A New Approach to Effective Circuit Clustering*

Lars Hagen and Andrew B. Kahng

UCLA CS Dept., Los Angeles CA 90024-1596

Abstract

The complexity of next-generation VLSI systems will exceed the capabilities of top-down layout synthesis algorithms, particularly in netlist partitioning and module placement. Bottom-up clustering is needed to "condense" the netlist so that the problem size becomes tractable to existing optimization methods. In this paper, we establish the DS quality *measure, the first general metric for evaluation of clustering algorithms. The DS metric in turn motivates our RW-ST algorithm, a new* self-tuning *clustering method based on random walks in the circuit netlist. RW-ST efficiently captures a globally good circuit clustering. When incorporated within a two-phase iterative Fiduccia-Mattheyses partitioning strategy, the RW-ST clustering method improves bisection width by an average of 17% over previous matching-based methods.*

1 Introduction

Top-down approaches are widely used to cope with increasing problem complexity in layout synthesis. Recursive calls to a partitioning algorithm generate a circuit hierarchy which subsequently guides the placement/routing phases of layout. Typical partitioning objectives such as minimum-width bisection and minimum ratio cut are NP-complete and require such heuristics as simulated annealing [12], greedy k-opt interchange [11], or quadratic optimization (via relaxation [3] [13] or spectral [8] methods). However, the partitioning algorithms used in top-down layout are beginning to fail as designs approach millions of gates: (i) the space/time requirements of current partitioning approaches become infeasible; (ii) the stability and solution quality of iterative methods deteriorate; and (iii) k-way partitioning formulations may force "unnatural" solutions because of their a priori specification of k.

Given these difficulties, bottom-up clustering can enable *successful* top-down partitioning by condensing the circuit netlist and reducing problem size. Clustering is attractive because it avoids making the far-reaching decisions that are inherent in a top-down approach. Moreover, top-down partitioning solutions can be *enhanced* by first condensing the input through bottom-up clustering [1] [2]. Nevertheless, in practice clustering is avoided because of inherent weaknesses in current bottom-up algorithms, namely, that grouping decisions are based only on local criteria such as the number of connections to modules in an existing cluster. While this locality is needed to maintain reasonable algorithm complexity, it may lead to unfortunate grouping decisions. Thus, top-down partitioning, *while it remains tractable*, remains the preferred method of decomposing a given layout problem. The goal of clustering is then to reduce problem size while deferring far-reaching decisions until *well-considered* top-down optimizations become feasible.

Previous work in circuit clustering ranges from highly *local* to highly *global* approaches. Generally speaking, local approaches are more efficient but can result in unnatural groupings of modules. On the other hand, global approaches give potentially more useful and "natural" results, but may require prohibitive amounts of computation. For our discussion, two particularly relevant approaches are respectively due to Bui et al. [1] [2] and to Garbers et al. [7].

In [1] [2], Bui et al. proposed a two-phase matching based compaction strategy. With this approach, the modules pairs of a maximal random matching in the netlist graph are used to induce a compacted partitioning instance on $n/2$ vertices which correspond to the matching edges. A heuristic Kernighan-Lin partitioning of this compacted netlist is found and then reexpanded into an initial "flat" starting configuration for a second Kernighan-Lin phase. The approach may be iterated, with matching performed recursively on the compacted netlist until the problem size becomes manageable [2].[1]

The approach of [1] [2] in effect performs clustering by finding cliques of size 2, i.e., the matching edges. We may generalize compaction into a more global approach by finding c-cliques for $c > 2$. Even more generally, we could find netlist subgraphs that have size c and a prescribed *density* (e.g., if more than $\epsilon \cdot C(c, 2)$ edges are present among c modules in the netlist, then the c modules would be considered to form a cluster). While density-based clustering is cited in [7] as a folk-

*This work was supported in part by NSF MIP-9110696, ARO DAAK-70-92-K-0001, and ARO DAAL-03-92-G-0050. A. B. Kahng is also supported by an NSF Young Investigator Award and California MICRO grants from Zycad Corporation and Cadence Design Systems.

