Acyclic Multi-Way Partitioning of Boolean Networks

Jason Cong, Zheng Li, and Rajive Bagrodia

Department of Computer Science
University of California, Los Angeles, CA 90024

Abstract

Acyclic partitioning on combinational boolean networks has wide range of applications, from multiple FPGA chip partitioning to parallel circuit simulation. In this paper, we present two efficient algorithms for the acyclic multi-way partitioning. One is a generalized FM-based algorithm. The other is based on the theory of maximum fanout-free cone (MFFC) decomposition. The acyclic FM-algorithm usually results in larger cut-size, as expected, compared to the undirected FM-algorithm due to the acyclic constraint. To our surprise, however, the MFFC-based acyclic partitioning algorithm consistently produces smaller (50% on average) cut-sized solutions than the conventional FM-algorithm. This result suggests that considering signal directions during the process can lead to very natural circuit decomposition and clustering, which in turn results in better partitioning solutions. We have also implemented parallel gate level simulators in Maisie and applied our partitioning algorithms to evaluate their impact on circuit simulation.

1. Introduction

Partitioning a large VLSI network into two or multiple blocks with roughly equal sizes to minimize the number of interconnections between the blocks is a very important problem in VLSI design. It has many applications ranging from circuit layout to logic simulation and emulation. The existing algorithms in the literature can be grouped into two-way algorithms and multi-way algorithms, depending on the number of blocks in the resulting partitioning solution.

The objective of two-way partitioning is to either minimize the cut-size when partitioning the network into two (roughly) equal-size blocks, or to minimize the ratio-cut size between the two blocks [WeCh89]. The two-way partitioning algorithms include the iterative improvement methods [KeLi70, Kr84, KiGV83], the graph spectrum method [Ba82, Bo87, HaKa91], and the net-based partitioning method [HaKa92b, CoHK92]. The multi-way partitioning algorithms include the recursive two-way partitioning method [KeLi70], the generalization of the FM-algorithm with lookahead scheme [Sa89], and a generalization of the spectrum-based partitioning method [ChSZ93]. A number of techniques have been introduced recently to further improve the quality of partitioning solutions, including cluster-based partitioning methods [CoHK91, HaKa92, YeCL92, CoSm93], partitioning with module duplication [KrNe91, HwGa92] and communication-complexity based partitioning [BeSa93].

Most of existing partitioning methods model a network as an undirected graph or hypergraph, and ignore the signal directions during the partitioning process. (Few exceptions include the work on partitioning with cell replication [KrNe91, HwGa92], the communication-complexity based partitioning algorithm [BeSa93], and the uni-directional two-way partitioning method [ImPF93]). However, the study in this paper shows that considering signal directions is very helpful in identifying the underlining circuit structure, which can lead to significant improvement on the partitioning results. Moreover, a number of applications impose restrictions on partitioning solutions in terms of signal directions. One common restriction is that the final partitioning solution has to be acyclic, i.e., the edges between different blocks can not form any direct cycles, which is required in the following applications:

(1) Pipelining in multi-chip designs: To implement a circuit using multiple chips (such as FPGAs), we want an acyclic partitioning solution to pipeline the final design for performance optimization.

(2) Partitioning based logic minimization: Due to the high complexity of logic minimization, we often have to partition a circuit into a set of smaller blocks and then minimize each block. Cyclic dependency among different blocks makes it very difficult to propagate Don't-Care conditions and other information between the blocks.

(3) Parallel circuit simulation: When we partition a large network to several processors, cyclic dependency among the subnetworks on different processors can cause unnecessary roll-backs when optimistic simulation strategies are used. In fact, parallel simulation is our main motivation to study the problem.

In short, our study shows that the acyclic multi-way partitioning problem is of both theoretical and practical

This work is partially supported by the California MICRO program with Cadence, Hewlett-Packard, and Zycad, by ARPA/CSTO under Contract J-FBI-93-112, and by the NSF Young Investigator Awards under MIP-9357582 and ASC-9157610.

interests. Such a partitioning scheme is especially important when we try to combine logic synthesis and layout in the VLSI design process.

