Relaxed Partitioning Balance Constraints in Top-Down Placement*

Andrew E. Caldwell, Andrew B. Kahng and Igor L. Markov

UCLA Computer Science Department, Los Angeles, CA 90095-1596 USA
{caldwell,abk,imarkov}@cs.ucla.edu

Abstract

Recent work of Simon and Teng [17] observes that the recursive *bisection* (i.e., bipartitioning with equal partition target areas, and minimum possible allowed deviation from targets) heuristic for *k*-way minimum-cut graph partitioning can have unbounded error, but that relaxing the balance constraints in each call to the bipartitioning engine can result in *k*-way net cuts within a small ($O(\log k)$) factor of optimal. Motivated by this result, we experimentally determine whether relaxing the traditional exact bisection constraint in a top-down partitioning-based placement tool can improve the resulting cutsizes, and hence total wirelength, of the placement solution. We find that this simple change reduces total wirelength by up to several percent, with no change in placement uniformity and under 10% runtime penalty. Finally, we observe that the stability (predictability) of the placement process appears unimpaired by this modification: both wirelength stability, and stability of Rent parameter based wirelength and wireability estimates, appear to be preserved.

1 Introduction

Global placement of standard-cell VLSI designs seeks non-overlapping placements of cells in cell rows, such that wirelength is minimized while constraints (routability, timing, pre-specified cell locations, cell groupings, etc.) are satisfied. Typically, the wirelength objective is approximated by the sum of net bounding box half-perimeters [19]. For placement runtimes to scale well with netlist size, a top-down partitioning-based (i.e., divide-and-conquer) approach is typically used.

A top-down partitioning-based placer maintains a list of rectangular regions, and lists of cells assigned to each region. Exact bisection [4] or quadrisection [18] is applied recursively to partition a given region and its cells until each region contains exactly one cell. Terminal propagation [6] [18] or more location-sensitive wiring estimators [11] are used to maintain correlation between the minimum-cut partitioning objective and the minimum-wirelength placement objective. The overall approach scales well, and can address variant objective functions as well as various placement constraints, depending on the flexibility of the partitioning engine. The following observations pertain to modern placement approaches, and are directly relevant to our work.

1. Ostensibly to maintain uniform region sizes and region alignments, a top-down placer calls its partitioning engine with *exact balance constraints*, i.e., the partition sizes must be as close to equal as possible.[1]
2. Partitioning solution quality appears to be a major determinant of final placement quality [1] This is reasonable, given the similarity between the minimum net cut partitioning objective and the minimum wirelength placement objective.[2]
3. For both flexibility and solution quality, the state-of-the-art partitioning engine appears to be the *multilevel* implementation of the Fiduccia-Mattheyses heuristic [8] [3] [14].
4. As noted in [13], the top-down placement down to a given level (say, 6 levels of bisection = 64 regions) can be viewed as a multi-way partitioning.[3]
5. The best known approach to multi-way partitioning, embodied in the public-domain hMetis package [14], performs *k*-way balanced hypergraph partitioning by *recursive balanced 2-way partitioning*.

Contributions of This Work

In this work, we seek improvements to the traditional recursive bisection approach to top-down placement. We do not seek a new multi-way placement-specific partitioner, but rather a simple, transparent variation of current practice. There are two motivating observations. First, Simon and Teng [17] have recently analyzed the recursive bisection approach to multi-way graph partitioning. They demonstrate that the approach has unbounded error: families of instances exist for which the optimal *k*-way partition has constant cutsize, but the recursive *optimal* bisection has cutsize proportional to $\Omega(n^2/k^2)$ (dense graphs) or $\Omega(n/k)$ (sparse graphs). On the other hand, Theorems 5.2 and 5.3 of the same work imply that *relaxing the balance constraints* for both the original problem and each bisection allows for provably successful recursive bisection, with at most an $O(\log k)$ factor of suboptimality. Second, we observe that in partitioning a region by a vertical cut there is no need to insist on exact bisection, since the boundary between the two child regions can be shifted to match the partition areas. (Since cell rows are more discrete, this freedom does not apply to horizontal cuts.) Thus, relaxing the bisection constraint is *transparent* to all vertical cuts in the top-down placer.

Given these and the previous observations, it is natural to ask whether relaxing the traditional exact bisection constraint in a top-down placer can yield better net cuts and, per the result of [17], better multi-way cuts. It is also natural to ask whether the result corresponds to improved total wirelength, and how the resulting

* This research was supported by a grant from Cadence Design Systems, Inc.