[1] The heuristic justification for this approach [1] [2] is that the Kernighan-Lin k-opt method yields significantly better results when the graph topology is sufficiently dense, i.e., has large average degree. Bui et al. claim that compacting until average degree in the netlist is ≥ 3 suffices for K-L to become essentially optimal. The authors of [1] conjecture that this is because there are fewer local minima in the k-interchange neighborhood structure when the graph has higher average degree.

lore method, it entails checking all module subsets of cardinality c, which is impractical. Hence, the closely related concept of (k, l)-connectivity was recently proposed by Garbers et al. [7] for use in circuit clustering.

If there are k edge-disjoint paths of length l between modules u and v, then u and v are said to be (k, l)-connected; [7] showed that for certain highly structured classes of random inputs, the transitive closure of the (k, l)-connectedness relation gives an equivalent clustering to that induced by the edge density criterion. However, (k, l)-connectivity may yield nonintuitive results: modules v_i and v_j can belong to a cluster even when no module on any path between v_i and v_j belongs to the cluster. Moreover, the values of k and l which lead to the "correct" clustering must be determined experimentally for each netlist; this determination is not easy, as seen in [7].

Three other methods should be noted. The epitaxial growth or "direct" method [6] iteratively adds the most closely connected unclustered module to the current cluster. This method is highly local, and depends on heuristic choices of cluster seeds, the number of clusters, the tie-breaking rules, etc. The global "top-down clustering" method of [15] is essentially equivalent to top-down recursive application of the ratio cut partitioning approach given in [14]. We do not consider it to be a bona-fide clustering algorithm because it assumes heuristic partitioning can be performed on the flat netlist, and our premise is that if such is possible, clustering is not needed. Finally, [4] gave a global method which, like the present work, is based on a random walk in the netlist.

2 A Proper Clustering Metric

Our primary goal is to find an efficient clustering algorithm which is effective in the sense that it loses as little global structural information as possible. To this end, we first present the *DS quality* measure, which gives an objective metric for distinguishing good clustering decompositions and clustering algorithms.

The DS metric is motivated by the following question: given a graph $G = (V, E)$, how easy is it to separate two nodes $s,\ t \in V$? Observe that (i) if s and t are hard to separate, then there must be more s-t paths and it is more likely that s and t belong to the same natural cluster; (ii) conversely, if s and t are easy to separate, then there must be fewer s-t paths and s and t probably do not belong to the same natural cluster.

We have found that the weighted average of the cluster *degree / separation* (DS) is a robust quality measure: (i) cluster *degree* is the average number of nets incident to each module of the cluster and having at least two pins in the cluster; and (ii) cluster *separation* is the average length of a shortest path between two nodes in the cluster, with separation ∞ if two nodes in the cluster are disconnected.

We calculate the DS quality of a clustering as the weighted average of the DS quality of each cluster, with a cluster containing a single node having DS quality equal to zero. The DS qualities of several different clusterings for the same eight-node graph are shown in Figure 1.[2] The intuition behind maximizing the DS quality is that we wish to find a decomposition of the graph such that nodes will *on average* have the highest possible degree and the shortest possible separation from the other nodes in their respective clusters.

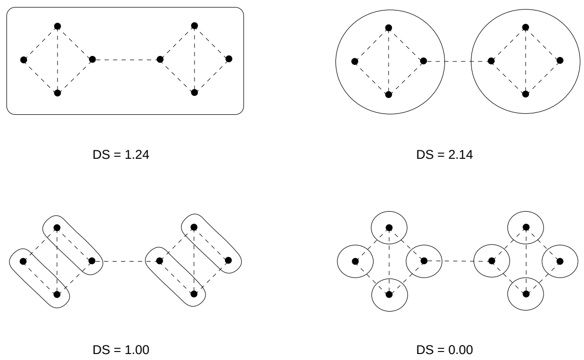

Figure 1: DS quality of various clusterings of the same graph.

The DS quality suggests that the goal of a clustering algorithm should entail finding the neighborhood structure of a node v and comparing it with the neighborhood structure of other nodes to determine which nodes should be clustered with v. This notion of recognizing a node's neighborhood structure motivates our random walk based clustering algorithm.[3]

3 Random Walks for Clustering

We now present our new RW-ST methodology, which computes a circuit clustering based on a random walk in the netlist graph. A *random walk* is a discrete-time stochastic process which iteratively moves from the current module (vertex) to a random adjacent module, with all adjacencies equiprobable. The *cover time* of G is the maximum, over all possible starting vertices, of the expected length of a random walk that visits all vertices in G. The following result shows that a random walk will with high probability manage to explore the netlist structure in a small number of steps.

