Multi-Way VLSI Circuit Partitioning Based on Dual Net Representation

Jason Cong, Wilburt Labio, and Narayanan Shivakumar
UCLA Computer Science Department
University of California, Los Angeles, CA 90024

Abstract

In this paper, we study the area-balanced multi-way partitioning problem of VLSI circuits based on a new dual netlist representation named the hybrid dual netlist (HDN). Given a netlist, we first compute a K-way partition of the nets based on the HDN representation, and then transform a K-way net partition into a K-way module partitioning solution. The main contribution of our work is the formulation and solution of the K-way module contention (K-MC) problem, which determines the best assignment of the modules in contention to partitions, while maintaining user-specified area requirements, when we transform the net partition into a module partition. Under a natural definition of binding factor between nets and modules, and preference function between partitions and modules, we show that the K-MC problem can be reduced to a min-cost max-flow problem. We present efficient solutions to the K-MC problem based on network flow computation. Extensive experimental results show that our algorithm consistently outperforms the conventional K-FM partitioning algorithm by a significant margin.

1. Introduction

The K-way partitioning problem is one of partitioning the modules in a network into K subsets (partitions) of "approximately" the same size while minimizing the number of interconnections between the K partitions. This problem has many applications in VLSI circuit design ranging from circuit layout to logic simulation and emulation.

The existing partitioning algorithms in the literature can be grouped into two-way partitioning (bipartitioning) algorithms and multi-way partitioning algorithms. The bipartitioning algorithms include the iterative improvement methods [KeLi70, FiMa82, Kr84, KiGV83], the graph spectral method [Bo87, HaKa91], and the net-based partitioning method [HaKa92b, CoHK92]. The multi-way partitioning algorithms include the recursive bi-partitioning by Kernighan and Lin [KeLi70], a generalization of the FM-algorithm with lookahead by Sanchis [Sa89], the primal-dual algorithm [YeCL91], and a generalization of the graph spectral-based partitioning method to multi-way ratio-cut by Chan, Schlag, and Zien [ChSZ93]. To reduce the computational complexity for partitioning very large circuits, cluster-based partitioning methods have been introduced based on various clustering techniques, such as random-walk clustering [CoHK91, HaKa92], multicommodity-flow based clustering [YeCL92], clique based clustering [CoSm93], geometric embedding with min-diameter clustering [AlKa93], and clustering based on maximum fanout-free cones (MFFCs) [CoLB94].

Since the objective of the partitioning problem is to minimize the number of nets to be cut, we believe that assigning nets, instead of modules, to partitions will lead to better partitioning solutions in general. The net-based bipartitioning algorithm by Cong, Hagen, and Kahng [CoHK92] is therefore of particular interest to us. This algorithm first computes a bipartitioning of the nets using the graph spectral method, and then transforms the net bipartitioning solution into a module bipartitioning solution by solving the *module contention problem*. It was shown that the module contention problem for bipartitioning can be solved optimally by computing a minimum vertex covering in a bipartite graph, and very encouraging experimental results were reported. However, the minimum vertex covering formulation for the module contention problem is inherent to bipartitioning and cannot be easily generalized to multi-way partitioning.

In this paper, we present a K-way net-based partitioning algorithm with consideraton of the area balance constraint. We introduce a new dual netlist representation named *hybrid dual netlist (HDN)*. Given a netlist, we first compute a K-way partition of the nets based on the *HDN* representation, and then transform the K-way net partition into a K-way module partition. The main contribution of our work is the formulation and solution of the *K-way module contention (K-MC)* problem.

The rest of the paper is organized as follows. We present the problem formulation and terminologies in Section 2. Section 3 presents the hybrid dual netlist representation and our K-way partitioning algorithms. Section 4 presents experimental results. We conclude the paper in Section 5 with some observations and directions for future work.

2. Problem formulation

Given a netlist *NL* to be partitioned into *K* partitions, we use $M = \{ m_1, m_2, ..., m_p \}$ to denote the set of modules in *NL*, $N = \{ n_1, n_2, ..., n_q \}$ to denote the set of nets in *NL*, and $P_1, P_2, ..., P_K$ to denote the K partitions, where p is the number of modules, and q is the number of nets in *NL*. The modules may have different areas.