[1] In most implementations, the target partition sizes are equal to half the total cell area in the instance, and the partition sizes are allowed to deviate from exact equality by at most the size of the largest cell in the instance.

[2] The two objectives are clearly equivalent for "placement of a netlist onto two points". Furthermore, various accounts (e.g., [16] observe that the minimum cut and minimum crossing-count objectives are equivalent to minimum wirelength "in the limit" of closely spaced cutlines.

[3] [13] also notes other aspects of how placers and partitioners "co-evolved". An interesting open issue, which we do not address here, is how the divergence between the (min-cut) partitioning objective and the (min-wirelength) placement objective should be properly addressed in multi-way partitioning.

algorithm complexity is affected (since uneven partition sizes increase the number of levels in the partitioning hierarchy). Finally, we ask whether our variation affects the predictability of the top-down placer. We assess the stability of both the total wirelength results and the Rent parameter [15] of the associated partitioning hierarchies (the latter determines a number of routability and wirelength estimators in the literature).

In the following, Section 2 describes the algorithms and measures (e.g., computation of the Rent parameter of a partitioning hierarchy) that comprise our experimental testbed. Section 3 describes our experimental protocols and results showing that relaxing the bisection constraint reduces total wirelength with negligible implementation cost or runtime penalty, and no effect on placement predictability. To our knowledge, such a study has not been previously reported in the literature. We conclude in Section 4 with other interpretations of experimental data and directions for future work.

2 Experimental Testbed

Our experimental testbed consists of a top-down partitioning-based placer for standard-cell designs, an interface to industry test cases, and code to compute the Rent exponent of a partitioning hierarchy following the method of [9].

2.1 Partitioning-Based Placer

Our placer is based on top-down bipartitioning. Each bipartitioning instance (i.e., placement region with more than a few cells) containing more than one row is partitioned along its longer side. A bipartitioning instance with horizontal cutline has partition target areas as close to equal as possible while respecting the discrete nature of the cell rows; the balance tolerance is the size of the largest cell in the instance. A bipartitioning instance with vertical cutline has exactly equal partition target areas, but we vary the balance tolerance[4] from the minimum possible (i.e., the size of the largest cell in the instance) to 40%.

Within the placer, our multilevel-FM (MLFM) bipartitioning engine uses a multilevel implementation [3] of the Clip-FM bipartitioning algorithm [7]. We use linear-time *heavy-edge matching* (HEM) clustering [14] with matching ratio of $1/6$ in the coarsening and uncoarsening phases of the multilevel partitioner. We stop coarsening when the instance size is 100 clusters or smaller. The HEM clustering is randomized, and the MLFM engine is deterministic (i.e., will always yield the same Clip-FM local minimum given the same instance and initial solution). For a given multilevel HEM clustering hierarchy, our partitioner can operate in a multi-start fashion by maintaining multiple solution paths as it performs the multilevel partitioning. We call the MLFM engine with 10 starts for instances with 200 or more cells, and with 5 starts for instances with fewer than 200 cells; we return the best result over these multistarts. The top-down partitioning continues until each region of the placement contains five or fewer cells.

2.2 Industry Interface

Our placement system reads industry designs in Cadence LEF/DEF format. Three standard-cell test cases in this format, provided by colleagues in industry, have been used in our experiments. Parameters of these netlists are given in Table 1. Each has no cell bigger than 1% of the total area. Area utilization in each of these test cases is sufficiently high that the placement and partitioning heuristics must be closely aware of both area balance and solution quality.

Testcase	Core Cells	Nets	I/O Pads
Case 1	2741	2842	545
Case 2	11471	11673	662
Case 3	20392	25634	185

Table 1: Parameters of standard-cell test cases from industry.

2.3 Computation of the Rent Exponent

Rent's rule is an empirical relation observed in "good" layouts; it reflects a power-law scaling of the number of external terminals of a given subcircuit with the number of modules in the subcircuit. Specifically,

$$T = k \cdot C^p \tag{1}$$

where T is the expected number of external terminals for a given subcircuit or partition; k is a scaling constant equal to the average number of pins per module; C is the number of cells in the given subcircuit or partition; and p is the *Rent parameter*, with $0 \leq p \leq 1$.

During top-down partitioning, for each region we record the number of cells in the region as well as the number of *external nets* (i.e., nets that are connected to at least one cell in the region, and at least one cell outside the region). After appropriate bucketing and averaging, we estimate k and p by applying linear regression to a log-log plot of region size versus number of external nets. For this, we use the Rent parameter computation described in [9].[5] An example Rent parameter fit is shown in Figure 1.