Fact: The cover time of a random walk in a d-regular graph of n nodes is $O(n^2)$ and $\Omega(n\ log\ n)$, and there and there exist examples which show that both bounds are tight [10]. $\square$

The $O(n^2)$ upper bound also applies to cover times for the class of d-bounded graphs [5], which includes gate-level netlists. Therefore, we may compute a single random walk of length $\Theta(n^2)$ and expect to sample the entire netlist graph.

We propose a method for extracting clusters from the random walk via the following concept of a *cycle*. Consider the sequence of nodes encountered during

[2] For example, each cluster in the 2-clustering has average node degree = 10/4, and average separation = 14/12.

[3] Additional motivations for the DS metric and the random-walk methodology, based on the theory of graph spectra and intrinsic graph structure, are discussed in [9].

the random walk. A *cycle* is a contiguous subsequence $\{v_p, v_{p+1}, \ldots, v_q\}$ in the walk with $v_p = v_q$ and all v_i distinct, $i = p, p+1, \ldots, q-1$. The set of modules in each cycle should correspond to (part of) a natural cluster because if there is a more tightly coupled node subset of the cycle, then the random walk will recur (i.e., complete a smaller cycle) within that subset and we would not have found the original cycle. This is shown intuitively in Figure 2, where the y-y portion of the walk does not delimit a natural cluster since it contains a denser x-x cycle; the x-x cycle does not contain any denser portion, so we say that it is a bona fide cluster.

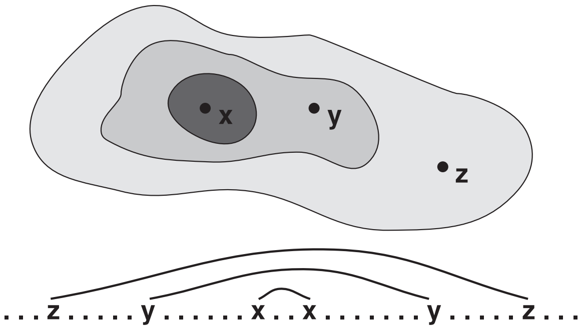

Figure 2: Progress of a random walk through areas with different edge density.

We have designed a *linear-time* algorithm for identifying all of the cycles in a random walk. This algorithm is given in Figure 3.

```
Find-Cycles(RW)
Input: A sequence of nodes RW
for each node i
    visited[i] := FALSE
first := 1
last := 1
visited[last] := TRUE
while last < |RW|
    increment last
    if visited[RW[last]] = TRUE
        while RW[first] ≠ RW[last]
            visited[RW[first]] := FALSE
            increment first
        increment first
    visited[RW[last]] := TRUE
```

Figure 3: Finding cycles in linear time.

In [4], a random walk was computed in the netlist, *maximal* cycles $C(v_j)$ were determined for all modules v_j, and then the transitive closure of the relation $\bowtie$, defined by $v_a \bowtie v_b$ if $v_a \in C(v_b)$ and $v_b \in C(v_a)$, was used to induce a heuristic clustering. However, the experimental results of [4] fail to reflect the intuitively "correct" circuit organization. Our present work offers a different approach, the RW-ST algorithm, which extracts a good heuristic clustering from the cycle information. It should be emphasized that the RW-ST is a heuristic and that we do not yet have strong theoretical justification for its observed success.

The RW-ST algorithm clusters node pairs based on their *sameness*. The sameness of nodes u and v reflects the commonality of the sets of nodes that are visited in cycles originating at u and at v. To calculate the sameness, for each node v we must keep track of how often a node u occurs in some cycle originating at v. This number is saved in the array CC (CycleCount).

```
Sameness(u, v)
Input: A pair of nodes u and v
Output: The sameness value S of u, v
if (CC[u][v] = 0) or (CC[v][u] = 0)
    S := 0
else
    S := 2 · (CC[u][v] + CC[v][u])
    for each node w in the circuit
        if (w ≠ u) and (w ≠ v)
            if CC[u][w] > CC[v][w]
                S := S + 4 · CC[v][w] - CC[u][w]
            else
                S := S + 4 · CC[u][w] - CC[v][w]
```

Figure 4: Computing sameness of two nodes.

