Clusters First? A Study of Circuit Structure and Placement

Satoshi Ono and Patrick H. Madden
SUNY Binghamton Computer Science Department
Box 6000, Binghamton NY 13902
and
The University of Kitakyushu
Kitakyushu, Japan

Abstract—**M**ultilevel clustering has proven to be an effective technique for a number of problems. In this paper, we study a clusters-first approach to circuit placement, compare it to more traditional methods, and study the relationship between cluster sizes and placement quality. We also reveal shortcomings in recently developed benchmarks, that can serve to bias experiments towards a clusters-first methodology.

I. Introduction

Over the past few decades, the number of transistors in processors has been growing at an exponential rate. Every 18 months, the number of transistors on a chip roughly doubles, and circuit design problems become more complex. There is a great need for improvements to physical design tools, both to handle the increased design sizes, and to address the growing problem of interconnect delay.

As interconnect wires now dominate system delay[7], there is a need to reduce the length of wires in high performance designs[1]. Optimization of this sort is primarily addressed by improved circuit placement; recent experiments suggest that there is a significant degree of suboptimality[5].

In recent years, multilevel clustering has resulted in a great deal of improvement on a variety of problems. Perhaps best known is the impact on partitioning. An early paper by Cong[6] performed a limited amount of clustering, reducing the size of a hypergraph; the authors then used the well known Fiduccia-Mattheyses[11] heuristic to perform partitioning. The limited amount of clustering proved beneficial; further clustering provided even greater gains, resulting in the state-of-the-art multilevel clustering partitioners such as hMetis[13] and ML-Part[3]. These partitioners are used in bisection-based placement tools such as Feng Shui[1] and Capo[4].

Recently, Hu and Marek-Sadowska[12] presented a placement approach that first applies a limited amount of clustering, reducing the size of the placement problem; the authors then used Capo to perform placement. The primary impact of the approach was improved run times; there was also a modest improvement in placement wire length. The parallels between single and multilevel clustering for partitioning, and current placement efforts, is striking. The focus of this work is on an investigation of multilevel clustering for placement. We find that there are fundamental problems with cluster creation that present significant challenges for placement; the nature of these problems also reveals shortcomings in recently presented synthetic benchmarks.

The remainder of this paper is organized as follows. In Section II., we provide a brief summary of prior work in placement and in multilevel clustering based partitioning. We next present a "clusters first" placement approach in Section III.; we utilize a clustering tree within a combinatorial optimization framework. Section IV. presents experimental results, comparing our work to other recently developed tools. We conclude the paper with Section V..

[1]Wire length minimization alone is not equivalent to delay minimization: we focus on wire length here, as it is a commonly used metric, and easy to quantify.

II. Background

A hypergraph $G(V,E)$ is a graph with vertex set V and hyperedge set E. The degree of each vertex v in V, $d(v)$, is the number of edges incident to it. Each vertex v has a size $s(v)$. The degree of hyperedge e, $d(e)$, is the number of vertices incident to it. Number of edges between a vertex v and a vertex u is denoted by $N(v,u)$.

A clustering tree is a binary tree, in which the leaves are the individual vertices. Internal nodes of the tree are clusters of vertices. If all of the vertices are clustered, there is a single tree; the root of the tree represents all of vertices. At intermediate stages of the clustering process, there may be a number of separate trees, representing portions of the graph. When vertices are to be grouped together in a clustering based approach, the connectivity between vertices v and u is normally denoted as $W(u,v)$: a variety of functions to compute $W(u,v)$ have been proposed.

In VLSI design, a "cell" is roughly equialent to a vertex, while the signal net connecting a group of cells is roughly equivalent to a hyperedge. We refer to a "pin" as the connection between a vertex and a edge, so number of pins on a vertex v is the same as $d(v)$, and the number of pins of a edge e is the same as $d(e)$.

We will formulate the placement problem as one of assigning vertices of a hypergraph to physical locations; the objective is to minimize the total (bounding box) length of all hyperedges. Objectives for partitioning are to minimize the total number (or weight) of cut edges, subject to area constraints on each side.