Hagen et al. [9] note that the Rent parameter p characterizes the partitioning algorithm itself (in their work, recursive spectral ratio-cut partitioning was shown to lead to lower Rent parameter values than recursive min-cut Fiduccia-Mattheyses bisection). However, no demonstration of reduced wirelength in top-down placement is attempted in [9]. The Rent parameter has also been extensively studied in the field of wiring estimation, where it affords accurate predictions of the wiring requirements for a given partitioning hierarchy. In particular, given two partitioning trees for the same design, the one with lower Rent parameter will lead to less wirelength and consequently a denser final layout [5] [16]. The Rent parameter is therefore well suited as a quality measure for the complete tree of subcircuits generated by a particular partitioning algorithm.

[4] Following the standard convention in the netlist partitioning literature [2], we say that bisection with, e.g., 10% tolerance permits partition areas ranging from 45% to 55% of the total cell area.

[5] The raw data for the Rent parameter computation consists of ordered pairs of form (number of cells , number of terminals) for regions in the top-down placement. We bucket these (with buckets indexed by k) in either of two ways: (i) according to whether the number of cells in a region (i.e., its size) is between c^k and c^{k+1}, or (ii) according to whether the index of the region (i.e., according to the order in which regions are generated by the placer, which reflects *level* in the placement) is between c^k and c^{k+1}. In our analyses, c values ranging from 1.5 to 2 yielded similar results; the results presented correspond to $c = 1.5$. Within each bucket, both the number of cells and the number of terminals are geometrically averaged (this corresponds to arithmetic averaging in the log-log plot) to yield one averaged data point for the bucket. Linear regression is then used to fit a straight line to the data in the log-log plot.

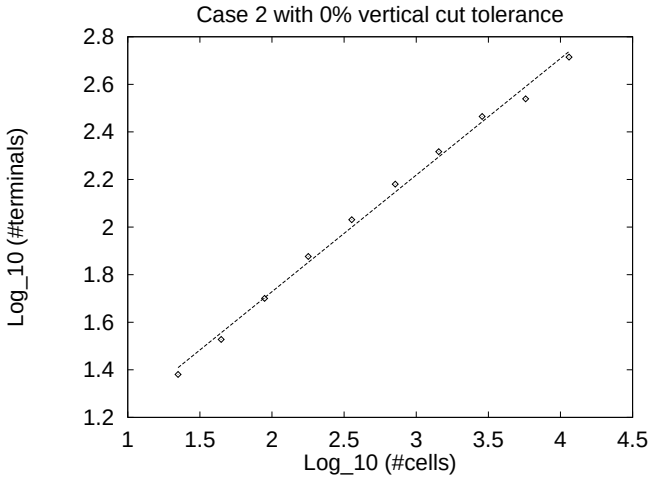

Figure 1: Example Rent parameter fit for Case 2 corresponding to one partitioning hierarchy with vertical cut balance tolerance = 0. All data points are within a few percent of the straight line produced by linear regression.

3 Experimental Protocol and Results

Our experimental protocol is as follows.

- We place each of the three industry test cases with our top-down partitioning-based placer.
- Balance tolerance for vertical cuts is set at 0% (size of the largest cell in the instance), 5%, 10%, 15%, 20%, 25%, 30% or 40%; it is kept constant at 0% (size of largest cell) for horizontal cuts. Near values of balance tolerance that appear most beneficial, a more fine-grain study of the effects of balance tolerance is performed.
- The entire placement process is run 20 times for each value of the balance tolerance.
- Over each set of 20 placement runs, we record minimum, maximum and average values for final wirelength; average total CPU time (seconds on a 140MHz Sun Ultra-1); minimum, maximum and average net-cut in the top-level bipartition; and average value of the fitted Rent parameter of the partitioning hierarchy.[6]

Our experimental results are presented in Table 2. Average and minimum values of total wirelength generally decrease with increasing tolerance, until minimum values are reached for cut tolerances in the range of 15% to 20%. We see that solution quality can improve by up to 3%, while CPU times increase only slightly (never by more than 10%), indicating a favorable tradeoff between solution quality and runtime. Since the implementation cost is negligible, we believe this is a reasonable modification to existing placer implementations.

With regard to the predictability of the placement result, we note that minimum and average wirelength values do not drift apart, i.e., the placer's *stability* (ability to produce predictably good solutions with only a few starts) is unaffected. We also see that the fitted Rent parameter values in the Table are relatively constant, suggesting that Rent-based wireability and wirelength estimates will not change significantly. On the other hand, the sign of changes in Rent parameter seem uncorrelated to that of changes in total wirelength; we comment on this in the next section.

