Random Walks for Circuit Clustering

Jason Cong, Lars Hagen and Andrew Kahng

UCLA Computer Science Dept., Los Angeles, CA 90024-1596

Abstract

We introduce a fast, parallelizable approach to circuit clustering based on analysis of random walks in the netlist. The method yields good clustering solutions for classes of "difficult" inputs in the literature as well as for industry benchmark circuits. We characterize our results using a new clustering metric which facilitates comparison with future work. Extensions to a number of other CAD applications are proposed.

1 Introduction

Advances in process technology have improved the switching speed of circuit elements at a much faster rate than corresponding improvements have occurred in packaging technology. This has led to a "hierarchy bottleneck", where choice of high-level organization dramatically constrains system performance. To address this issue, new packaging approaches have been developed (e.g., MCM technology), and hierarchical top-down design has become an important CAD paradigm. In particular, high-level *partitioning* and *clustering* problems have attracted much attention.

Partitioning is important for hierarchy-related performance optimization (e.g., signal time-of-flight across system partitions, lossy transmission line effects, etc.), and because early partitioning decisions in top-down synthesis strongly constrain the quality of the final layout. Clustering analysis of the circuit netlist is traditionally distinguished from partitioning as a bottom-up process arising in floorplanning, constructive initial placement, and even multi-way partitioning when the sizes of the partitions are not known *a priori*. However, as design complexity increases, top-down partitioning and bottom-up clustering have been used to complement each other in addressing the same issues. A primary use of clustering is for aggregation analysis of the netlist, i.e., so that small clusters can be treated as single elements, resulting in a smaller, "condensed" instance for top-down partitioning. Without this simplification, newer designs can have complexity beyond the capabilities of traditional partitioning algorithms or their hardware platforms. More importantly, the solution quality of iterative partitioning methods (particularly Kernighan-Lin and simulated annealing) does not scale well with problem size, and such authors as Bui et al. [1] and Lengauer [8] have noted that a good clustering will improve the performance of such partitioning algorithms. Finally, traditional partitioning methods do not readily identify "natural" divisions which might be discovered by bottom-up analysis [9]. The surveys of Donath [4] and Lengauer [8] give additional discussion of partitioning and clustering.

The remainder of this paper is organized as follows. In Section 2, we discuss previous work in clustering, particularly the recent (k,l)-connectivity method of Garbers et al. [5]. Section 3 proposes a new approach to clustering: we infer graph structure from the node sequence generated as we iteratively move to a random adjacent node in the netlist. Two theoretical results bound the required length of the random walk; another result shows that dense, highly clustered graph inputs will be correctly analyzed via the random walk process. Section 4 develops a practical clustering algorithm for netlist hypergraphs, and Section 5 gives experimental results for both random input classes from the literature as well as industry benchmark circuits. The random walk method has a number of attractive features, including stability and perfect parallelizability, which are increasingly useful as design complexity increases. We conclude in Section 6 by listing several extensions and a number of additional applications of the random walk method.

2 Previous Work

Previous work in clustering is limited, primarily relying on intuitive notions of *edge density* among sets of modules. For example, if more than $\epsilon \cdot C(c,2)$ edges are present among c vertices in the netlist graph $G = (V,E)$, then the c vertices are said to form a cluster. Practical use of such a definition is impossible since it requires checking all subsets of V with cardinality c. Thus, the related concept of (k,l)-connectivity was recently used by Garbers et al. [5] for circuit clustering.