A. Multilevel Partitioning

Partitioning is a well studied problem. Methods by Kernighan and Lin[14], and Fiduccia and Matthyeses[11] are well known, and the basic methods developed by these authors are used in many current implementations. An improvement in solution quality was obtained through the use of multilevel clustering[13]; a clustering tree is first constructed, with traditional partitioning methods being used to find an initial solution, and then refine the solution as the tree is unclustered.

B. Circuit Placement

In this section, we briefly discuss the issues involved in Standard Cell Placement. Standard cells are the smallest indivisible components of a VLSI design; when placing these cells on a wafer of silicon, we have many candidate positions.

Our objective is to find a placement of standard cells that gives us a minimum total wire length. The total wire length required is simply the sum of the lengths for each interconnect tree required to complete the connections.

Assuming that we have roughly the same number of candidate positions as standard cells to be placed, we can estimate the order of complexity involved in finding an optimal solution. If there are n cells, and we find all permutations of the placement, we end up with complexity $O(n!)$. We would also need to calculate the total wire length of the placement using the netlist.

With the number of cells reaching hundreds of thousands or millions, it is obvious that a brute force mechanism to find an optimal solution is not feasible. The placement problem is in fact NP-hard, resulting in the wide use of heuristics. Common methods for standard cell placement include recursive bisection, analytic methods, and simulated annealing.

In partitioning based placement (such as [2], [8], [4], [1]), we partition the standard cells recursively into two parts such that there is minimum crossover of connections between the cells over the partition. Doing this recursively ensures that the cells that are connected remain relatively close to each other and cells that do not belong to the same net remain further away from each other.

Analytic methods, such as [16], [9], [17], model the placement problem with linear programming-like methods. Cells in a placement can be viewed as nodes, connected by a set of springs. Attractive and repulsive forces are assigned between cells, and the manipulation of these forces will results in the relative positioning of objects. By repeatedly solving linear (or quadratic) equations which represent the forces, and then adjusting the positions of individual cells, a "low energy" solution can be found. This corresponds to a placement with good total wire length.

Simulated Annealing[15], [18], [19] is a probabilistic approach; cell positions are randomly changed, with each move being accepted or rejected based on a "temperature" based function. At high temperatures, both moves are accepted, with little regard for the impact of the move on an objective function. At lower temperatures, there is a preference for moves that improve the objective function. By allowing the chance that a move that degrades solution quality will be accepted, annealing can escape local minima. Run times for annealing methods are typically high, but solution quality is in general quite good.

III. Clusters First Approach

It is natural to suspect that the success of multilevel clustering for partitioning can be extended to placement. The initial success of the single-level clustering method of Cong and Smith[6] has parallels to a recent placement effort[12]. In our preliminary experiments, a simple top-down combinatorial approach was able to find optimal placements for mesh-like graphs, when provided with an initial optimal clustering tree[10].

In this section, we describe the construction of a cluster tree for our placement approach, and the combinatorial method we use to convert the clustering solution into a placement.

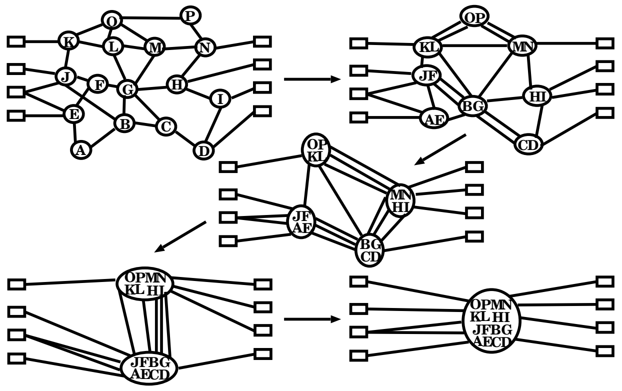

Fig. 1. Multilevel clustering; individual logic elements are repeatedly grouped, forming a heirarchy.

