi-way partitioning, into each vertex gain as the potential gain. Let $G_P(e)$ denote the potential gain of net e imposed on its incident vertices. The potential gain also consists of two parts corresponding to the two steps in a move operation:

$$G_P(e) = G^A_{Pget}(e) + G^B_{Pput}(e) \tag{18}$$

where

$$G^A_{Pget}(e) = \begin{cases} 0 & \text{if } \beta_A(e) = |e| \text{ or } \beta_A(e) = 0 \\ -P & \text{if } \beta_A(e) = \infty \\ -P\,W(e)\frac{\alpha_A(e)}{|e|} & \text{if } 0 < \beta_A(e) < |e| \end{cases} \tag{19}$$

$$G^B_{Pput}(e) = \begin{cases} 0 & \text{if } \beta_B(e) = |e| \text{ or } \beta_B(e) = 0 \\ P & \text{if } \beta_B(e) = \infty \\ P\,W(e)\frac{\alpha_B(e)}{|e|} & \text{if } 0 < \beta_B(e) < |e| \end{cases} \tag{20}$$

where $W(e) = \frac{|e|}{D_{max}}$ if $|e| < D_{max}$ and $W(e) = 1$ otherwise and D_{max} is the prespecified upper bound on the number of nets incident on a vertex, P is a constant. Both $G^A_{Pget}(e)$ and $G^B_{Pput}(e)$ favor vertices incident on the cut nets which likely become internal nets of B after the vertex move operation. Then, the total gain for whole move operation can be expressed as:

$$G_{AB}(c) = G_{span}(c) + \sum_{e \in E_c} (G^A_{Pget}(e) + G^B_{Pput}(e)) \tag{21}$$

4.3. Relaxation of Balance Constraints

One issue with FM-based algorithms is that moving a vertex is only feasible if the move operation dose not violate the balance constraints involved. However this will confine the solution space especially when the constraint is strict [1]. This problem can be alleviated by temporally relaxing the balance constraints and allowing a sequence of move operations of vertices, called *macro-step*, to be carried out as long as the balance constraints are restored after the macro-step.

More specifically, Let MG[i] and IN[i] be the maximum gain of all vertices and number of internal nets in subset i. Consider a vertex c_v to be moved from subset V_A to subset V_B without considering the balance constraint. We observe that after the move operation, IN[A] is always reduced, and IN[B] is always increased. So in case the move operation causes the violations of balance constraints in either A or B, or both, the replacement strategy that involves a sequence of vertex move operations can be used to restore the balance. Let F be the set of free vertices. The new balance-relaxed multi-way partitioning algorithm (BRMP) is described in Fig. 2.

4.4. Multi-way Tree Partitioning

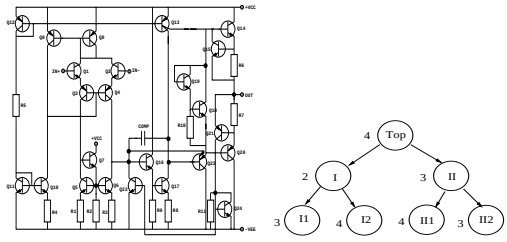
Two-level partitioning can be extended to tree partitioning by recursively applying the algorithm to each subcircuit obtained from previous partitioning and decomposing each subcircuit into smaller parts until the constraint (8) on the number of the internal nets of subcircuits are satisfied. Such tree partitioning scheme can be viewed as a special two-level, multi-way partitioning where each two-level mutlti-way partitioning is solved by only allowing vertics moved among some of subsets and all the vertices in the other subsets are locked during the entire process of partitioning. So the gain calculations (21) and internal-net constraints (19) and (20) still remain valid for tree partitioning.

5. Experimental Results

The proposed balanced multi-level multi-way partitioning algorithm has been implemented in C++ and integrated into our hierarchical symbolic analyzer [6, 8]. Before symbolic analysis, all the BJT and MOS transistors in a circuit are replaced by their corresponding

BRMP ALGORITHM(F)
  while($|F| \neq 0$) do /* macro-step */
1 Move a free vertex c_v with the highest gain from V_A to V_B without considering constraint (15) and (14), where MG[A] is the largest among all the subsets.
2 while (constraint (15) is violated) do
       move a free vertex c_x with the highest gain out of V_B subject to constraints (15) and (14).
3 while (constraint (14) is violated) do
       move a free vertex c_x with the hightest gain from V_k into V_A subject to constraints (15) and (14), where $IN[k]$ is the largest among all subsets except V_A.
4 Recursively repeat steps (2) and (3) until there are no violations of balance constraints in all the subsets.
5 Lock all the moved vertices and add them into current macro-step; record the current span size.