If there are k edge-disjoint paths of length l between nodes u and v, then u and v are said to be (k,l)-connected; [5] showed that for certain highly structured inputs, the transitive closure of the (k,l)-connectedness relation gives an equivalent clustering to that induced by the edge density criterion. Indeed, on some instances of a class of random inputs and for a highly structured standard-cell benchmark, the (k,l) criterion yields reasonable solutions. However, the connectivity based method suffers from two main weaknesses. First, although determining (k,l)-connectivity is more "algorithmically tractable" than checking edge density, the methods proposed in [5]

91TH0379-8/91/0000-P14-2.1$01.00

still require time exponential in k and l, even for $l = 2$. Second, (k,l)-connectivity yields nonintuitive results: nodes v_i and v_j can belong to a cluster even when no node on any path between v_i and v_j belongs to the cluster (e.g., a cycle of length four through nodes A, B, C and D will be broken into an (A,C) cluster and a (B,D) cluster by the $(2,2)$-connectivity criterion; this solution has twice the cutsize of the more natural $(A,B)(C,D)$ clustering).

With this in mind, we wish to develop an efficient method that returns intuitively reasonable circuit clusterings. In examining the literature, one notes that the lack of previous work is in no small part due to the difficulty of even defining the clustering problem. While a number of metrics have been devised to assess the quality of a graph partition, the nature of a "good" circuit clustering is still very much an open issue. Without knowledge of what constitutes an optimal clustering, researchers have thus constructed examples for which the desired output is known, and measured clustering heuristics by their ability to output the correct answer. (Although we evaluate our new method with respect to such input classes, note that Section 5 proposes a highly general, new metric for circuit clustering.)

Two classes of "difficult" instances for partitioning and clustering are respectively due to Bui et al. [1] and Garbers et al. [5]. The first is given by the random graph model $G_{Bui}(n,d,b)$ and was developed in [1] to analyze graph bisection (partitioning) algorithms. A graph in $G_{Bui}(n,d,b)$ has n nodes, is d-regular and has minimum bisection width almost certainly equal to b. The second input class is given by the $G_{Gar}(c,m,p_{int},p_{ext})$ random model of [5], which prescribes c clusters of m nodes each, with each of the $c \cdot C(m,2)$ edges inside clusters independently present with probability p_{int} and each of the $m^2 \cdot C(c,2)$ edges between clusters independently present with probability p_{ext}.

The essential quality of the G_{Bui} and G_{Gar} inputs is that they have optimal cutsize (i.e., the number of edges between clusters) that is significantly smaller than the optimal cutsize of a random graph with similar node and edge cardinalities. It has been shown that on these instances, traditional Kernighan-Lin and simulated annealing methods usually return solutions an unbounded factor worse than optimal [1]. For inputs in $G_{Bui}(n,3,b)$, these standard approaches give bisection results that are no better than *random* solutions, a disturbing observation which further motivates clustering as an alternative way of dealing with difficult partitioning instances. As discussed below, our approach can effectively deal with such "difficult" partitioning instances.

3 Random Walks in Graphs

Our main contribution is a new methodology which computes a circuit clustering based on *random walks* in the netlist hypergraph. A *random walk* on a graph is defined to be a discrete-time stochastic process which iteratively moves from the current vertex in the random walk to a random adjacent vertex, with all adjacent vertices (i.e., all transitions) equiprobable. Our method relies on the fact that such a random walk will with high probability quickly capture the graph structure. It is instructive to consider the progress of a random walk on the "barbell" example of Figure 1, which consists of two cliques joined by a chain. With some thought, a number of standard probabilistic results (cf. [2] [7]) are clear: (i) if we start at x, then all nodes in cluster A will be visited within the first $O(n \log n)$ steps, but (ii) the walk will not escape to the middle chain until $O(n^2)$ steps have been taken; if we consider the same example with cluster A deleted, then (iii) a walk from y will require $O(n^2)$ steps to reach z, but (iv) a walk from z will need $O(n^3)$ steps to reach y (since every time we return to z, the walk will wander around cluster B before again escaping into the chain).

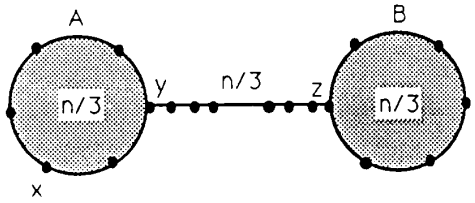

Figure 1: "Barbell" graph example.