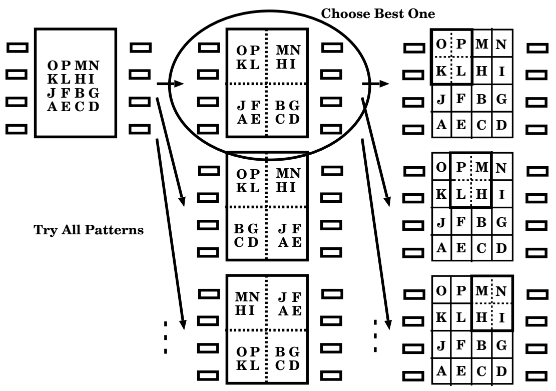

Fig. 2. Combinatorial Optimization during placement.

A. Clustering

As with most clustering methods, we merge smaller clusters into larger ones. A new cluster has all of the nets of the two child clusters except the nets between the two; the size is the sum of the two child clusters.

Construction of a good cluster tree requires a good objective function during clustering. In our experiments, we have considered a variety of connectivity objectives, $W(u,v)$, to determine the merit of merging cluster u and v. We use the cluster sizes, number of common nets, net cardinality, cluster degree, and difference between cluster sizes in a variety of ways. The information about the number of nets between them, $N(u,v)$, size of components of the pair, $s(u)$ and $s(v)$, and relationship to the other clusters, $d(u)$ and $d(v)$ and/or $d(e[u,v])$, is updated during clustering. Figure 1 illustrates the clustering process.

In terms of placement quality, perhaps the most significant element of successful clustering is to maintain clusters of roughly equal size throughout the process. For combinatorial-based placement, described in the next section, a mismatch of cluster sizes degrades solution quality considerably.

While there is a wide range of clustering objective functions, and extensive work on the subject, the experiments presented in Section IV. use a clustering tree obtained from the top-down application of the partitioner hMetis. Motivation for the use of this tree is that it in many respects represents the "best possible" clustering tree one might expect to obtain. As the partitioning is performed in a top-down manner, we find good locations to break the circuit into smaller clusters. Using a variety of different metrics (number of cut nets, ratio of internal to external nets, cluster sizes, etc.), the clustering tree obtained in this manner was better than any tree we were able to construct in a bottom-up fashion.

B. Combinatorial Placement

Following clustering, a placement can be generated by simply enumerating different locations for each of four "top level" clusters. After fixing the locations of the top four clusters, we continue to the next "level of the clustering tree, refining the placement further.

In our combinatorial approach, we operate by default on groups of four clusters, finding optimal configurations for the group. As refinement proceeds, optimization windows can span boundaries between clusters.

This process is illustrated in Figure 2. If our cluster tree has groupings of "JFAE," "BGCD," "OPKL," and "MNHI" as the top four clusters, the twelve different permutations are evaluated; the permutation with minimum wirelength is then selected. Each cluster is then divided into groups four smaller clusters, and the optimal configuration for each group of four is found. An optimization window then passes over the placement, potentially moving some small clusters from the top group across the earlier boundaries.

C. Placement Legalization

The size of clusters can vary somewhat, and it is in general unlikely that the area of a cluster will fit exactly into a fixed number of rows in a placement. The inital placement found can be considered "abstract," and legalization is required.

We use a dynamic programming method to assign cells to non-overlapping positions in rows. The method we use was described in [1], and was in fact developed first for the placement approach we focus here.

IV. Experimental Results

To evaluate the impact of a "clusters first" approach, we tested the approach described here on a number of recent benchmarks, and compare to state-of-the-art academic tools. Obviously, the quality of clustering impacts the results, and we use the "top-down" tree obtained by hMetis, for the reasons mentioned earlier.

A. Placement Images

To visualize the nature of clusters found, and their locations within a placement, we show Figures 3 to 6. The "top level" clusters are shaded in different colors, to show their locations. A good placement for a cluster is not necessarily a tightly packed rectilinear region.

