FRACTIONAL CUT: IMPROVED RECURSIVE BISECTION PLACEMENT

Ameya Agnihotri Mehmet Can YILDIZ Ateen Khatkhate Ajita Mathur Satoshi Ono Patrick H. Madden

SUNY Binghamton Computer Science Department
http://vlsicad.cs.binghamton.edu

ABSTRACT

In this paper, we present improvements to recursive bisection based placement. In contrast to prior work, our horizontal cut lines are not restricted to row boundaries; this avoids a "narrow region" problem. To support these new cut line positions, a dynamic programming based legalization algorithm has been developed. The combination of these has improved the stability and lowered the wire lengths produced by our *Feng Shui* placement tool.

On benchmarks derived from industry partitioning examples, our results are close to those of the annealing based tool *Dragon*, while taking only a fraction of the run time. On synthetic benchmarks, our wire lengths are nearly 23% better than those of *Dragon*. For both benchmark suites, our results are substantially better than those of the recursive bisection based tool *Capo* and the analytic placement tool *Kraftwerk*.

1. INTRODUCTION

Advancing fabrication processes has enabled an explosion in the number of logic gates in "typical" designs, resulting in an industry need for placement methods that can handle millions of movable objects. At the same time, design cycle times have shrunk; time-to-market pressure has put a premium on speed for design tools.

While increased capacity and speed are desired, we also wish to avoid sacrificing quality. The work presented improves the quality of results obtained by a fast placement approach. We use a traditional formulation, but relax one common assumption.

Our primary contributions are the development of a *fractional cut* approach, and a complementary legalization algorithm. In recursive bisection, horizontal cut lines are normally aligned with cell row boundaries–this places a number of constraints on the partitioning engine. Legalization for bisection based approaches is usually trivial; new horizontal cut positions require a more sophisticated approach. We take advantage of the uniform distribution of cell area that results from our bisection based approach, and develop a legalization method that is efficient and effective.

The remainder of this paper is organized as follows. We first briefly survey prior placement methods. We next describe our fractional cut approach, and provide an explanation of how we legalize placements. Experimental results show the dramatic improvement in wire length our methods have produced. We conclude the paper with a summary of current and future work.

2. PRIOR WORK

Placement is a well studied area of physical design. We focus on standard cell placement here, and briefly survey the three major approaches to the problem–simulated annealing, analytic placement, and partitioning based placement.

ICCAD'03, November 11-13, 2003, San Jose, California, USA.

Simulated annealing[12] is well known to produce excellent results, but with high run times. This approach starts with an initial placement and "temperature." Modifications to the placement are randomly generated, with move acceptance guided by a probabilistic function. *Timberwolf*[15] is one well-known annealing tool; given sufficient time, it produces excellent results. *Dragon*[16] is a hybrid that combines recursive bisection with annealing. The *hMetis*[11] partitioner first divides the cells into a set of *bins*; an annealing process is then used to move cells between bins to optimize a wire length objective.

Analytic placement algorithms transform the placement problem into systems of equations; solving these equations results in a global optimum. This solution, however, may contain a large number of overlapping cells. A legalization step is required to move cells to rows and to remove overlap. Well known analytic placement tools include *GORDIAN*[13], *Kraftwerk*[10] (sometimes known as "Plato" or "FD-98"), and *mPL*[6].

Most partitioning based placement methods follow the approach of Dunlop and Kernighan[9]. Current partitioning based tools include *Capo*[5] and *Feng Shui*[18]. Both use multi-level min-cut hypergraph partitioning methods. *Capo* uses *MLPart*[3] where as *Feng Shui* uses *hMetis* and a *k*-way algorithm based on iterative deletion. Both tools use local branch-and-bound optimization.

3. IMPROVED RECURSIVE BISECTION

An overview of our placement tool, *Feng Shui 2.0*, is shown in Figure 1. In most respects, our approach is similar to traditional recursive bisection placement tools. Starting from an initial net list and placement area, a partitioning algorithm is used to divide the net list, while cut lines are used to split the placement area. The direction of cut lines in our approach is determined by an aspect ratio parameter[18]. In addition to traditional partitioning with *hMetis*, we also use large-scale *k*-way partitioning by iterative deletion[17] to obtain initial terminal propagation information.