We may define the *cover time* of G as the maximum, over all possible starting vertices, of the expected length of a random walk that visits all vertices in G. The following results are found in [2] [7] or are simple to derive:

Theorem 1: For a dense connected graph $G = (V,E)$ with all node degrees $\geq |V|/2$, the cover time of G is $\Theta(n \log n)$. □

Theorem 2: In a random clustered graph with all node degrees $\geq c_1 \cdot |V|$, all clusters of size $\leq 2c_1 \cdot |V|$ and total intercluster cutsize $O(\sqrt{|V|})$, two random walks of length $\Theta(|V|^{1.5})$ from v_1 and v_2 will encounter identical node sets with probability $1 - o(1)$ exactly when v_1 and v_2 belong to the same cluster. □

Theorem 2 implies that in a graph with dense clusters, we may compare *short* random walks starting at two nodes, and reliably determine whether the nodes belong together in a cluster. A slightly weaker result holds for sparse graphs [3]. We can also show that when the input is sparse (e.g., netlists are degree-bounded by a constant due to fanout/pinout limitations), the cover time remains small:

Theorem 3: For a sparse connected graph $G = (V,E)$ that is d-regular with $d \leq n/2$, the cover time of G is $\Theta(n^2)$. □

Thus, we may reasonably use a *single* random walk to sample the entire graph, instead of computing separate walks from every vertex.

4 A Practical Algorithm

Two basic issues must be addressed to derive a practical clustering algorithm from the theoretical results of Section 3. The first issue is definition of the random walk transition probabilities between vertices (modules) of the netlist *hypergraph*, since the number of modules in each signal net (i.e., hyperedge sizes) can vary greatly. Because the random walk should remain within each natural cluster for a "long" time, larger signal nets should have smaller probability of traversal, since they have a greater chance of leading out of the current cluster. We have found the following natural transition weighting to be effective. The *net sum* of the current module v_{i-1}, denoted $sum_{v_{i-1}}$, is $\sum_{s_k \ni v_{i-1}} 1/|s_k|$, where $|s_k|$ denotes the number of pins in signal s_k. From v_{i-1}, the walk will traverse an incident signal net s_k with probability inversely proportional to $|s_k|$, i.e., $netprob(s_k|v_{i-1}) = \frac{1/|s_k|}{sum_{v_{i-1}}}$. Once signal net s_i has been chosen for traversal, the next module in the walk is chosen from the other $|s_i| - 1$ modules in s_i with uniform probability $nodeprob(v_i|s_i) = 1/(|s_i| - 1)$.

Given this random walk process, the second issue is how to extract useful clustering information from the random walk within a reasonable time bound. The obvious approach motivated by Theorem 2 would be to look at two random walks respectively starting from nodes v_i and v_j, then assign v_i and v_j to the same cluster if the random walks are sufficiently similar. However, without prior knowledge of the netlist structure we cannot prescribe a reasonable similarity measure, nor can we exactly prescribe the length of the random walks (each walk should be long enough to explore its "natural cluster", but not long enough to escape the cluster). Furthermore, because comparing two random walks requires time at least linear in the combined length of the walks, testing all pairs of walks (i.e., all pairs of starting nodes) is impractical. Therefore, we examine the sequence of modules encountered by a single random walk in the netlist, and use the notion of *recurrence* in the random walk to efficiently determine clustering structure.