In this paper, we study the acyclic multi-way partitioning problem and present two efficient algorithms. The remainder of the paper is organized as follows: Section 2 presents a general formulation of the acyclic multi-way partitioning problem. Sections 3 and 4 describe the algorithms for computing acyclic multi-way partitioning solutions. Experimental results are presented in Section 5. Conclusions and future work are presented in Section 6.

2. Problem Formulation

In this paper, we study the partitioning problem for combinational Boolean networks. A combinational Boolean network N can be represented as a directed acyclic graph where each node represents a logic gate and a directed edge (i, j) exists if the output of gate i is an input of gate j. A primary input (PI) node has no incoming edge and a primary output (PO) node has no outgoing edge. Each node i in N has an area $a(i)$. A *balanced K-way partitioning solution* $S = (A_1, A_2, ..., A_K)$ satisfies the following conditions:

(i) $A_i \bigcap A_j = \phi$ for $i \neq j$ and $\bigcup_{i=1}^{K} A_i$ contains all the gates in the network;

(ii) $(1-\alpha)\cdot \frac{|A|}{K} \leq |A_i| \leq (1+\alpha)\cdot \frac{|A|}{K}$ for each block A_i, where $|A|$ is the total area of all the gates in N, and α is a user-specified parameter controlling the slack of the area balance constraint.

Since we are always interested in the balanced partitioning solutions, the word 'balanced' might be omitted in later discussions. Our objective is to minimize the total number of nets between different partitions. Moreover, for a multi-way partitioning solution S, we can define a directed graph $D(S)$, called the *dependency graph* of S, such that each node in $D(S)$ represents a block in S, and there is a directed edge (A_i, A_j) in $D(S)$ if and only if there exists an edge (x, y) in N such that $x \in A_i$ and $y \in A_j$. We call S is an *acyclic K-way partitioning solution* if the dependency graph $D(S)$ is a directed acyclic graph. Figure 1 shows the dependency graphs of an acyclic 4-way partitioning solution and a cyclic 4-way partitioning solution, respectively.

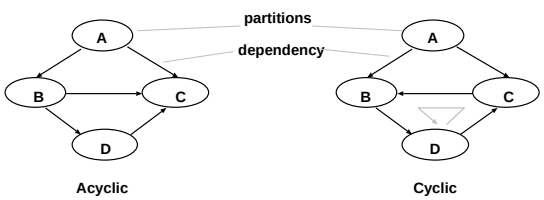

Figure 1. Dependency graphs for partitioning solutions.

The assumption that we are given a combinational network guarantees the existence of an acyclic K-way partitioning solution, when α is large enough. When we are given a general netlist, we can first remove all the sequential elements to obtain a combinational network. Next, we compute an acyclic K-way partitioning of the combinational network. Then, we assign the sequential elements to proper blocks (based on the area constraint and the connectivity information between the sequential elements and the blocks) to obtain a K-way partitioning solution of the entire network. Such a two-step approach has been used in practice, for example, for partitioning circuits into programmable logic blocks in FPGA designs. Therefore, assuming a combinational Boolean network as input in our partitioning formation does not exclude us from partitioning general Boolean networks with sequential elements.

3. FM-Based Acyclic Multi-Way Partitioning Algorithm

The FM algorithm [FiMa82] is an improvement over the two-way partitioning algorithm by Kernighan and Lin [KeLi70]. The FM algorithm starts with a balanced initial two-way partition. At each step, one gate is selected to move to the other side of the partition. Once a gate is moved, it is *locked* during the remainder of the current pass. A gate is feasible to move if moving it to the other side of the partition does not violate the balance constraint. The gate to be moved is selected from among the unlocked feasible gates with the highest gain, where the *gain* of a gate is defined to be the amount of the cut size reduced when moving the gate to the other side of the partitioning. If moving a gate would increase the cut size, the gain of that gate is negative. Moving gates with negative gains is allowed by the algorithm in order to avoid stopping at local minima in the solution space. When all gates are either locked or infeasible for moving, the current pass is completed and the best partition encountered during the pass is saved as the initial partition for the next pass. When a pass makes no improvement to the partitioning solution, the algorithm stops. An important contribution of the FM-algorithm is that it can complete each pass in linear time with respect to the number of gates and edges in the network, based on efficient gain computation and updating procedure and the use of bucket data-structure for selecting a gate with the maximum gain. The FM-algorithm has been generalized to solve the K-way partitioning problem by Sanchis. Detailed implementation and experimental results were reported in [Sa89]. We use the *K-FM algorithm* to refer to the K-way FM-based partitioning algorithm by Sanchis.