3.1. Fractional Cuts

Traditionally, horizontal cuts are made along cell row boundaries; in [4], the authors note "a straight-line cut perpendicular to rows can take a much larger set of locations, *while straight-line cuts parallel to rows can effectively be only between rows.*" We challenge this common assumption in this paper.

The motivation for restricting horizontal cuts to row boundaries is obvious: row assignments for each cell can be determined easily. Relative horizontal positions of cells can be found by simply comparing the X coordinates of the cells in any given row. While this restriction simplifies many things, it introduced a "narrow region problem" that was a source of instability for an earlier

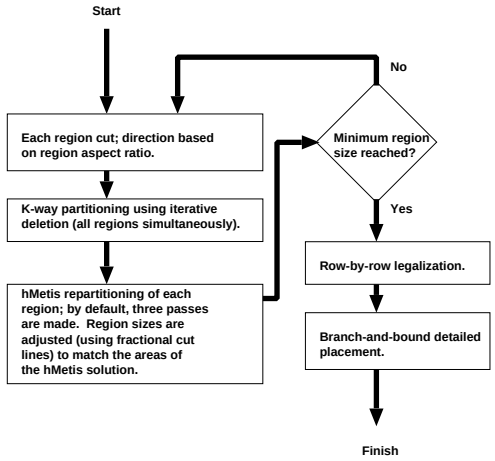

Figure 1: Flow chart for *Feng Shui 2.0*.

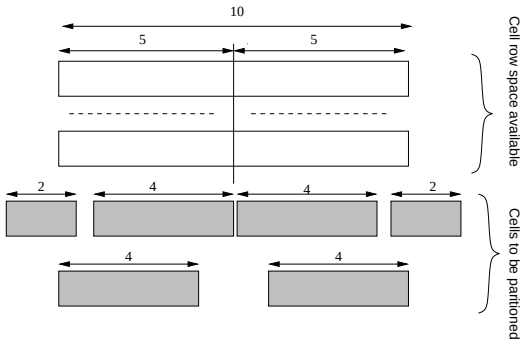

Figure 2: In recursive bisection, if regions become tall and narrow, horizontal partitioning may be difficult; it may not be possible to split cells into rows while obeying row area constraints, or feasible solutions may have high numbers of cut nets.

version of our tool, *Feng Shui 1.5*.

Figure 2 illustrates the problem: we have two regions that each span two cell rows. The width of each region is 5 units; the cells to be assigned to each region have widths of 4, 4, and 2 respectively. If we cut the regions horizontally, no partitioning solution exists that meets the area constraints of the rows. In the earlier versions of our placement tool, the narrow region problem could result in unbalanced partitioning results, and also unbalanced row lengths.

To avoid this problem, the placement tool *Capo* partitions horizontally if there are fewer than 15 cells per row in a region. *Capo* also uses the additional white space found in most designs to increase partitioning tolerance, improving the chance that an acceptable solution exists. The *aspect ratio* used in *Feng Shui 1.5*, along with the cell widths and spacings found in the MCNC benchmarks, allowed the problem to be avoided. For more recent benchmarks, the simple "row packing" approach used by default in *Feng Shui 1.5* produces pathological behavior, with row lengths varying substantially. Between versions 1.5 and 2.0, many legalization methods were explored; we present the most effective method here.

This problem motivated our consideration of alternate horizontal cut positions. Allowing a region to occupy a fraction of a row (a "fractional cut") provided greater tolerance in partitioning.

3.2. Legalization

Obviously, if our bisection process does not align cuts with row boundaries, it cannot be used to assign cells to rows. To obtain a legal placement, cells must be shifted to cell rows, and overlaps must be removed. The approach described here introduces relatively small amounts of movement per cell; the cells are uniformly distributed across the placement area, ensuring that there is sufficient space nearby for legalization. The motion of cells from non-legal positions to legal positions is shown in Figure 3. Our approach operates on a row-by-row basis, and is as follows.

- We use the center of each region as an initial "preferred" coordinate for each cell. All cells are sorted by their preferred Y coordinate; legalization of cells is done starting with the top-most row. A set of *candidate* cells (those with the highest Y coordinates) is considered for each row; the total area of the candidate cells exceeds the desired row capacity.
- A subset of the candidate cells is selected for assignment to a row such that we minimize the assignment cost for the row. If a cell is assigned to the row, the cost is the square of the distance between the non-legal and legal positions. If a cell is deferred to a subsequent row, the cost is the square of the distance to the next row. Deferred candidate cells become candidates for subsequent rows; processing continues until all cells have been assigned to a row.
- After row assignment, the cells in each row are sorted by their preferred X coordinate, and are packed without additional space from left to right.