Consider the sequence of nodes $\{v_0, v_1, \ldots, v_{N-1}\}$ encountered during the random walk. A *cycle* of the random walk 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 *maximum cycle* for each node v_j, denoted $C(v_j)$, is the longest cycle in the random walk which begins and ends at v_j. The set of modules in each $C(v_j)$ intuitively corresponds to (part of) a natural cluster, for two reasons: (i) if there is a more tightly coupled subset, then the random walk will recur within that subset and we would not have found the $C(v_j)$ cycle; and (ii) it is unlikely that the random walk will leave a cluster and return to the same cluster without traversing some smaller cycle in the process. Based on this idea, our algorithm is as follows. We compute a random walk in the netlist, find $C(v_j)$ for all modules v_j, and then define the relation $\bowtie$ by $v_a \bowtie v_b$ if $v_a \in C(v_b)$ and $v_b \in C(v_a)$. The transitive closure of the $\bowtie$ relation (equivalently, the connected components of the *clustering graph* induced by $\bowtie$) yields the netlist clustering. This method, which we call Algorithm RW1, is summarized in Figure 2.

```
/* Compute a random walk of length N */
v_0 = starting node in walk
for i = 1 to N - 1
    s_i = random signal net, using netprob(s_i|v_{i-1})
    v_i = i^th node in walk, using nodeprob(v_i|s_i)
Return sequence {v_i} of nodes in walk

/* Perform clustering analysis */
for j = 1 to |V|
    Compute C(v_j)
for i = 1 to |V|
    for j = i + 1 to |V|
        Add edge (i, j) to clustering graph if v_i ⋈ v_j
Output connected components of clustering graph.
```

Figure 2: Summary of Algorithm RW1.

5 Experimental Results

We ran Algorithm RW1 on random graphs from the class G_{Bui} with between 100 and 800 nodes, following the parameters (n, d, b) reported by Bui et al. (Table I, p. 188 of [1]). We also generated a number of 1000-node and 500-node examples of clustered inputs in G_{Gar}, using the same values (c, m, p_{int}, p_{ext}) as in Table 1 of [5]. Typical results are shown in Table 1 below: six 200-node G_{Bui} examples and three 500-node G_{Gar} examples are listed, with RW1 results for random walks of length $n^{1.5}$, n^2 and $n^{2.5}$. For the G_{Gar} results, g denotes the number of large clusters found, and l denotes the number of "loose" nodes which did not belong to any large cluster. Since the G_{Bui} instances represent 2-clustered graphs, our results show the number of nodes assigned to the two largest clusters, A and B. The results demonstrate that walks of length $O(n^2)$ are indeed appropriate for sparse, degree-regular instances of the G_{Bui} and G_{Gar} classes; for most examples, the results are comparable to or superior to those reported in [5].

To measure the quality of a clustering when the "correct" solution is not known, we propose to use the weighted average of the cluster (*degree/separation*) (DS) *quality*: (i) cluster *degree* is the average number of nets incident to each node 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 $= \infty$ if two nodes in the cluster are disconnected. The DS measure is highly robust: it accounts for multiplicity of connections, is not unduly sensitive to small perturbations of the netlist, and provides a very general metric for clustering solutions. (The DS measure can be slightly harsh: if two disconnected nodes are placed together in a cluster, the entire cluster receives DS quality $= 0$.) We believe that the DS metric is suitable as a

Bui Examples			$n^{1.5}$		n^2		$n^{2.5}$	
n	d	b	A	B	A	B	A	B
200	3	2	2	5	10	12	89	73
200	5	2	4	4	33	33	99	97
200	9	2	7	7	86	84	100	99
200	3	10	3	3	8	6	81	86
200	5	10	3	7	54	20	89	96
200	9	10	6	4	84	93	196	1

Garbers Examples				$n^{1.5}$		n^2		$n^{2.5}$	
c	m	p_{int}	p_{ext}	g	l	g	l	g	l
5	100	0.1	0.00025	5	383	5	40	4	2
5	100	0.1	0.00050	5	409	5	49	2	9
5	100	0.1	0.00075	0	500	2	71	1	12

Table 1: RW1 for G_{Bui} and G_{Gar} inputs.

standard measure of clustering quality.