Using the CC array, the sameness value for nodes u and v is calculated as shown in Figure 4. If both $CC[u][v]$ and $CC[v][u]$ are greater than zero, i.e., each node occurs at least once in the other's cycles, sameness is initialized to $2 \cdot (CC[u][v] + CC[v][u])$. For each node w, the u-v sameness is increased if the values $CC[u][w]$ and $CC[u][v]$ are approximately equal; the sameness is decreased if these quantities vary by a significant amount. To be specific, for each node w in the circuit other than nodes u and v, we add $4 \cdot min - max$ to the sameness value, where min and max are respectively the smaller and larger of the two values $CC[u][w]$ and $CC[v][w]$.

Note that the term $4 \cdot min - max$ measures the commonality of nodes u and v with respect to w. If min and max are equal, sameness is increased by $3 \cdot min$; if min is zero or if max is considerably greater than min, sameness is decreased by max. Intuitively, this bias toward increasing the sameness affords some leeway in how close min and max must be in order to still have a positive impact on the sameness value; this is because the random walk cannot guarantee to visit u and v equally often even if they look identical to the rest of the circuit.

As shown in Figure 5, algorithm RW-ST first finds and processes all cycles in the random walk, then computes sameness for all node pairs, and finally clusters those node pairs with sameness greater than zero. In some sense, the sameness computation within the random walk implicitly compares the neighborhood structures of a given node pair. The time complexity of RW-ST is a function of the time required to process

RW-ST(G)
Input: A graph G
Output: A set of clusters C

Construct a random walk RW on G
Find-Cycles(RW)
for each node u in G
    $C(u) := u$
for each pair of nodes u and v in G
    $S :=$ Sameness(u, v)
    if $S > 0$
        $C(u) := C(u) \cup C(v)$

Figure 5: High-level description of RW-ST.

the random walk and the time required to calculate sameness for all node pairs. As mentioned above, we use a random walk of length $O(n^2)$ and find all cycles in the random walk in $O(n^2)$ time. Processing a cycle of length l_c requires $O(l_c)$ operations, yielding worst-case time complexity of $O(n^3)$ to process the random walk. However, in practice the average l_c value seems to grow sublinearly in n. Calculating the sameness of a node pair requires $O(n)$ operations, resulting in $O(n^3)$ time to calculate sameness values for all $O(n^2)$ node pairs. Since processing the random walk and calculating sameness values both have complexity $O(n^3)$, the overall worst-case complexity of RW-ST is $O(n^3)$. RW-ST is observed to be much faster since most node pairs have no cycles in common, thus eliminating the need to calculate their sameness. The space requirements of our heuristic are $O(n^2)$ because the CC array records the cycle count for each node pair. Sparse matrix techniques can be used to reduce the required space at the expense of added time complexity.

4 Experimental Results

We tested the RW-ST method on two very distinct classes of inputs: (i) the random clustered inputs $G_{Gar}(m, n, p_{int}, p_{ext})$ studied by Garbers et al. [7], and (ii) the Primary and Test circuit netlists from the MCNC benchmark suite. Three different experiments were performed: (1) discovery of known clusters in G_{Gar} graphs; (2) DS measures of MCNC benchmark clusterings generated by RW-ST and the matching based compaction (MBC) scheme of Bui et al. [2]; (3) two-phase Fiduccia-Mattheyses (FM) style partitioning using RW-ST and MBC clusterings.