An *optimal area-balanced K-way partitioning solution* of a given netlist *NL* satisfies the following conditions:

(i) Each module is assigned to exactly one partition.

(ii) The total area of the modules in each partition are within the user-specified area bounds, i.e.

$$(1 - \alpha) \cdot \frac{A}{K} \leq A_i \leq (1 + \alpha) \cdot \frac{A}{K}$$

for each partition P_i, where A is the total area of all the modules in *NL*, A_i is the total area of all the modules in partition P_i, and α is a user-specified parameter controlling the allowable slack in the area constraint.

(iii) The number of nets being cut is minimized.

Given a netlist *NL* (for example, shown in Fig. 1(a)), we introduce the following definitions:

(i) **Netlist Hypergraph:** $NH = (V(NH), H(NH))$, where each vertex in $V(NH)$ represents a module m_i $(1 \leq i \leq p)$ and each hyperedge in $H(NH)$ represents a net n_j $(1 \leq j \leq q)$ (see Fig. 1 (b)).

(ii) **Net Intersection Graph (NIG):** $NIG = (V(NIG), E(NIG))$, where each node in $V(NIG)$ represents a net n_i $(1 \leq i \leq q)$, and there is an edge in $E(NIG)$ between n_i and n_j iff $n_i \cap n_j \neq \phi$ (i.e the two nets share common modules). Note that *NIG* is a graph instead of a hypergraph (see Fig. 1 (c)).

(iii) **Dual Netlist Hypergraph (DNHG):** $DNHG = (V(DNHG), H(DNHG))$ where each node in $V(DNHG)$ represents a net and each hyperedge in $H(DNHG)$ represents $N(m_i)$, the set of nets incident to module m_i $(1 \leq i \leq p)$ (see Fig. 1 (d)).

3. The K-DualPART algorithm

3.1. Overview of K-DualPART

Our dual netlist based K-way partitioning algorithm, K-DualPART, consists of the following phases:

(1) We first *convert the netlist hypergraph to a dual net representation* named *hybrid dual netlist(HDN)*, which is a combination of *NIG* and *DNHG*.

(2) *Assign nets to partitions*. We use the K-FM partitioning algorithm [FiMa82, Sa89] to partition the nets into K partitions.

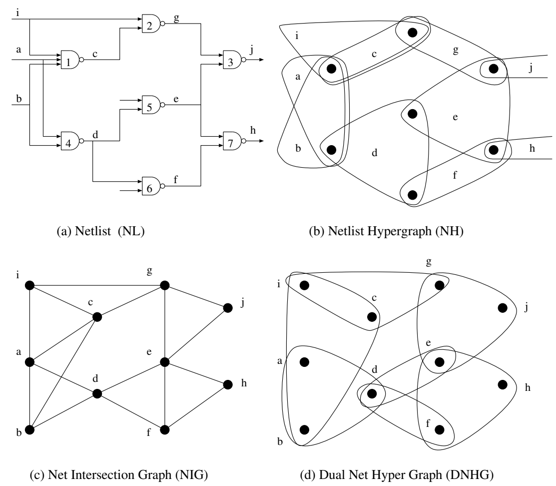


Figure 1 Different Circuit Representations

(3) We transform the net partitioning solution into a module partitioning solution by solving the *K-way module contention problem* (K-MC) based on the min-cost max-flow formulation.

(4) We further *improve the module partitioning solution* again using K-FM partitioning algorithm.

The subsequent subsections describe these phases in detail.