We have used RW1 to cluster several industry benchmarks, including *Primary1* from the MCNC test suite and *bm1*, a Hughes benchmark discussed in [9]. For these two circuits, Table 2 lists DS statistics for the entire netlist, for the two largest clusters found by RW1, and average DS quality over all clusters found by RW1. Note that the DS average is significantly higher for the clusters than for the entire netlist, implying that the RW1 algorithm indeed discovers tightly coupled subsets of nodes.

	Primary1 # nodes	cluster degree	cluster separation	DS quality
Total	833	3.491	4.467	0.782
1st	50	3.100	3.236	0.958
2nd	40	3.675	2.458	1.495
Average over all RW1 clusters				0.922

	bm1 # nodes	cluster degree	cluster separation	DS quality
Total	882	3.299	4.810	0.686
1st	71	3.451	3.046	1.133
2nd	45	3.667	2.823	1.299
Average over all RW1 clusters				0.852

Table 2: DS statistics for Primary1 and bm1.

6 Conclusions

We have proposed a new methodology for extracting circuit clustering structure via random walks on the netlist hypergraph. Results on "locality" in netlist structure [6] suggest that the bottom-up random walk based clustering, may well yield more useful results than traditional top-down recursive partitioning. Our method is provably good for dense clustered graphs, and gives good empirical results for the difficult input classes of [1] [5] as well as industry benchmark designs. In analyzing the industry benchmark designs, for which no "correct" solution is known, we have characterized our clustering results using a new clustering metric which may standardize comparisons between current and future algorithms.

Our random walk approach has several computational advantages, notably the perfect parallelizability of taking the random walk and computing the $C(v_j)$. Furthermore, a random walk of length $O(n^2)$ can reliably cover a degree-regular graph and capture its clustering structure; this theoretical result is confirmed in our experiments. Since all cycles $C(v_j)$ in the random walk can be found in time linear in the length of the walk, our algorithm is more practical than traditional methods of analyzing graph structure, which involve $\Omega(n^3)$-complexity matrix computations.

A number of extensions are the subject of current work. (1) Maximum cycles in the random walk can be merged into clusters via an *overlap* measure between two cycles. If the overlap between two cycles $C(v_i)$ and $C(v_j)$ exceeds a threshold parameter τ, then $v_i \diamond v_j$; the transitive closure of the $\diamond$ relation gives a circuit clustering, as noted in [3]. The τ parameter may be varied to force a prescribed number of clusters in the final output, giving an added degree of freedom beyond the variable length of the random walk itself. (2) We can vary the transition probability model used in computing the random walk, depending on design parameters and prevailing technology. For example, changing the relative weighting of large nets and small nets will yield a qualitatively different circuit clustering. (3) As noted above, the RW1 algorithm has perfect parallel speedup; we are investigating implementation on multiprocessor platforms. (4) Finally, the random walk methodology is very general, and is applicable whenever we must implicitly analyze the structure of a netlist. Thus, recursive partitioning and floorplanning, layout area prediction, wireability estimation, etc. all offer promising areas of application, and are the subject of current investigation [3].

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] A. Chandra, P. Raghavan, W. L. Ruzzo, R. Smolensky and P. Tiwari, "The Electrical Resistance of a Graph Captures its Commute and Cover Times", *Proc. ACM Symp. on Theory of Computing*, May 1989, pp. 574-586.

[3] J. Cong, L. Hagen and A. Kahng, *draft*, June 1991.

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

[5] J. Garbers, H. J. Promel and A. Steger, "Finding Clusters in VLSI Circuits", *extended version* of paper in *Proc. IEEE Intl. Conf. on Computer-Aided Design*, 1990, pp. 520-523.

[6] L. Hagen and A. B. Kahng, "Fast Spectral Methods for Ratio Cut Partitioning and Clustering", to appear in *Proc. IEEE Intl. Conf. on Computer-Aided Design*, Santa Clara, November 1991.

[7] 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.

[8] T. Lengauer, *Combinatorial Algorithms for Integrated Circuit Layout*, Wiley-Teubner, 1990.

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