While it might appear that the selection of cells would be computationally intractable, our method is in fact quite efficient. Candidate cells are first sorted by increasing X coordinate, and the order of cells assigned in a row is constrained by this sequence. We refer to the candidate cells as $C_0, C_1, ...C_n$, and use the notation $< \{\text{assigned}\}, \{\text{deferred}\} >$ to indicate the sets of cells that are either assigned to the row in question, or deferred for later consideration. For simplicity, assume that the cells are of uniform width.

Initially, we have $< \{\}, \{\} >$, or no cells assigned or deferred. By considering cell C_0, we obtain two possible partial solutions, $< \{C_0\}, \{\} >$, and $< \{\}, \{C_0\} >$ (each with different costs). When considering cell C_1, we obtain new partial solutions $< \{C_0C_1\}, \{\} >$, $< \{C_0\}, \{C_1\} >$, $< \{C_1\}, \{C_0\} >$, and $< \{\}, \{C_0C_1\} >$. A brute-force approach to the problem would generate an exponential number of partial solutions; observe, however, that $< \{C_0\}, \{C_1\} >$ and $< \{C_1\}, \{C_0\} >$ both consume the same amount of space in the row. As assignment cost is dependent only on the distance between legal and non-legal postions, and the "right boundary" of the two partial solutions is identical, the optimal solution for the remainder of the cells is independent of which of C_0 or C_1 is selected.

Elimination of partial solutions with high cost reduces the number of cases that must be considered. In most designs, there are a small number of legal cell positions. If there are p possible cell locations, and c cells considered for assignment, dynamic programming provides a solution to the problem in $O(p \times c)$. Under the constraint that the order of the cells is not changed, this solution is optimal. We would recommend [8] as a good reference for dynamic programming methods.

After legalization, we apply branch-and-bound reordering on both single and multiple rows. *Feng Shui 2.0* also supports cell mirroring and space insertion; both are essential for improving routability. Due to space constraints, we do not discuss these here.

Bench mark	Feng Shui 2.0		Feng Shui 1.5+DP			Feng Shui 1.5			Capo 8.6			Dragon 2.23			Kraftwerk (not legal)		mPL 2.0	
	wl	RT	wl	×	RT	wl	×	RT	wl	×	RT	wl	×	RT	wl	×	wl	×
ibm01	0.52	154	0.54	1.053	153	1.85	3.580	147	0.57	1.103	47	0.51	0.988	686	0.70	1.359	0.64	1.238
ibm02	1.50	297	1.51	1.009	292	5.63	3.754	291	1.60	1.067	109	1.44	0.962	1077	2.15	1.433	1.61	1.073
ibm07	3.30	760	3.36	1.020	753	16.72	5.073	713	3.71	1.125	292	3.31	1.003	1732	5.12	1.553	4.07	1.235
ibm08	3.60	933	3.72	1.033	918	13.66	3.796	864	3.89	1.082	314	3.39	0.943	4357	4.66	1.295	4.25	1.181
ibm09	3.02	872	3.22	1.065	857	16.95	5.603	807	3.31	1.095	316	2.96	0.979	3278	4.26	1.408	3.81	1.260
ibm10	5.66	1266	5.88	1.037	1249	40.00	7.063	1209	6.34	1.119	475	5.61	0.991	5392	7.61	1.344	6.61	1.167
ibm11	4.48	1231	4.80	1.073	1220	31.73	7.091	1123	4.89	1.093	454	4.43	0.991	3523	5.80	1.296	5.96	1.332
ibm12	7.74	1379	8.08	1.044	1340	42.85	5.533	1268	8.76	1.132	533	7.60	0.982	6133	10.41	1.344	9.44	1.219
Avg				1.042			5.187			1.102			0.980			1.379		1.213

Table 1: 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. 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*.