3.2. Generating dual netlist representations

The net intersection graph (*NIG*) was used in [CoHK92] since the graph spectral based algorithm used in their method for net partitioning applies only to graphs and cannot be used for hypergraphs. However, we notice that for many examples, there are a large number of nets incident on the same module (see Fig. 2(a)), and these nets will form a large clique (complete graph) in the *NIG* (see Fig. 2(b)). In this case, the memory requirement for storing *NIG* is high and partitioning *NIG* also tends to be more difficult and time consuming. Moreover, since *NIG* can be very dense when large nets exist in the circuit, iterative improvement based partitioning algorithms may easily be trapped in local optima. The dual netlist hypergraph (*DNHG*) (Fig. 2(c)) defined in Section 2 is more economical in terms of memory requirement when compared to the *NIG*. However, our study shows that use of *DNHG* directly as the dual netlist representation does not give the best partitioning results either since *DNHG* representation does not distinguish the sizes of the nets. To avoid these problems, we introduce a threshold parameter *CF* when constructing the net intersection graph. When the number of nets incident to the same module is more than *CF*, we connect these nets by a hyperedge instead of a large clique. The resulting dual netlist representation is called the *hybrid dual netlist* (*HDN*) representation. Note that if we set *CF* to be 2, then the *HDN* is the same as the *DNHG*. In general, *HDN* (shown in Fig. 2(d)) is a combination of *NIG* and *DNHG*. Our experimental results confirm that net partitioning based on *HDN* produces better results than those based on *NIG* or *DNHG* representations. In our implementation, *CF* was chosen to be 5. Note that *HDN* is a hypergraph in general. Since we use the K-FM algorithm for net partitioning (see next sub-section), a hypergraph representation presents no problem to us.

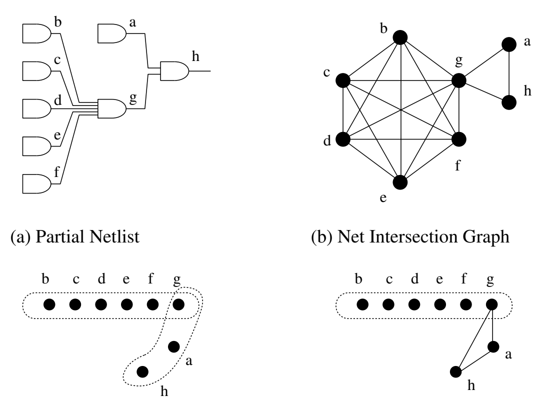
3.3. Partition of dual netlist representation

After constructing the *HDN* hypergraph, we use the K-way Fiduccia-Mattheyses (K-FM) algorithm [FiMA82, Sa89] to compute a K-way partitioning of *HDN* to obtain a K-way partitioning of the nets in the original netlist. We want to minimize the number of edges cut in *HDN* so that the subsequent module contention is easier to solve. We apply K-FM to a number of random initial net partitions as well as an initial net partition computed using a simple deterministic greedy algorithm.

3.4. Solution to the K-MC problem

3.4.1. Problem statement

We say that a module m is in contention if there exist two nets n_1 and n_2 containing m such that n_1 and n_2 are in two partitions in the net partitioning solution. We use M_{cont} to denote the set of modules in contention.



(a) Partial Netlist (b) Net Intersection Graph

(c) Dual Net Hypergraph (d) Hybrid Dual Netlist (CF=5)

Figure 2 Advantage of Hybrid Dual Netlist

The K-MC (K-way module contention) problem is *to assign modules in* M_{cont} *to proper partitions* so that the total number of nets being cut is minimized. If we start with a good net partitioning (which is usually the case after applying K-FM algorithm on *HDN*), the size of M_{cont} is much smaller than the number of modules in the original netlist. From our experiments, we see that for the MCNC benchmarks, the percentage of modules in contention ranges from 50 - 62% for K = 2, 45 - 60% for K = 3, 40 - 50% for K = 4, and 30 - 42% for K = 5. Therefore, the K-MC problem is much simpler than the original K-way partitioning problem, and judging by the trend of our results, it gets simpler with increasing K.