B. IBM Benchmarks

In terms of wire length, the combinatorial approach did not obtain results that were as good as methods such as Feng Shui 2.0 and Dragon, although it does outperform Capo and mPL. Despite the high quality of the clustering tree, the impact of terminal propagation (essentially ignored during clustering) bounds solution quality. Results are shown in Table I.

From these experiments, we would suggest that while "pure clustering" can capture a great deal of the complexity of the placement problem, ignoring the physical component (and terminal propagation) limits the quality of results.

C. PEKO Benchmarks

While "clusters first" did not obtain the best results on the IBM benchmarks, it is significantly more effective when given the synthetic "PEKO" benchmarks. Results of these experiments are shown in Table II. This set of benchmarks contains only local nets; to develop a placement problem in which an optimal solution is known, nets are inserted into a regular grid. If a net of degree d is to be inserted, cells within a small rectangle are selected to be attached to the net; the size of the rectangle is required to have the smallest possible area that can contain d cells.

The benchmark construction can be viewed as heavily favoring a clustering approach; at the first level of clustering, when connected cells are grouped, *every grouping is consistant with an optimal placement.* This is clearly not the case with the IBM benchmarks, explaining the difference in performance of our "clusters first" approach, as well as that of the clustering-driven tool mPL[5].

The contrast in results leads to an observation: *it is unlikely that a placement with minimum-length nets is possible for the IBM benchmarks.* While the PEKO and IBM benchmarks are similar from a statistical perspective, it is likely that some nets in the IBM benchmarks would not be placed with minimum possible wire length in an optimal solution.

V. Conclusion

In this paper, we have studied a "clusters first" approach to circuit placement. Experiments indicate that even with high quality clusters, the effect of terminal propagation must be considered during placement if we wish to obtain high quality results. This contrasts with partitioning problems, in which terminal propagation is not a constraint.

While the combinatorial approach was not able to match leading the best bisection based tool or an annealing based tool, our research in this area did result in the development of the fractional cut approach and the dynamic programming based legalizer described in [1].

As part of our current work, we are continuing our investigation into the structure of placement benchmarks, and how this can be utilized to improve results. While the PEKO benchmarks are in some respects similar to industry designs, there are clearly aspects which favor some placement methods, while putting others at a disadvantage.

This work was supported in part by SRC under project 947.1, an IBM Faculty Partnership award, NYSTAR MDC, and an equipment grant from Intel.

References

[1] A. Agnihotri, M. C. YILDIZ, A. Khatkhate, A. Mathur, S. Ono, and P. H. Madden. Fractional cut: Improved recursive bisection placement. In *Proc. Int. Conf. on Computer Aided Design*, pages 307–310, 2003.

[2] M. A. Breuer. A class of min-cut placement algorithms. In *Proc. Design Automation Conf*, pages 284–290, 1977.

[3] A. E. Caldwell, A. B. Kahng, and I. L. Markov. Improved algorithms for hypergraph bipartitioning. In *Proc. Asia South Pacific Design Automation Conf.*, pages 661–666, 2000.

[4] Andrew E. Caldwell, Andrew B. Kahng, and Igor L. Markov. Can recursive bisection alone produce routable placements? In *Proc. Design Automation Conf*, pages 477–482, 2000.

[5] C. C. Chang, J. Cong, and M. Xie. Optimality and scalability study of existing placement algorithms. In *Proc. Asia South Pacific Design Automation Conf.*, pages 621–627, 2003.