We have generalized the K-FM algorithm to solve the acyclic K-way partitioning algorithm. We denote it as K-AFM algorithm. In order to deal with the acyclic

constraint, the following enhancements have been included in the K-AFM algorithm.

3.1. Generating An Initial Acyclic Partitioning

An arbitrary random K-way partitioning can no longer be used as an initial solution since it may violate the acyclic constraint. Our study shows that an efficient way of generating an initial random acyclic partitioning solution is to use a random topological sorting solution. A topological sorting solution of a given combinational network N is a linear ordering L of all gates in N, such that gate i appears before gate j in L if the output of i is an input of j. Given a topological sorting solution L, we can partition the list L from left to right into K sub-lists such that the total gate area in each sub-list does not violate the area constraint. If we let the gates in each sub-list to form a block in the partitioning solution, it is easy to show that the resulting partitioning solution is always acyclic.

In order to use a different topological sorting solution each time, we generate a random topological sorting solution as follows: Let $sources(N)$ denote the set of gates with no inputs (originally these gates are PIs). Initially the list L is empty. At each step, we randomly select a gate i from $sources(N)$ and put it into L. Then, we remove i and its related edges from N. We repeat this process until all the gates are in L. It is not difficult to show that this procedure is able to generate any topological sorting solution. Moreover, we can show any acyclic partitioning solution can be derived from a topological sorting solution.

3.2. Maintaining The Acyclic Constraint

Before moving a gate from one block A_i to another block A_j, we shall check whether the dependency graph $D(S)$ has any directed cycle, where S corresponds to the resulting partitioning solution after the proposed move. An efficient algorithm for checking if $D(S)$ has a directed cycle is to, again, use our topological ordering algorithm. When $sources(D(S))$ becomes empty in the topological sorting procedure, but not all the nodes in $D(S)$ are removed, we stop sorting because the graph contains a cycle. Clearly the algorithm has time complexity $O(K^2)$ in the worst case. Since the value of K (number of partitions) is usually very small, the algorithm is very efficient.

3.3. The K-AFM Algorithm

The overall structure of the K-AFM algorithm is similar to the original the K-FM algorithm. In our K-AFM algorithm, a move is *feasible* if it involves a unlocked gate and if the move does not violate either the area or the acyclic constraint. A bucket-array data-structure is used for selecting a feasible move with the maximum gain. A main difference from the K-FM algorithm is that instead of maintaining K bucket-arrays, one for each block, our K-AFM algorithm uses only one global bucket-array. Efficient procedures have been developed for updating the global bucket-array after each move. The use of single bucket-array both simplifies the program and speed-ups the computation. Another difference is that the dependency graph $D(S)$ needs to be updated after each cell move. Details of these procedures are omitted here due to the length restriction.

We normally run the K-AFM algorithm multiple times with different initial partitioning solutions and select the best partitioning solution from multiple runs as the final solution.

4. MFFC Based Acyclic Multi-Way Partitioning

To further improve the quality of the partitioning result, we use a cluster-based partitioning approach to first identify and collapse clusters in the given network, and then apply the K-AFM algorithm on the condensed network. Existing cluster-based partitioning approaches have reported consistent improvement, in terms of both the cut-size and the run-time, over direct partitioning on the original network. Although a number of clustering algorithms, such as the random-walk based clustering method [CoHK91, HaKa92], the multicommodity-flow based method [YeCL92], and the clique based method [CoSm93], have been proposed, most of them do not consider the signal directions during the clustering process. As a result, the resulting clustered network often contains cycles, which makes it impossible to get an acyclic multi-way partitioning solution. In order to obtain an acyclic network after clustering, we apply the maximum fanout-free cone (MFFC) decomposition technique to compute clusters in the given network. The MFFC-decomposition technique was originally proposed by Cong and Ding [CoDi93] to compute duplication-free area-optimal technology mapping in lookup-table based FPGA designs. Our study shows that since the MFFC-decomposition considers both the signal directions and logic gate dependency relation in the given network, it not only guarantees the clustered network to remain acyclic, but also leads to a very natural clustering solution. As a result, MFFC-based clustering leads to much better partitioning solutions.