3.4.2. Binding factor and preference function

Good solutions to the K-MC problem should minimize the number of nets being cut under the area constraint. Since this problem is NP-Hard in general, we resort to efficient heuristic algorithms. We introduce a metric to approximate the number of nets cut when a module is assigned to partition P_i. Intuitively, a net n_j has a high affinity for a module m_k in contention if it has a high probability of being satisfied (uncut) after attracting m_k into its partition, and a low affinity if it is most likely to be cut even after obtaining m_k. We introduce a *binding factor* (*bf*) metric to measure this affinity between a net and a module. Let n_j be a net and $m_k \in n_j$ be a module in contention. The binding factor $bf\,(n_j, m_k)$ should depend on the following factors:

(i) $S(n_j)$*, the number of modules in* n_j*:* As the number of modules in a net increases, the probability of the net being satisfied (uncut) is reduced. So the $bf\,(n_j, m_k)$ should be inversely proportional to the net size $S(n_j)$.

(ii) $C(n_j)$*, the number of modules in contention in net* n_j: If $C(n_j)$ is high, the probability of the net being satisfied is low. Hence, the $bf\,(n_j, m_k)$ should be inversely proportional to $C(n_j)$.

(iii) $S(n_j) - C(n_j)$*, the number of modules of net* n_j *in its partition already*, i.e. the number of modules in n_j not in contention. The $bf\,(n_j, m_k)$ should be directly proportional to this factor since the probability of the net being satisfied increases as this number increases.

Therefore, we consider the ratios $\frac{S(n_j) - C(n_j)}{S(n_j)}$ and $\frac{S(n_j) - C(n_j)}{C(n_j)}$ to be of primary importance in determining the *binding function bf*. From the two ratios, we define the *binding function* of net n_j for module m_k to be

$$bf\,(n_j,\ m_k) = \frac{(\ S(n_j) - C(n_j)\ )^2}{S(n_j) \times C(n_j)}$$

We define that $bf(n_j,\ m_k) = 0$ if m_k is not in n_j. Also, if two modules m_k and m_l in net n_j are already assigned to two different partitions (i.e. n_j is already cut), then $bf\,(n_j,\ m_i) = 0$ for any $m_i \in n_j$.

Based on the definition of the binding factor, we define the *preference function* $pf\,(P_i, m_k)$ between a module m_k in contention and a partition P_i as follows:

$$pf\,(m_k,\ P_i) = \sum_{n \in P_i} bf\,(n, m_k)$$

That is, the preference function between module m_k and partition P_i is the sum of binding functions between m_k and all nets in partition P_i. Our objective is to find an optimal assignment of the modules in M_{cont} to the partitions such that the *cumulative preference* over all assignment edges is maximized.

3.4.3. Flow-based formulation of K-MC problem

We use the min-cost max-flow algorithm to compute the optimal module assignment. First, we construct a *assignment network (AN)* as follows. We construct a bipartite graph in which the nodes represent the modules in M_{cont} and the partitions in P and each directed edge (m_k, P_i) connects module m_k to partition P_i. Then, we add a source node s to AN and connect it to every module node m_k in AN. Similarly, we add a sink node t to AN and connect every partition node P_i to the sink t. Fig. 3 shows an example of the assignment network. For each edge e in the assignment network, we define its capacity $cap\,(e)$ and cost $cost\,(e)$ as follows:

(i) if $e = (s, m_k)$, $cap(e) = 1$, $cost(e) = 0$;

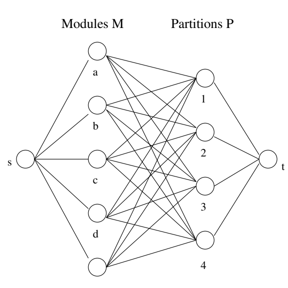


Figure 3 Assignment Network (AN)

(ii) if $e = (P_i,\ t)$, $cap(e) = cap\,(P_i)$, $cost(e) = 0$, where $cap\,(P_i)$ is the number of modules that partition P_i can accept without violating its area constraints.

(iii) if $e = (m_k,\ P_i)$, $cap(e) = 1$, $cost(e) = MAX - pf\,(P_i,\ m_k)$, where MAX is a positive constant larger than any preference function value.