[6] J. Cong and M. Smith. A parallel bottom-up clustering algorithm with

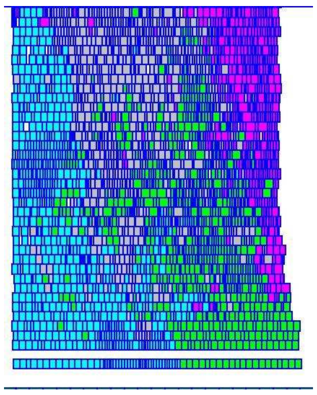
Fig. 3. Balanced clustering

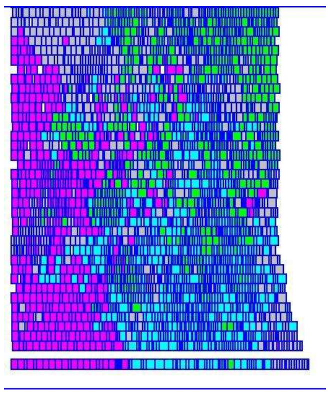
Fig. 4. Semi-balanced clustering

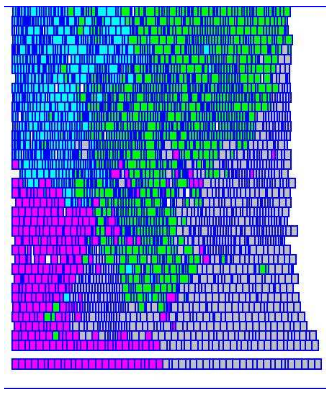
Fig. 5. Free clustering

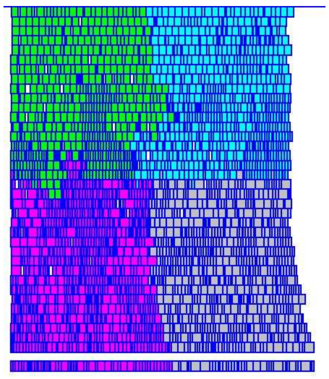
Fig. 6. hMetis clustering

Bench mark	Clustering Based	Feng Shui 2.0		Capo 8.6		Dragon 2.23		Kraftwerk (not legal)		mPL 2.0	
	wl	wl	×	wl	×	wl	×	wl	×	wl	×
ibm01	0.55	0.52	0.95	0.57	1.04	0.51	0.93	0.70	1.27	0.64	1.16
ibm02	1.52	1.50	0.99	1.60	1.05	1.44	0.95	2.15	1.41	1.61	1.06
ibm07	3.50	3.30	0.94	3.71	1.06	3.31	0.95	5.12	1.46	4.07	1.16
ibm08	3.95	3.60	0.91	3.89	0.98	3.39	0.86	4.66	1.18	4.25	1.08
ibm09	3.32	3.02	0.91	3.31	1.00	2.96	0.89	4.26	1.28	3.81	1.15
ibm10	6.05	5.66	0.94	6.34	1.05	5.61	0.93	7.61	1.26	6.61	1.09
ibm11	4.83	4.48	0.93	4.89	1.01	4.43	0.92	5.80	1.20	5.96	1.23
ibm12	8.28	7.74	0.93	8.76	1.06	7.60	0.92	10.41	1.26	9.44	1.14
Avg			0.94		1.03		0.92		1.29		1.13

TABLE I Average wire length results for the IBM version 2 "easy" benchmarks. All wire lengths are scaled by 10^8. Results for the placement tools *Kraftwerk* and *mPL* were provided by Prof. Igor Markov through his GSRC Bookshelf.EXE project. The placements by *Kraftwerk* are not legalized, and contain a number of overlaps; attempts to legalize the placements using *DOMINO* timed out or aborted on most benchmarks. All scaled wire lengths are relative to those of *Feng Shui 2.0*.