Garbers et al. [7] presented a class of random graphs $G_{Gar}(m, n, p_{int}, p_{ext})$, where m is the number of clusters, n is the size of a cluster, and an edge (u, v) is independently present with probability p_{int} if u and v are in the same cluster and probability p_{ext} otherwise. We used this class of random examples in our first set of experiments, to determine whether RW-ST could find the "correct" graph clustering. Our results are compared against the published statistics in [7]. The RW-ST algorithm was run on walks of length $(nm)^2$ and $10(nm)^2$ in order to see how walk length affected solution quality. The results of Table 1 show that RW-ST gives much more consistent results than (k, l)-connectivity. For walks of length $10(nm)^2$, RW-ST found 10 distinct clusters for each benchmark tested. In contrast, (k, l)-connectivity found "correct" clusterings for only two of the benchmarks, even when we allow the best results over a range of k values. This gives experimental confirmation of the self-tuning property inherent in RW-ST.

The "proper" field in the table indicates which of the 10-clustering or the 1-clustering (i.e., the complete circuit) has higher DS quality. Note that for $G_{Gar}(10, 100, 0.1, 0.004)$ these two values are nearly identical, i.e. this circuit no longer has an obvious clustering structure by our criterion.

The second set of experiments compared the RW-ST method with the matching based compaction (MBC) method of Bui et al. [2] by examining the DS quality of their respective clusterings on MCNC benchmarks. To ensure a "fair" comparison, we required the MBC clustering to have the same number of clusters as the RW-ST clustering. The original MBC results in [2] were based on constructing a clustering by finding a random maximal matching of the nodes. However, the number of clusters in an RW-ST clustering will normally be much less than half the original size of the circuit. We therefore modified the original MBC code to iteratively compute maximal random matchings, with each new matching performed on the graph induced from the previous clustering, until the desired reduction in problem size was obtained.

Table 2 shows the DS quality of the RW-ST and MBC clusterings. The RW-ST clusterings uniformly dominate the MBC clusterings in terms of DS quality. In addition, the improvement in DS quality is greater for larger circuits, possibly indicating that the random matching method breaks down as the problem size increases. For the two large examples Test04 and Test05, we observe improvements in DS quality of over 30%. Finally, note that the work of [4] only analyzed the Primary1 and bm1 benchmarks, obtaining DS qualities of 0.922 and 0.852, respectively.

To further confirm the greater utility of the RW-ST clusterings over MBC clusterings, we ran Fiduccia-Mattheyses (FM) partitioning on the resulting clustered graphs. These results are also summarized in Table 2, and we readily observe that the MBC clusterings produce very poor partitionings. This is somewhat surprising, since random matching based clustering was reported to be an efficient way of obtaining good initial starting points for the Kernighan-Lin approach [1] [2].

Our final experiments tested the original conjecture in [1], namely, that a good clustering will improve the solution quality of FM partitioning. For each heuristic clustering, we applied a two-phase FM algorithm which in the first phase partitioned the graph induced by the clustering, and then in the second phase used the expanded partition from the first phase as the starting point for FM partitioning on the "flat" circuit.

The results of this experiment are summarized in Table 3. Note that the results presented in Tables 2

m	n	p_{int}	p_{ext}	proper (by DS)	Garbers (k,l) Big/Small	RW-ST $(nm)^2$ Big/Small	RW-ST $10(nm)^2$ Big/Small
100	10	0.1	0.0001	10	(2,2) 9/3	10/57	10/20
100	10	0.1	0.0002	10	(2,2) 3/3	10/62	10/20
100	10	0.1	0.0003	10	(2,2) 3/0	10/90	10/24
100	10	0.1	0.0004	10	(2,2) 1/0	10/88	10/27
100	10	0.1	0.001	10	(3,2) 9/49	10/264	10/61
100	10	0.1	0.002	10	(3,2) 1/45	6/881	10/242
100	10	0.1	0.003	10	(3,2) 2/40	0/1000	10/427
100	10	0.1	0.004	10/1	(3,2) 1/40	0/1000	10/527

Table 1: Comparison of random walk based clustering with (k,l)-connectivity based clustering. Randoms walks of lengths $(nm)^2$ and $10(nm)^2$ were examined. The results give the numbers "Big" and "Small" for each clustering: following the presentation of Garbers et al., "Big" is defined as the number of clusters containing more than $\frac{1}{10}n$ nodes, while "Small" is the number of nodes that do not belong to any "Big" cluster.