[6] We use the second bucketing approach described in Footnote 5 above, i.e., bucketing by index, with $c = 1.5$.

V-cut tol.	Time Ave	Wirelength Max	Wirelength Min	Wirelength Ave	Top-level net-cut Max	Top-level net-cut Min	Top-level net-cut Ave	Rent par Ave
Case 1, wirelength $\times 10^7$								
0%	137	6.75	6.35	6.57	247	223	240	0.663
5%	151	6.77	6.44	6.60	243	220	232	0.665
10%	150	6.67	6.29	6.44	243	213	226	0.673
15%	151	6.50	6.30	6.42	228	213	222	0.670
17%	154	6.57	6.32	6.39	223	214	219	0.673
19%	150	6.47	6.26	6.37	225	209	216	0.672
20%	151	6.59	6.28	6.39	222	208	214	0.672
21%	151	6.55	6.25	6.39	223	206	214	0.673
23%	150	6.54	6.26	6.40	216	199	206	0.671
25%	152	6.56	6.33	6.42	208	195	203	0.673
30%	153	6.57	6.37	6.47	195	189	192	0.672
40%	157	6.60	6.40	6.48	181	168	175	0.670
Case 2, wirelength $\times 10^8$								
0%	879	3.02	2.93	2.97	335	296	302	0.489
5%	915	3.08	2.99	3.03	310	291	295	0.492
10%	909	2.96	2.89	2.93	304	292	296	0.495
15%	909	2.97	2.87	2.91	306	292	299	0.496
20%	926	2.98	2.86	2.90	303	292	295	0.496
25%	921	2.96	2.87	2.91	312	293	298	0.498
30%	930	2.94	2.88	2.92	312	289	297	0.496
40%	1001	2.99	2.90	2.94	311	291	297	0.497
Case 3, wirelength $\times 10^8$								
0%	2590	6.76	6.41	6.59	383	298	331	0.422
5%	2528	6.70	6.41	6.57	332	287	298	0.420
10%	2567	6.63	6.31	6.42	307	281	296	0.439
15%	2662	6.61	6.28	6.41	316	282	290	0.441
20%	2691	6.61	6.31	6.43	325	263	283	0.438
25%	2714	6.62	6.27	6.43	273	252	258	0.434
30%	2607	6.58	6.28	6.40	260	255	250	0.435
40%	2649	6.58	6.30	6.45	260	253	251	0.449

Table 2: Estimated wirelength, total CPU time, top-level bipartition cutsize, and fitted Rent parameter values for placements with various vertical cut tolerances. Results are averages of 20 runs.

4 Discussion and Ongoing Work

Our studies have identified a very simple, transparent change to existing partitioning-based placers which can reliably reduce the total wirelength of the placement solution. The implementation cost and the runtime overhead of this change are both negligible. Our ongoing research addresses three main questions.

First, we ask whether larger improvements in wirelength can be obtained using relaxed balance constraints. VLSI netlists are likely "well-behaved", and do not resemble the pathological constructions of [17]. Nevertheless, large balance tolerances do allow substantial cutsize reductions.[7] We seek a better understanding of why large

[7] Table 1 shows the monotone reduction in cutsize for the top-level bipartition as

cutsize reductions end up producing only small wirelength reductions. One possible reason is that we have not extended the relaxed balance constraints to horizontal cuts, i.e., every other cut in the placement process remains highly constrained, and prevents "natural" division of the circuit. Thus, we are extending our approach to horizontal cuts as well. Since rows are discrete, one cannot simply relax the tolerance, but must either choose the discrete partition area targets that give the best result (still with zero tolerance) consistent with row structure, or allow non-rectangular regions.[8]

Second, our fitted Rent parameter values for partitioning hierarchies raise interesting questions as to the accuracy of Rent-based wirelength and wireability estimates. We believe that the averaging and bucketing steps in the Rent parameter computation can "mask" the wirelength and cutsize reductions obtained with relaxed-balance partitioning hierarchies. In particular, reducing cutsizes at *all* levels of the partitioning hierarchy – which normally reduces the total wirelength – can easily increase the Rent parameter or leave it unchanged.[9] Hence, we seek Rent parameter calculations and Rent-based wirelength/wireability estimators that capture the effects of area imbalances within the partitioning hierarchy.

Finally, recall from Section 1 that top-down placement down to a given level can be viewed as a multi-way partitioning. Pathological counterexamples notwithstanding, the best known approaches to multi-way partitioning – both for netlist partitioning and for "embedded" partitioning (i.e., placement) – are based on simple recursive 2-way partitioning. With this in mind, our current efforts seek true multi-way placement-specific partitioners that can outperform recursive 2-way partitioning.