Bench mark	Known Optimal	Clustering Based		Feng Shui 2.0		Capo 8.6		Dragon 2.20		Kraftwerk (not legal)		mPL 2.0	
	wl	wl	×	wl	×	wl	×	wl	×	wl	×	wl	×
Peko01	0.81	1.28	1.58	1.25	1.54	1.29	1.59	1.46	1.8	1.39	1.72	1.10	1.36
Peko02	1.26	2.11	1.67	2.09	1.66	2.03	1.61	2.43	1.93	1.98	1.57	1.76	1.40
Peko03	1.50	2.49	1.66	2.62	1.75	2.66	1.77	2.93	1.95	3.02	2.01	2.05	1.37
Peko04	1.75	2.84	1.62	2.82	1.61	3.12	1.78	3.87	2.21	3.25	1.86	2.31	1.32
Peko05	1.91	3.10	1.62	3.01	1.58	3.16	1.65	3.79	1.98	3.92	2.05	2.57	1.35
Peko06	2.06	3.31	1.61	3.44	1.67	3.57	1.73	4.35	2.11	4.07	1.98	2.78	1.35
Peko07	2.88	4.81	1.67	4.91	1.70	5.07	1.76	6.24	2.17	5.73	1.99	3.95	1.37
Peko08	3.14	5.3	1.69	5.25	1.67	5.57	1.77	6.79	2.16	5.87	1.87	4.99	1.59
Peko09	3.64	6.06	1.66	5.98	1.64	6.47	1.78	7.72	2.12	8.52	2.34	4.76	1.31
Peko10	3.73	7.81	2.09	7.87	2.11	8.00	2.14	8.49	2.28	8.90	2.39	6.59	1.77
Avg			1.69		1.69		1.76		2.07		1.98		1.42

TABLE II Average wire length results for the PEKO suite 1 benchmarks. Results for the placement tools *Capo 8.6*, *Dragon 2.20*, *Kraftwerk* and *mPL 2.0* were provided by Prof. Igor Markov. All wire lengths are scaled by 10^6, and relative comparisons are to those of the known optimal solution.

applications to circuit partitioning in VLSI design. In *Proc. Design Automation Conf*, pages 755–780, 1993.

[7] Jason Cong, Lei He, Cheng-Kok Koh, and Patrick H. Madden. Performance optimization of VLSI interconnect layout. *Integration, the VLSI Journal*, 21:1–94, 1996.

[8] A. E. Dunlop and B. W. Kernighan. A procedure for placement of standard-cell VLSI circuits. *IEEE Trans. on Computer-Aided Design of Integrated Circuits andSystems*, CAD-4(1):92–98, January 1985.

[9] H. Eisenmann and F. M. Johannes. Generic global placement and floorplanning. In *Proc. Design Automation Conf*, pages 269–274, 1998.

[10] M. ElAbdellaoui. New optimization techniques for VLSI placement. Master's thesis, State University of New York at Binghamton, 2000.

[11] Charles M. Fiduccia and R. M. Mattheyses. A linear-time heuristic for improving network partitions. In *Proceedings of the* 19^{th} *IEEE Design Automation Conference*, pages 175–181, 1982.

[12] B. Hu and M. Marek-Sadowska. Fine granularity clustering for large scale placement problems. In *Proc. Int. Symp. on Physical Design*, pages 67–74, 2003.

[13] G. Karypis, R. Aggarwal, V. Kumar, and S. Shekhar. Multilevel hypergraph partitioning: Application in VLSI domain. In *Proc. Design Automation Conf*, pages 526–529, 1997.

[14] Brian W. Kernighan and S. Lin. An effi cient heuristic procedure for partitioning graphs. *Bell System Technical Journal*, 49:291–307, 1970.

[15] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, May 1983.

[16] J. Kleinhans, G. Sigl, F. Johannes, and K. Antreich. GORDIAN: VLSI placement by quadratic programming and slicing optimization. *IEEE Trans. on Computer-Aided Design of Integrated Circuits andSystems*, 10(3):356–365, 1991.

[17] A. Rohe and U. Brenner. An effective congestion driven placement framework. In *Proc. Int. Symp. on Physical Design*, pages 1–6, 2002.

[18] W. Swartz and C. Sechen. Timing driven placement for large standard cell circuits. In *Proc. Design Automation Conf*, pages 211–215, 1995.

[19] Maogang Wang, Xiaojian Yang, and Majid Sarrafzadeh. Dragon2000: Standard-cell placement tool for large industry circuits. In *Proc. Int. Conf. on Computer Aided Design*, pages 260–263, 2000.