First, we introduce a few definitions related to MFFC decomposition. We use $input(v)$ to denote the set of nodes which are the fanins of node v, and $output(v)$ to denote the set of nodes which are the fanouts of node v. For a node v in the network, a *cone of v*, denoted C_v, is a subgraph of logic gates (excluding PIs) consisting of v and its predecessors such that any path connecting a node in C_v and v lies entirely in C_v. We call v the *root* of C_v. A *fanout-free cone (FFC) at v*, denoted FFC_v, is a cone of v such that for any node $u \neq v$ in FFC_v, $output(u) \subseteq FFC_v$. The *maximum fanout free cone (MFFC) of v*, denoted $MFFC_v$, is an FFC of v such that for any non-PI node w, if $output(w) \subseteq MFFC_v$, then $w \in MFFC_v$. Figure 2 shows the MFFC of each node (the smallest shadowed area including that node) in a

network. It is not difficult to show that MFFC is unique for every node, and any FFC of v is contained in $MFFC_v$. Clearly, if a gate u is in $MFFC_v$, its value affects only gate v and the descendants for gate v. Therefore, it is very natural to cluster u and v together. In general, we can treat each MFFC as a cluster. The results in [CoDi93] showed that MFFCs have the following important properties.

Lemma 1 If $w \in MFFC_v$, then $MFFC_w \subseteq MFFC_v$.

Lemma 2 Two MFFCs are either disjoint or one must contain another.

Based on these results, we can decompose a given combinational network N into a set of MFFCs as follows: Let set Q, initialized to empty, store the list of MFFCs in the final decomposition. We shall repeat the following three steps to obtain the MFFC decomposition of network N: (i) Choose an arbitrary PO v in N; (ii) Compute $MFFC_v$ and include it in Q; (ii) Set $N = N - MFFC_v$ and update the PO list of N to include the gates with outputs to some gates in $MFFC_v$. We stop this process when N is empty. Figure 3 shows the MFFC decomposition of the network shown in Figure 2. Note that MFFC decomposition is different from tree decomposition, because an MFFC can contain reconvergent fanout.

Given a combinational network N, let $MFFC_1, MFFC_2, \cdots MFFC_m$ be the set of MFFCs in the MFFC decomposition of N, the results in [CoDi93] showed that when computing a duplication-free area-optimal mapping of N into K-input lookup tables, we can simply compute a duplication-free area-optimal mapping for each $MFFC_i$ $(1 \le i \le m)$ independently. Our study shows another important property of *MFFC* decomposition, which reveals the relationship between the minimum-cut partitioning and MFFC decomposition.

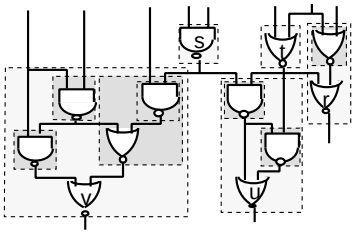

Figure 2 Maximum fanout free cones (MFFCs).

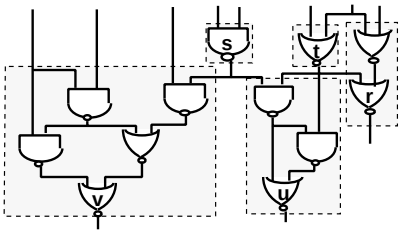

Figure 3. MFFC decomposition of Figure 2.

Theorem If we do not consider the area constraint, there is an optimal acyclic two-way partition $(X, \bar{X})$ of N, such that for each $MFFC_i$ in the MFFC decomposition of N, either $X \bigcap MFFC_i = \phi$ or $MFFC_i \subseteq X$.

Proof Omitted due to the length restriction.

This theorem tells us intuitively that there exists an optimal acyclic two-way partition that does not cut through any MFFC in the MFFC decomposition of N. This result further suggests that MFFC decomposition leads to a natural clustering solution.