Figure 2: Balance-relaxed multi-way partitioning.



(a) $\mu A741$ Opamp (b) Three-level, 2-way partitioning for $\mu A741$

Figure 3: $\mu A741$ Opamp

SPICE small-signal models at their DC operating points computed by SPICE. The results from bipolar Opamp $\mu A741$, which has 26 transistors and 11 resistors shown in Fig. 3(a) is presented here.

We first perform a two-level multi-way partitioning of $\mu A741$. The total number of nets for $\mu A741$ is 23. For the small-signal AC analysis, the power and ground nets are the reference node in the nodal formulation method. They will not appear in the circuit matrix, and are ignored by partitioning. All the nets corresponding to the circuit inputs and outputs are always cut nets. We select the *deviation factor*—τ to be 2. Figures 4(a), 5(a) and 5(b) show, respectively, the results of 2-way, 3-way and 4-way balanced partitionings of $\mu A741$, where each partitioned subcircuit is marked by an index (I up to IV). Table 1 summarizes the statistics of these partitionings. Columns 1, 2 and 3 list, respectively, the number of subcircuits, the span/number of cut nets, and the number of nets in a balanced subcircuit. Columns 4 to 7 list, respectively, the number of internal nets in each subcircuit and its corresponding DDD size. Column 8 gives the number of DDD vertices in the top-level circuit, $|topD|$, where the last column describes the total number of DDD vertices, $|totalD|$. We have the following observations:

- The total number of DDD vertices used for representing all the subcircuits decreases from 837 in the balanced 2-way partitioning to 237 in the balanced 4-way partitioning. Meanwhile, the number of DDD vertices used for the top-level circuit increases from 20 to 114. The total number of DDD vertices decreases as k increases from 2 to 4. However, if we further increase k, more nets will be cut and the size of the DDD for the top-level circuit will increase rapidly, and dominate the overall DDD size.
- As we have reported in [6], without partitioning and hierarchical symbolic analysis, the total number of DDD vertices

to represent the $\mu A741$ is 7431, where the number of product terms in fully-expanded classical symbolic analysis is 374884. Thus, hierarchical symbolic analysis with automatic balanced partitioning leads to a very compact behavioral model (237 vs 7431)! Since the number of multiplications is linear in the DDD size, hierarchical symbolic analysis with automated partitioning speeds up canonical symbolic analysis (without partitioning) by a factor of 31, where canonical symbolic analysis already speeds up fully-expanded symbolic analysis by several orders of magnitude (7431 multiplications vs 374884 product terms)

Table 1: Statistics of 2-level, multi-way partitionings of $\mu A741$.

k	#sp/#cut	$\frac{\|E\|-\|S\|}{k}$	#internal nets/$\|DDD\|$				$topD$	$totalD$
			I	II	III	IV		
2	8/4	9	9/260	10/557	-	-	20	837
3	13/7	5	5/29	6/79	5/55	-	80	241
4	18/18	3	3/5	2/5	5/53	5/57	114	237

We further perform a 3-level 2-way partitioning of $\mu A741$ based on the hierarchical tree shown in Fig. 3(b). The third-level partitioning is based on the two-level two-way partitioning shown in Fig. 4(a). The resulting partitioning is shown in Fig. 4(b). Table 2 summarizes

Table 2: Statistics of 3-level,2-way partitioning of $\mu A741$.

leaf subcircuits	I1	I2	II1	II2
$\|DDD\|$	10	36	23	6
#internal nets	3	4	4	3

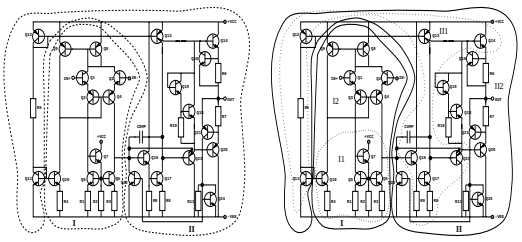
middle subcircuits	I	II
$\|DDD\|$	4	18
#internal nets	2	3
span	8	10
#cut nets	5	6

top-level span	6
top-level #cut nets	4
top-level $\|DDD\|$	20
total $\|DDD\|$	117
total $\|DDD\|$ (w/o)	7431
SCAPP	#mul:182, #add:357