		MBC			RW-ST		
Benchmark	Size	DS	Areas	Net cut	DS	Areas	Net cut
19ks	2844	1.166	5619:5383	456	1.578	5501:5501	153
bm1	882	1.189	1812:1668	94	1.221	2197:1283	39
PrimGA1	833	1.258	1719:1712	82	1.325	2180:1251	37
PrimSC1	833	1.258	1377:1376	91	1.325	1701:1052	40
PrimGA2	3014	1.238	4187:4186	303	1.566	4464:3909	154
PrimSC2	3014	1.238	3877:3829	266	1.566	4079:3627	145
Test02	1663	1.231	38141:18909	75	1.593	37132:19918	42
Test03	1607	1.185	14748:7481	132	1.566	12629:9600	74
Test04	1515	1.297	21105:20935	61	1.879	21055:20985	45
Test05	2595	1.275	62437:10161	51	1.689	39067:33531	10
Test06	1752	1.331	8485:8483	381	1.367	9444:7524	89

Table 2: DS qualities and Fiduccia-Mattheyses partitioning results of RW-ST and MBC clusterings.

and 3 are the best of 20 trials. We compared the results from running the two-phase FM partitioning algorithm on RW-ST and MBC clusterings against the results from running FM partitioning on the original circuit. Also in conformance with [2] we verified that the average degrees of the MBC clustering graphs were all greater than three (in fact, they ranged from 8 to 15, which more than meets the criterion given by Bui et al. [2] for the two-phase strategy to return "near-optimal" Kernighan-Lin results). In both cases there was a significant improvement over the standard FM solution quality, with a 12% improvement obtained using MBC clusterings and a 17% improvement obtained using RW-ST clusterings. These results in some sense confirm the conclusions of [2].

An interesting observation is that the huge discrepancy in FM partition quality between the RW-ST and MBC clusterings, as shown in Table 2, are not reflected in the two-phase FM partitioning results, i.e., a large improvement in the quality of the starting partition does not translate into a correspondingly large increase in the quality of the final partition.

5 Extensions

There are many promising directions for future work. We are currently pursuing a parallel implementation of the random walk methodology. In other words, we partition the random walk computation evenly among p available processors; the cycle-finding within the random walks is also performed on separate processors. This is appropriate for two reasons: (i) the hierarchical organization and sparsity of real netlist graphs permit only very short self-avoiding walks (i.e., cycles), and so little information is lost by breaking the random walk up among several processors; and (ii) results of Coppersmith et al. [5] show that the separate walks together will reproduce a single long walk.[4] This parallel approach would achieve perfect speedup over our current uniprocessor formulation.

[4]This is in the sense that two random walks will "collide" within a very short time when the graph is of low maximum degree and small diameter, as is the case with netlist graphs.

		Standard FM		MBC		RW-ST	
Benchmark	Size	Areas	Net cut	Areas	Net cut	Areas	Net cut
19ks	2844	5501:5501	151 (1.000)	5501:5501	156 (1.033)	5501:5501	146 (0.967)
bm1	882	1740:1740	65 (1.000)	1740:1740	54 (0.831)	1740:1740	58 (0.892)
PrimGA1	833	1716:1715	66 (1.000)	1718:1713	48 (0.727)	1716:1715	47 (0.712)
PrimSC1	833	1377:1376	59 (1.000)	1377:1376	61 (1.034)	1377:1376	58 (0.983)
PrimGA2	3014	4187:4186	242 (1.000)	4187:4186	187 (0.773)	4187:4186	165 (0.682)
PrimSC2	3014	3853:3853	235 (1.000)	3858:3848	175 (0.745)	3853:3853	159 (0.677)
Test02	1663	37132:19918	42 (1.000)	37132:19918	42 (1.000)	37132:19918	42 (1.000)
Test03	1607	11115:11114	84 (1.000)	13729:8500	59 (0.702)	13188:9041	71 (0.845)
Test04	1515	40732:1308	12 (1.000)	40938:1102	20 (1.667)	40932:1108	14 (1.167)
Test05	2595	38753:33845	24 (1.000)	62586:10012	4 (0.167)	39089:33509	5 (0.208)
Test06	1752	8484:8484	87 (1.000)	8484:8484	83 (0.954)	8484:8484	82 (0.943)

Table 3: Comparison of two-phase Fiduccia-Mattheyses partitioning of random walk clusterings and random matching based clusterings. Standard Fiduccia-Mattheyses partitioning results are included as a control. RW-ST clusterings lead to a 17% improvement in net cut over standard FM.