Let $m = |M_{cont}|$. In general, we have

$$\sum_{i=1}^{q} cap\,(P_i) \geq m\ (=\ |M_{cont}|). \qquad \text{(C1)}$$

That is, the total excess capacity in all partitions is larger than the number of modules in contention. (It is easy to show that when all modules are uniform in size, condition C1 is always true. When module sizes vary significantly, we can only use $cap\,(P_i)$ to estimate the maximum number of modules allowed in P_i without violating the area balance constraint, and such estimation usually tends to be conservative. In practice, if we relax the area constraint parameter α as defined in Section 2, we can always satisfy condition C1.) When condition C1 is true, we have the following results (proofs of these results can be found in [CoLS94]).

Lemma The value of the maximum flow in the assignment network is m.

Theorem 1 The min-cost max-flow in the assignment network induces a module assignment whose total preference function is maximum.

In fact, it is easy to see that the max-flow in the assignment network consists of m edge disjoint paths from s to t. Each path includes exactly one edge of type (m_k, P_i), which defines the assignment of module m_k to partition P_i. Moreover, the cost of the max-flow equals $m \cdot MAX$ minus the total preference function of the corresponding module assignment.

We use the augmenting path algorithm [FoFu62] for computing a minimum-cost maximum-flow in the assignment network. We start with a flow of value zero. At each step, we compute the minimum cost augmenting path in the residual graph of the assignment network. Then, we augment the flow value by one, and update the residual graph of the assignment network. The augmentation process stops after m steps. It is easy to show that the time complexity of the minimum-cost maximum-flow computation in our case is $O\,(K \cdot m^2)$. After we obtain a min-cost max-flow, we can determine the assignment of modules in contention in linear time.

When condition C1 is not satisfied (it occurs in rare cases when the area slack parameter α is very small and the module sizes vary significantly), the max-flow in the assignment network has a value less than m, which means that some modules in M_{cont} are left unassigned. In this case, we remove the assigned modules from M_{cont} and the assignment network, update the excess capacity estimation $cap(P_i)$ for each partition P_i based on the knowledge of newly assigned modules, and update the binding factors and preference functions associated with the unassigned modules in M_{cont}. Usually, the partition capacity estimation is much more accurate after considering the newly assigned modules. Therefore, condition C1 is very likely to be satisfied for the remaining unassigned modules. Then, we can perform another min-cost max-flow computation on the assignment network to compute the assignment of modules in M_{cont}. In theory, we may need to go through a number of flow computation steps to assign all the modules in contentions. In practice, however, for most real circuits and reasonable choice of the area slack parameter (such as 10%), we need to go through flow computation only once.

3.4.4. Dynamic updating of binding factors

One problem with the solution presented in the preceding sub-section is that the static *preference functions* are not reflective of the changes of the binding factors of the nets as more and more modules are assigned during flow computation. In Fig. 4, when m_1 is assigned to P_2 based on n_3's strong affinity, it is clear that $bf(n_4, m_2)$ should increase since the probability of its net being satisfied is increased, and hence the $pf(P_2, m_2)$ increases. Also, $bf(n_1, m_2)$ should be assigned zero since n_1 is being cut, and $pf(P_1, m_2)$ should be decreased accordingly.

We have developed efficient procedures to update the binding factors and preference functions after each module assignment or re-assignment during the flow computation. Details of these procedures can be found in [CoLS94]. It is interesting to note that this dynamic updating of binding factors and preference functions can be easily incorporated in our min-cost max-flow algorithm after each flow augmentation. Observe that each flow augmenting path assigns one more module to a partition

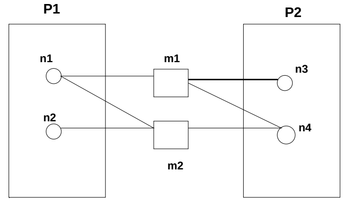


Figure 4 Dynamic Nature of Binding Functions

and also possibly specifies re-assignment of some other modules. Therefore, even with dynamic updating of edge costs in the assignment network, we can still show that flow augmentation stops after m steps. So, we have the following results:

Theorem 2 With dynamic update of binding factors and preference functions, the K-MC problem can be solved in $O(K \cdot m^2)$ time based on min-cost max-flow computation in the assignment network, where K is the number of partitions and m is the number modules in contention.