the partitioning statistics. Rows 2 and 3 describe the DDD sizes and the numbers of internal nets in the leaf subcircuit $I1$, $I2$, $II1$, $II2$. Rows 4 to 7 list, respectively, the DDD size, the numbers of internal nets, spans and the number of cut nets in the two middle circuits I and II. Note that internal nets in a middle circuit level are the cut nets in the lower level and are invisible to the parent of the current subcircuit. Rows *top-level span* and *top-level #cut nets* are the number of spans and the number of cut nets at the top-level circuit. Rows *#top-level* $|DDD|$ and *total* $|DDD|$ show the corresponding DDD size for the top-level circuit and the total DDD size. Row *total* $|DDD|$ *(w/o)* is the DDD size without partitioning. We can see that hierarchical symbolic analysis with automated 3-level 2-way partitioning reduces the number of DDD vertices from 7431 to 117! Since only one multiplication is needed for one vertex, the total number of multiplications is 117. To compare with best-known hierarchical symbolic analyzer—SCAPP [4], row *SCAPP* lists the best result from SCAPP with automated partitioning, where $\#mul$ and $\#add$ are numbers of multiplications and additions, respectively.

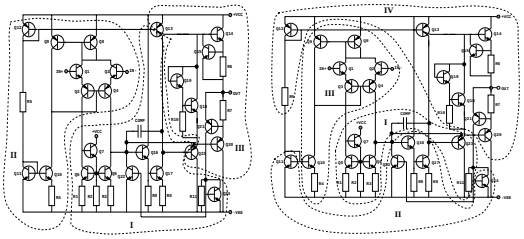
6. Conclusions

An efficient algorithm for balanced multi-way multi-level partitioning of very large analog circuits is presented for hierarchical symbolic analysis. It is based on the iterative vertex moving heuristic due



(a) 2-way partitioned $\mu A741$ Opamp (b) 3-level, 2-way partitioned $\mu A741$.

Figure 4: 2-way partitioned $\mu A741$ Opamp



(a) 3-way partitioned $\mu A741$ Opamp (b) 4-way partitioned $\mu A741$ Opamp

Figure 5: Multi-way partitioned $\mu A741$ Opamp

to Fiduccia and Matheyses [2]. The novelties are the introduction of a new formula for potential gain computation in multi-way partitioning and the relaxation of the balance constraints. These ideas have been explored for digital circuit layout partitioning recently and shown to be superior to other related methods [9]. In this paper, we have described an application to analog circuit partitioning for symbolic analysis and shown its advantage over the best analog symbolic analysis program SCAPP.

References

[1] J. Cong, L. Hagen and A. Kahng, " Net partitioning yield better module partitions", 1992, in *Proc. 29th IEEE/ACM Design Automation Conference*, pp. 47-52, 1992.

[2] C.M. Fiduccia and R.M. Mattheyses, "A linear time heuristic for improved network partitions", *Proc. 19th ACM/IEEE Design Automation Conference*, pp.175-181. 1982.

[3] G. Gielen, P. Wambacq and W. Sansen, "Symbolic analysis methods and applications for analog circuits: A tutorial overview", *Proc. IEEE*, vol. 82, no. 2, pp. 287–304, Feb. 1994.

[4] M. M. Hassoun and P. M. Lin, "A hierarchical network approach to symbolic analysis of large scale networks", *IEEE Trans. Circuits and Systems*, vol. 42, no. 4, pp. 201–211, April 1995.

[5] B. Krishnamurthy, "An improved min-cut algorithm for partitioning VLSI networks", *IEEE Trans. on Computers*, vol. C33, no. 5, pp.438-446, 1984.

[6] C.-J. Shi and X.-D. Tan, "Symbolic analysis of large analog circuits with determinant decision diagrams", in *Proc. IEEE Int. Conf. Computer Aided Design (ICCAD)*, pp.366-373, 1997.

[7] L. A. Sanchis, "Multiple-way network partitioning", *IEEE Trans. on Computers*, vol. C38, no. 1, pp.62-81, 1989.

[8] X.-D. Tan and C.-J. Shi, "Hierarchical symbolic analysis of large analog circuits with determinant decision diagrams", in *Proc. IEEE Int. Symp. Circuits and Systems*, CA, May 1998.

[9] X.-D. Tan, J. Tong, P. Tang and F. Lombardi, "An efficient multi-way algorithm for balanced partitioning of VLSI circuits", *IEEE Int. Conf. Computer Design (ICCD'97)*, pp.608-613, 1997.