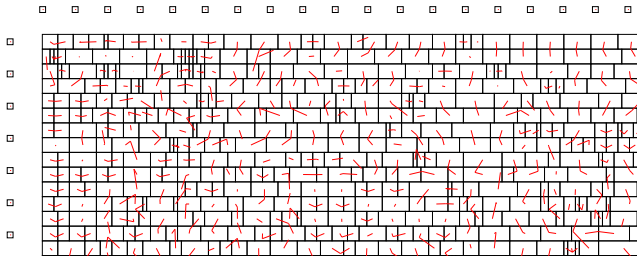

Figure 3: The legalization process moves cells from the centers of their regions to legal locations within a cell row. Our dynamic programming formulation finds solutions which move cells a small distance; the lighter shaded lines show the direction of travel for each cell for this portion of the benchmark IBM01.

4. EXPERIMENTAL RESULTS

Benchmarking and reporting of results for placement has been problematic; many previously reported results cannot be replicated, or contain skewed comparisons[14, 1].

In this paper, we use widely available benchmarks; wire lengths measured by our tools match those calculated by tools developed at the University of Michigan. The benchmark sets we consider here are "IBM Version 2 easy"[16] and "PEKO"[7].

The placement tools considered are our *Feng Shui 2.0*, *Capo 8.6*[4], *Dragon 2.23*[16], *Kraftwerk*[10], and *mPL 2.0*[6]. To evaluate the impact of fractional cut lines on wire length, we use *Feng Shui 1.5* to produce placements where cells are row-aligned, but cell overlaps have not been removed; we then perform legalization with our new dynamic programming method, and refer to this as *Feng Shui 1.5 + DP*. We also include results from *Feng Shui 1.5* using the simplest legalization option, sorting cells in a row by their X coordinate. While this approach worked well for the cell sizes and spacings found with *MCNC* benchmarks, the performance here is poor. We use this configuration to emphasize that run times are not dominated by the dynamic programming legalization step.

For these experiments, the optimization objective is half perimeter wire length minimization; the placement tools are run in default configurations, and without any sort of timing, congestion, or routability optimization. Run times listed are in seconds; we only list times for experiments that were performed on our research machines, identical Dell 340 PCs with 2.4ghz Pentium IV processors, running a Debian release of Linux.

4.1. IBM Benchmarks

A subset of the ISPD98[2] partitioning benchmarks have been adapted for use in placement by the authors of *Dragon*[16]; vertices are mapped to a commercial standard cell library such that vertex weights correspond with cell area, and cell pin counts match vertex degree.

Average results over five runs are shown in Table 1. We include the ratio of wire lengths and run times to aid in comparison. The use of fractional cut lines provides an improvement of roughly 4%; the use of dynamic programming legalization avoids pathological behavior. Estimated wire lengths from all three versions of *Feng Shui* are nearly identical until the last stages of bisection, and even then, there are only modest differences. Run times are shown only for *Feng Shui*, *Capo*, and *Dragon*; experimental results for the other tools were provided by Prof. Igor Markov.

Feng Shui 2.0 obtains wire lengths that are on average 10% better than *Capo 8.6*; we note that *Capo* is significantly faster. *Dragon 2.23* outperforms our tool on most benchmarks, but on average, the wire lengths are less than 3% better, while run times are three to four times higher. Our new tool also outperforms *Kraftwerk* and *mPL 2.0* by a wide margin.

4.2. PEKO Benchmarks

The PEKO[7] ("Placement Examples with Known Optimal") benchmarks are synthetic; net degree parameters are extracted from real circuits, which can then be used to create arbitrarily large placement problems that are statistically similar to real circuits. The circuits were constructed with a known optimal configuration.

We add our results to those reported in [1] in Table 2. Note that in this table, *Dragon 2.20* was used; the more recent *2.23* version improves on these results. On average, our placement tool outperforms all other tools except for *mPL*. Again, the cell shape and row spacing cause pathological behavior in *Feng Shui 1.5*, when using the simplest legalization procedure.