We also hope to use the DS quality measure as the basis of other "implicitly global" clustering methods. Certainly, standard combinatorial methods and direct epitaxial-growth approaches can both be modified to incorporate the DS criterion within the clustering objective. Finally, the concept of a "natural clustering" – one that is independent of both the number and size of the clusters – gives rise to new and interesting layout problems. In particular, the placement phase of layout becomes one of placing *malleable*, variable-size clusters which are of varying DS quality; this is certainly of independent research interest. Following the basic premise of our work, the natural clustering will also enable use of more sophisticated optimizations such as the spectral and relaxation methods in the context of "fast placement" for the next generation standard-cell and sea of gates designs.

Acknowledgements

Fiduccia-Mattheyses code was provided by J. Cong and M. Smith. We are also grateful to A. Steger for providing access to the preliminary results of [7].

References

[1] T. N. Bui, S. Chaudhuri, F. T. Leighton and M. Sipser, "Graph Bisection Algorithms with Good Average Case Behavior", *Combinatorica* 7(2) (1987), pp. 171-191.

[2] T. N. Bui, "Improving the Performance of the Kernighan-Lin and Simulated Annealing Graph Bisection Algorithms", *Proc. ACM/IEEE Design Automation Conf.*, 1989, pp. 775-778.

[3] C. K. Cheng and E. S. Kuh, "Module Placement Based on Resistive Network Optimization", *IEEE Trans. on CAD* 3(1984), pp. 218-225.

[4] J. Cong, L. Hagen and A. B. Kahng, "Random Walks for Circuit Clustering", *Proc. IEEE Intl. Conf. on ASIC*, June 1991, pp. 14.2.1 - 14.2.4.

[5] D. Coppersmith, P. Tetali and P. Winkler, "Collisions Among Random Walks on a Graph", to appear in *SIAM J. Discrete Math.*.

[6] W.E. Donath, "Logic Partitioning", in *Physical Design Automation of VLSI Systems*, B. Preas and M. Lorenzetti, eds., Benjamin/Cummings, 1988, pp. 65-86.

[7] J. Garbers, H. J. Promel and A. Steger, "Finding Clusters in VLSI Circuits", (*preliminary version* of paper in) *Proc. IEEE Intl. Conf. on Computer-Aided Design*, 1990, pp. 520-523. Also *personal communication*, A. Steger, April 1992.

[8] L. Hagen and A. B. Kahng, "Fast Spectral Methods for Ratio Cut Partitioning and Clustering", *Proc. IEEE Intl. Conf. on Computer-Aided Design*, 1991, pp. 10-13.

[9] L. Hagen and A. B. Kahng, "A New Approach to Effective Circuit Clustering", *technical report* UCLA CSD TR-920041, 1992.

[10] J. D. Kahn, N. Linial, N. Nisan and M. E. Saks, "On the Cover Time of Random Walks on Graphs", *J. of Theoretical Probability* 2(1) (1989), pp. 121-128.

[11] B. W. Kernighan and S. Lin, "An efficient heuristic for partitioning graphs", *Bell Syst. Tech. J.* 49(2) (1970), pp.291-307.

[12] C. Sechen and K. W. Lee, "An Improved Simulated Annealing Algorithm for Row-Based Placement", *Proc. IEEE Intl. Conf. on Computer-Aided Design*, 1987, pp. 478-481.

[13] R. S. Tsay and E. S. Kuh, "A unified approach to partitioning and placement" in *Proc. Princeton Conf. on Inf. and Comp.*, 1986.

[14] Y. C. Wei and C. K. Cheng, "Towards efficient hierarchical designs by ratio cut partitioning", in *Proc. IEEE Intl. Conf. on Computer-Aided Design*, 1989, pp. 298-301.

[15] Y.C. Wei and C.K. Cheng, "A Two-Level Two-Way Partitioning Algorithm", *Proc. IEEE Intl. Conf. on Computer-Aided Design*, 1990, pp. 516-519.