References

[1] C. J. Alpert, T. F. Chan, J.-H. Huang, I. L. Markov and K. Yan "Quadratic Placement Revisited", *Proc. ACM/IEEE Design Automation Conference*, 1997, pp. 752-757.

[2] C. J. Alpert and A. B. Kahng, "Recent Directions in Netlist Partitioning: A Survey", *Integration* 19 (1995), pp. 1-81.

[3] C. J. Alpert, J.-H. Huang and A. B. Kahng, "Multilevel Circuit Partitioning", *Proc. ACM/IEEE Design Automation Conference*, 1997, pp. 530-533.

[4] M. A. Breuer, "Min-Cut Placement", *J. Design Automation and Fault-Tolerant Computing* 1(4) (1977), pp. 343-362.

[5] W. E. Donath, "Logic Partitioning", in B. Preas and M. Lorenzetti, eds., *Physical Design Automation in VLSI Systems*, Benjamin/Cummings, 1988.

[6] A. E. Dunlop and B. W. Kernighan, "A Procedure for Placement of Standard Cell VLSI Circuits", *IEEE Transactions on Computer-Aided Design* 4(1) (1985), pp. 92-98.

[7] S. Dutt and W. Deng, "VLSI Circuit Partitioning by Cluster-Removal Using Iterative Improvement Techniques", *Proc. IEEE Intl. Conf. on Computer-Aided Design*, 1996, pp. 194-200.

[8] C. M. Fiduccia and R. M. Mattheyses, "A Linear Time Heuristic for Improving Network Partitions", *Proc. ACM/IEEE Design Automation Conference*, 1982, pp. 175-181.

[9] L. Hagen, A. B. Kahng, F. J. Kurdahi and C. Ramachandran, "On the Intrinsic Rent Parameter and Spectra-Based Partitioning Methodologies", *IEEE Transactions on Computer-Aided Design* 13(1) (1994), pp. 27-37.

[10] `http://www-users.cs.umn.edu/~karypis/metis/`

[11] D. J. Huang and A. B. Kahng, "Partitioning-Based Standard-Cell Global Placement with an Exact Objective", *Proc. ACM/IEEE Intl. Symp. on Physical Design*, 1997, pp. 18-25.

[12] F. R. Johannes, "Tutorial: Partitioning of VLSI Circuits and Systems", *Proc. ACM/IEEE Design Automation Conference*, 1996, pp. 83-87.

[13] A. B. Kahng, "Futures for Partitioning in Physical Design", *Proc. ACM/IEEE Intl. Symp. on Physical Design*, April 1998, pp. 190-193.

[14] G. Karypis, R. Aggarwal, V. Kumar and S. Shekhar, "Multilevel Hypergraph Partitioning: Applications in VLSI Domain", *Proc. ACM/IEEE Design Automation Conference*, 1997, pp. 526-529.

[15] B. Landman and R. Russo, "On a Pin Versus Block Relationship for Partitioning of Logic Graphs", *IEEE Transactions on Computers* C-20(12) (1971), pp. 1469-1479.

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

[17] H. D. Simon and S.-H. Teng, "How Good is Recursive Bisection?", *SIAM J. Scientific Computing* 18(5) (1997), pp. 1436-1445.

[18] P. R. Suaris and G. Kedem, "Quadrisection: A New Approach to Standard Cell Layout," *Proc. IEEE/ACM International Conference on Computer-Aided Design*, 1987, pp. 474-477.

[19] W. Swartz and C. Sechen. "Timing Driven Placement for Large Standard Cell Circuits", *Proc. ACM/IEEE Design Automation Conference*, 1995, pp. 211-215.

the balance constraint is relaxed. Similar reductions occur at succeeding levels of the partitioning, e.g., changing the balance tolerance from 0% to 40% reduces the *average* 4- and 8-way cutsize by 17% and 14% for the Case 1 example, by 9% and 11% for the Case 2 example, and by 3% and 5% for the Case 3 example.

[8] Other possible reasons for the small wirelength reductions include the mismatch between the net cut and wirelength objectives [13].

[9] Consider the example Rent parameter fit in Figure 1. If cutsizes are reduced while partition sizes become unbalanced and geometric averaging is applied, the averaged data points will shift "down" (reduced cutsizes) and "to the left" (geometric mean is not greater than arithmetic mean). How this affects the slope of the best-fit line is not easily predicted.