The K-DualPART algorithm with dynamic updating of binding factors in the flow computation is denoted as K-DualPART/DF, and the one with static binding factors is denoted as K-DualPART/SF. From the results shown in the next section, we shall see that K-DualPART/DF produces better partitioning solutions in general as compared K-DualPART/SF. The increase in computation time due to dynamic updating of edge costs is negligible due to the incremental updating.

3.5. Refinement of module partitioning solution

After solving the K-MC problem, we obtain a K-FM module partitioning solution. We apply another pass of K-FM partitioning algorithm to further refine the module partitioning solution. However, we observe that in all test cases the K-FM based refinement step converges very quickly with very few module moves, which is a strong indication that the K-FM module partitioning solution obtained from K-way net partitioning and module contention resolution is of very high quality.

4. Experimental results

We have implemented both the K-DualPART/SF and K-DualPART/DF algorithms on SUN SPARC workstations and Hewlett-Packard 735 workstations. We compared the two algorithms with the conventional K-FM algorithms on a set of MCNC benchmark circuits (Test02-06, PrimGA1, PrimGA2) and 5 large circuits provided by the Hewlett-Packard Research Lab (CPU, GA_machine, FPU, GA_machine2, FPU2). Circuits CPU, GA_machine and FPU consist of lookup-tables (for multi-FPGA implementation). Circuits FPU2 and GA_machine2 are the original netlist of FPU and GA_machine before technology mapping.

Table 1 shows the number of modules and the number of nets of the benchmark circuits.

Tables 2(a)-2(b) show the comparison of K-DualPART/SF and K-DualPART/DF with the K-FM algorithm for K ranging from 3 to 4. (More comparative

Ckt	# modules *	# nets
Test02	1724	1721
Test03	1664	1618
Test04	1541	1658
Test05	2650	2751
Test06	1813	1674
PrimGA1	914	902
PrimGA2	3121	3029
cpu	947	1729
GA_machine	6198	11049
FPU	8046	15901
GA_machine2	25452	25628
FPU2	30669	33826

Table 1. Characteristics of the Test Circuits.

results can be found in [CoLS94].) For each example, the K-FM algorithm was run 20 times, each on a random initial module partitioning. In order to obtain a fair comparison, we make sure that the runtime K-DualPART/SF and K-DualPART/DF are comparable with that of the K-FM algorithm. As a result, the K-DualPART algorithms were run once with the greedy net partition, and then approximately 10 times[1], each on a random initial net partitioning of the dual netlist representation (HDN). The area slack parameter α was set to be 10% in both K-FM and K-DualPART algorithms. While this area slack parameter can be satisfied in most cases, there were a few cases where the area of the largest module is almost the same or even larger than the allowed partition area (e.g. this happens to Test02 when K = 4 or 5). In those cases, the area slack parameter was relaxed to 25% - 45% for both K-FM and K-DualPART algorithms.

One can see from Tables 2(a) - 2(b) that the K-DualPART/DF algorithm consistently outperforms the K-FM algorithm by a significant margin, 23% to 30% reduction for K = 3, and 4. The K-DualPART/SF algorithm produces considerably better results with about 19% to 24% cutsize reduction as compared to K-FM for K = 3 and 4. In general, the K-DualPART/DF algorithm outperforms the K-DualPART/SF algorithm, but the difference decreases as the number of partition K increases. In terms of efficiency, the K-DualPART/SF algorithm is generally faster than the K-DualPART/DF algorithm, and the difference increases as the number of partitions increases. The K-DualPART/SF obtained a solution for FPU2 in 19080 seconds and 32400 seconds for K = 3 and K = 4, respectively. The K-DualPART/DF took 19260 seconds and 49260 seconds for the same example.