The size of each MFFC is usually not too big due to the existence of large number of multiple fanout nodes in a network(this is especially true for networks optimized using automatic synthesis tools). A typical MFFC usually contains three to ten gates. Occasionally, we may encounter an MFFC of very large size (for example, when a network has a single PO, the entire network is an MFFC). In order to avoid very large clusters, we set up a threshold of the maximum cluster size allowed in a circuit, defined by $\beta \cdot \frac{|A|}{K}$, where $|A|$ is the total area of all gates in N and β is a control parameter between 0 and 1. When an MFFC area exceeds this threshold, we further decompose this MFFC H into an FFC H' and a set of smaller MFFCs (obtained by the MFFC decomposition of $H - H'$), so that none of the nodes in the clustered network violates the area constraint. (Details are omitted due to the length restriction.) In our experiments, parameter β is chosen between 1/3 and 1/2.

In summary, our MFFC-based acyclic K-way partitioning algorithm first computes an MFFC decomposition of the given network. Next, each MFFC is treated as a cluster and is collapsed into signal node. (If an MFFC is too large, further decomposition is necessary.) Then, we apply the K-AFM algorithm described in the preceding section to compute an acyclic K-way partition of the clustered network. We denote this algorithm as the K-MAFM algorithm.

5. Experimental Results

We have implemented the K-AFM algorithm and the K-MAFM algorithm using C language on Sun SPARC workstations. For comparison purpose, we have also implemented the conventional K-FM algorithm which does not consider signal directions and the acyclic constraint. The benchmark examples are from the ISCAS85 suite, which is a set of large combinational circuits originally used to evaluate test generation algorithms. We choose this benchmark suite because it consists of combinational circuits with the signal direction information. Other netlist examples usually do not provide signal direction information. Table 1 shows the characteristics of the

benchmark circuits in terms of the number of gates, number of PIs, number of nets, and number of edges.

We have compared the three multi-way partitioning algorithms: undirected K-FM, acyclic K-AFM, and the MFFC-based acyclic K-MAFM algorithm. Cluster threshold parameter β were set to be 1/2 and 1/3 in two versions of K-MAFM algorithm. The first column under the K-MAFM results shows the number of nodes in the clustered network. As we can see, a factor of 2 to 10 reduction in terms of network size reduction is achieved after MFFC-based clustering. In our experiments, the numbers of partitions were chosen to be 4 and 8. Each algorithm was invoked 10 times and the best partitioning result was reported in the table. Each partition allows ±5% deviation from its target area. As we expected, due to the acyclic constraint, on the average, the K-AFM algorithm produced solutions with 26-80% larger net-cut sizes as compared to those by the K-FM algorithm. The acyclic constraint forces the K-AFM algorithm to choose from a smaller set of initial solutions (restricted to topological orderings) and have fewer feasible moves to choose from at each iteration. However, it is almost surprising to see that on the average the K-MAFM algorithm produced solutions with 30-50% smaller net-cut sizes as compared to those by the K-FM algorithm, despite of the acyclic constraint imposed on the K-MAFM algorithm. This suggests strongly that MFFC-based clustering method have produced very high clustering solutions. In fact, we observed from our experiment that even if we just apply simple topological sorting based partitioning algorithm on the clustered network obtained from the MFFC clustering (which is essentially the initial solution before we apply the K-AFM algorithm to the clustered network) , we can achieve very high quality partitioning solutions. Since both the MFFC decomposition and topological sorting can be done in linear time, this method suggests a very fast and high-quality partitioning algorithm for partitioning Boolean networks, which can be used as a fast estimation/prediction tool by the higher level synthesis tools. Our experiment results show that runtime of K-MAFM is comparable to that of K-FM.

Ciruit	No. of gates	No. of PIs	No. of nets	No. of edges
C880	383	60	443	695
C1355	546	41	587	1055
C1908	880	33	913	1490
C2670	1193	233	1426	1983
C3540	1667	50	2167	2911
C5315	2307	178	2485	4331
C6288	2418	32	2450	4800

Table 1. Benchmark circuits.

6. Conclusions and Future Research

In this paper, we have presented a general formulation of the acyclic multi-way partitioning problem for combinational Boolean networks and identified a number of

Circuit	K-FM	K-AFM	K-MAFM			
			β = 1/2		β = 1/3	
K=4	net-cut	net-cut	cluster	net-cut	cluster	net-cut
c880	69	94	77	31	79	42
c1355	61	82	58	17	58	17
c1908	121	198	160	80	182	99
c2670	220	256	239	123	332	161
c3540	271	343	353	178	353	178
c5315	359	587	385	194	388	186
c6288	721	316	1456	311	1456	311
	1	+26.4%		-50.6%		-42.1%

Circuit	K-FM	K-AFM	K-MAFM			
			β = 1/2		β = 1/3	
K=8	net-cut	net-cut	cluster	net-cut	cluster	net-cut
C880	81	156	91	52	116	60
C1355	75	184	58	23	90	43
C1908	161	327	187	112	189	111
C2670	252	443	379	246	355	208
C3540	337	575	353	232	356	229
C5315	386	866	405	238	461	248
C6288	1018	491	1456	487	1456	487
	1	+80.0%		-33.1%		-30.1%

Table 2. Partitioning results using different partitioning algorithms.

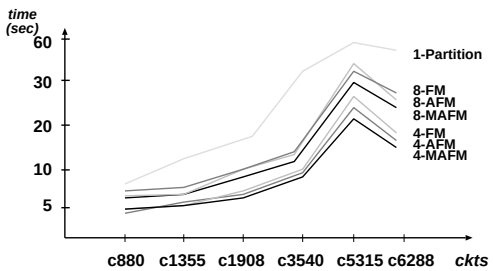

Figure 4. Simulation results.

important applications. Our study has led to a very efficient acyclic multi-way partitioning algorithm named K-MAFM algorithm based on the theory of MFFC decomposition. The K-MAFM algorithm not only guarantees an acyclic partitioning solution but also consistently outperforms the undirected K-FM algorithm. The success of the K-MAFM algorithm suggests that the MFFC decomposition is a very powerful technique for clustering and partitioning of large combinational Boolean circuits. We are confident that this technique will find other applications in VLSI designs. Our study also suggests that considering signal directions during clustering and partitioning process is very useful. We hope that our study on the acyclic multi-way partitioning problem can stimulate further investigation on this topic in the future.

As one application, we have evaluated the impact of different acyclic multi-way partitioning algorithms on circuit simulation. Our simulator is written in Maisie, a simulation language developed at UCLA, which allows a simulation model to be executed in either sequential, parallel conservative, or parallel optimistic modes with minimal changes to the model [BaLi94]. Maisie also supports a number of optimizations to reduce both checkpointing and blocking overheads for the corresponding simulation protocol. The simulations reported in this paper used the sequential algorithm to simulate circuits with one, four, and eight partitions on a single processor workstation. The simulation time is shown in Figure 4. The K-MAFM partitioning solutions lead to most efficient simulation for a fixed K because K-MAFM partitioning solutions require least amount of communication. It is interesting to see that the simulation time is *not* monotone with respect to K, since for smaller K, the event queue is longer in each partition, while for larger K, inter-partition communication is higher. Currently, we are investigating the problem of selecting optimal partition number for best speed-up. We are also developing parallel circuit simulation algorithm under both conservative and optimistic schemes on several parallel architectures.

References

[Ba82] Barnes, E. R., ''An Algorithm for Partitioning the Nodes of a Graph,'' *SIAM J. Alg. Disc. Math.*, Vol. **3**, pp. 541-550, 1982.

[BaLi94] Bagrodia, R. and W.-t. Liao, ''A Language for design of Efficient Discret-Event Simulations,'' *IEEE Transactions on Software Engineering*, March, 1994.

[BeSa93] Beardslee, M. and A. Sangiovanni-Vincentelli, ''Heuristic Methods for Communication-Based Logic Partitioning,'' *4th ACM/SIGDA Physical Design Workshop*, pp. 199-210, April 1993.

[Bo87] Boppana, R., ''Eigenvalues and Graph Bisection: An Average-Case Analysis,'' *IEEE Symp. on Foundations of Computer Science*, pp. 280-285, 1987.