*****The number of modules include the I/O pads.

1The number of runs of K-DualPART varies from example to example in order to match the runtime of 20-run K-FM algorithm on the same example. The unmapped HP circuits were run only with the greedy net-partition due to time considerations, while the mapped HP and the MCNC ranged from 8 - 13 runs.

Ckt	K-FM	SF	DF
Test02	114	82	81
Test03	271	231	201
Test04	588	337	329
Test05	1069	559	517
Test06	281	297	252
PrimGA1	197	155	159
PrimGA2	704	628	578
cpu	551	352	389
GA	2270	2321	2253
FPU	1238	1759	1815
GA2	5772	4181	2829
FPU2	5673	2860	3776
overall		-19.06%	-22.6%

Table 2 (a) Comparison of K-DualPART against K-FM for K = 3

Ckt	K-FM	SF	DF
Test02	724	455	318
Test03	368	324	298
Test04	843	459	514
Test05	1233	691	863
Test06	332	304	285
PrimGA1	203	205	187
PrimGA2	875	726	717
cpu	716	644	429
GA	3778	3021	3075
FPU	3178	2290	2365
GA2	10112	4915	2968
FPU2	7300	5684	5552
overall		-24.44%	-30.26%

Table 2 (b) Comparison of K-DualPART against K-FM for K = 4

5. Conclusion and possible future extensions

The results in this paper show convincingly that net partitioning based methods produce better solutions to the multi-way circuit partitioning problem than direct module partitioning. Our formulation and solution to the K-way module contention problem provide a general and effective method to convert a K-way net partitioning solution to a K-way module partitioning solution. Both the K-DualPART/SF and K-DualPART/DF algorithms can be extended easily to handle many practical constraints, such as I/O bound constraint on each partition, pre-specified assignment of modules to partitions, etc.

Our partitioning results for the test circuits, ranging from 1000 to 31000 modules, prove that our K-DualPART algorithm is scalable in terms of the size of the circuit. Other partitioning algorithms (such as the graph spectral based method) may fail to produce solutions for large circuits due to high memory usage (e.g. to store the Laplacian matrix) or speed inefficiency. The memory and speed efficiency of the K-DualPART algorithm enables us to handle problems of much larger sizes.

When the problem size is not too large, more elaborate K-way partitioning algorithms other than the simple K-FM algorithm can also be used to produce a better net partitioning solution on the dual netlist representation. The result of our K-MC resolution algorithm can also be improved by dynamically updating the capacity of each partition after an augmenting path is computed. This enhancement would be particularly useful when the modules sizes vary significantly.

6. Acknowledgments

This work is partially supported by ARPA/CSTO under contract J-FBI-93-112, the National Science Foundation Young Investigator Award award number MIP9357582, and grants from AT&T Bell Laboratories, Hewlett-Packard, and Xilinx under the California MICRO program. The authors would like to thank Eugene Ding and Yeanyow Hwang for their help in our work.

References

[AlKa93] Alpert, C. J. and A. B. Kahng, ''Geometric Embeddings for Faster (and Better) Multi-Way Netlist Partitioning,'' *Proc. ACM/IEEE Design Automation Conf.*, pp. 743-748, June 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.

[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.

[CoLB94] Cong, J., Z. Li, and R. Bagrodia, ''Acyclic Multi-Way Partitioning of Boolean Networks,'' *Proc. ACM/IEEE 31st Design Automation Conf.*, pp. 670-675, June 1994.

[CoLS94] Cong, J., W. Labio, and N. Shivakumar, ''Multi-Way VLSI Circuit Partitioning Based on Dual Net Representation,'' in *UCLA Computer Science Department Tech. Report CSD-940029*, (Aug. 1994).

[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.

[FoFu62] Ford, L. R. and D. R. Fulkerson, *Flows in Networks*, Princeton Univ. Press, Princeton, N.J. (1962).

[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.

[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.

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

[YeCL91] Yeh, C. W., C. K. Cheng, and T. T. Lin, ''A General Purpose Multiple-Way Partitioning Algorithm,'' *Proc. 28th ACM/IEEE Design Automation Conf.*, June 1991.

[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.