[ChSZ93] Chan, P., M. Schlag, and J. Zien, ''Spectral K-Way Ratio-Cut Partitioning and Clustering,'' *Proc. 30th ACM/IEEE Design Automation Conf.*, June 1993.

[CoDi93] Cong, J. and Y. Ding, ''On Area/Depth Trade-off in LUT-Based FPGA Technology Mapping,'' *Proc. 30th ACM/IEEE Design Automation Conf.*, pp. 213-218, June 1993.

[CoHK91] Cong, J., L. Hagen, and A. Kahng, ''Random Walks for Circuit Clustering,'' *IEEE 4th Int'l ASIC Conf.*, pp. P14-2.1, Sept. 1991.

[CoHK92] Cong, J., L. Hagen, and A. Kahng, ''Net Partitions Yield Better Module Partitions,'' *IEEE 29th Design Automation Conference*, pp. 47-52, June 1992.

[CoSm93] Cong, J. and M. Smith, ''A Bottom-up Clustering Algorithm with Applications to Circuit Partitioning in VLSI Designs,'' *ACM/IEEE Design Automation Conf.*, pp. 755-760, June 1993.

[FiMa82] Fiduccia, C. and R. Mattheyses, ''A Linear Time Heuristic for Improving Network Partitions,'' *ACM/IEEE Design Automation Conf.*, pp. 175-181, 1982.

[HaKa91] Hagen, L. and A. B. Kahng, ''Fast spectral methods for ratio cut partitioning and clustering,'' *Proc. ICCAD-91*, pp. 10--13, 1991.

[HaKa92] Hagen, L. and A. Kahng, ''A New Approach to Effective Circuit Clustering,'' *Int'l Conf. on Computer-Aided Design* , pp. 422-427, Nov. 1992.

[HaKa92b] Hagen, L. and A. B. Kahng, ''New Spectral Methods for Ratio Cut Partitioning and Clustering,'' *IEEE Trans. on CAD*, pp. 1074-1085, Sept. 1992.

[HwGa92] Hwang, J. and A. E. Gamal, ''Optimal Replication for Min-Cut Partitioning,'' *Int'l Conf. on Computer-Aided Design* , pp. 432-435, Nov. 1992.

[ImPF93] Iman, S., M. Pedram, C. Fabian, and J. Cong, ''Finding Uni-Directional Cuts Based on Physical Partitioning and Logic Restructuring,'' *4th ACM/SIGDA Physical Design Workshop*, pp. 187-198, April 1993.

[KeLi70] Kernighan, B. and S. Lin, ''An Efficient Heuristic Procedure for Partitioning of Electrical Circuits,'' *Bell System Technical J.*, Feb. 1970.

[KiGV83] Kirkpatrick, S., C. D. Gelat, and M. P. Vecchi, Jr., ''Optimization by Simulated Annealing,'' *Science*, Vol. **220**, pp. 671-680, May, 1983.

[Kr84] Krishnamurthy, B., ''An Improved Min-Cut Algorithm for Partitioning VLSI Networks,'' *IEEE Trans. on Computers*, Vol. **33**, pp. 438-446, 1984.

[KrNe91] Kring, C. and A. R. Newton, ''A Cell-Replicating Approach to Mincu-Based Circuit Partitioning,'' *IEEE Int'l Conf. on Computer-Aided Design*, pp. 2-5, Nov. 1991.

[Sa89] Sanchis, L., ''Multiple-Way Network Partitioning,'' *IEEE Trans. on Computers*, Vol. **38**, pp. 62-81, 1989.

[WeCh89] Wei, Y. and C. Cheng, ''Towards Efficient Hierarchical Designs by Ratio Cut Partitioning,'' *IEEE Intl. Conf. on Computer-Aided Design*, pp. 298-301, Nov. 1989.

[YeCL92] Yeh, C. W., C. K. Cheng, and T. T. Lin, ''A Probabilistic Multicommodity-Flow Solution to Circuit Clustering Problems,'' *Int'l Conf. on Computer-Aided Design*, pp. 428-431, Nov. 1992.