Bench mark	Known Optimal wl	Feng Shui 2.0 wl	×	Feng Shui 1.5+DP wl	×	Feng Shui 1.5 wl	×	Capo 8.6 wl	×	Dragon 2.20 wl	×	Kraftwerk (not legal) wl	×	Kraftwerk +DOMINO wl	×	mPL 2.0 wl	×
Peko01	0.81	1.25	1.55	1.30	1.60	4.59	5.67	1.29	1.59	1.46	1.80	1.39	1.72	1.74	2.15	1.10	1.36
Peko02	1.26	2.09	1.66	2.10	1.66	6.92	5.50	2.03	1.61	2.43	1.93	1.98	1.57	2.61	2.07	1.76	1.40
Peko03	1.50	2.62	1.75	2.61	1.74	8.18	5.45	2.66	1.77	2.93	1.95	3.02	2.01	3.78	2.52	2.05	1.37
Peko04	1.75	2.82	1.61	2.91	1.66	14.31	8.18	3.12	1.78	3.87	2.21	3.25	1.86	4.25	2.43	2.31	1.32
Peko05	1.91	3.01	1.58	3.14	1.65	15.70	8.22	3.16	1.65	3.79	1.98	3.92	2.05	4.79	2.51	2.57	1.35
Peko06	2.06	3.44	1.67	3.42	1.66	21.38	10.38	3.57	1.73	4.35	2.11	4.07	1.98	5.38	2.61	2.78	1.35
Peko07	2.88	4.91	1.71	4.98	1.73	38.54	13.38	5.07	1.76	6.24	2.17	5.73	1.99	7.56	2.62	3.95	1.37
Peko08	3.14	5.25	1.67	5.62	1.79	34.16	10.88	5.57	1.77	6.79	2.16	5.87	1.87	8.17	2.60	4.99	1.59
Peko09	3.64	5.98	1.64	6.32	1.74	36.03	9.90	6.47	1.78	7.72	2.12	8.52	2.34	10.00	2.75	4.76	1.31
Peko10	3.73	7.87	2.11	8.40	2.25	17.89	4.80	8.00	2.14	8.49	2.28	8.90	2.39	12.00	3.22	6.59	1.77
Avg			1.69		1.75		8.23		1.76		2.07		1.98		2.55		1.42

Table 2: 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.

5. CONCLUSION AND FUTURE WORK

In this paper we have introduced a "fractional cut" approach and a complementary legalization algorithm. Allowing arbitrary positions for horizontal cuts has improved wire lengths by 4%.

Our placement tool obtains half perimeter wire lengths 10% percent better than *Capo*, and comparable to the annealing based tool *Dragon* on the IBM benchmark circuits. Wire lengths for the synthetically generated PEKO benchmarks are on average better than *Dragon 2.20*, *Kraftwerk*, *Capo 8.6*; only *mPL* has better wire lengths than our tool.

Many techniques presented here may be useful to other tools; *Capo* and *Dragon* might benefit by adopting our cut line strategy, while *Kraftwerk* might benefit from our legalization algorithm. Our current work is to further improve the wire lengths and the run times of our tool; we are also actively working on circuit routing and on methods to optimize routability during placement. *Feng Shui 2.0* is freely available through our research group web site, and also through the GSRC Bookshelf (http://www.gigascale.org).

Acknowledgements: this work was supported in part by SRC project 947.1, an IBM Faculty Partnership Award, an equiment grant from Intel, and the New York State Microelectronics Design Center. Discussions with Prof. Roman Bazylevych and Dr. Bill Swartz spurred much of this research. We would like to thank our colleagues at IBM T. J. Watson and Austin Research Labs, and also Prof. Igor Markov of U. Michigan for useful comments and assistance. We used Prof. Markov's Bookshelf.EXE tools for some experimental results.

6. REFERENCES

[1] S. N. Adya, M. C. Yildiz, I. L. Markov, P. G. Villarrubia, P. N. Parakh, and P. H. Madden. Benchmarking for large-scale placement and beyond. In *Proc. Int. Symp. on Physical Design*, pages 95–103, 2003.

[2] C. J. Alpert. The ispd98 circuit benchmark suite. In *Proc. Int. Symp. on Physical Design*, pages 80–85, 1998.

[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] Andrew E. Caldwell, Andrew B. Kahng, and Igor L. Markov. Optimal partitioners and end-case placers for standard-cell layout. *IEEE Trans. on Computer-Aided Design of Integrated Circuits andSystems*, 19(11):1304–1313, 2000.

[6] T. Chan, J. Cong, T. Kong, and J. Shinnerl. Multilevel optimization for large-scale circuit placement. In *Proc. Int. Conf. on Computer Aided Design*, pages 171–176, 2000.

[7] 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.

[8] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. MIT Press, 1990.

[9] 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.

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

[11] 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.

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

[13] 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.

[14] Patrick H. Madden. Reporting of standard cell placement results. *IEEE Trans. on Computer-Aided Design of Integrated Circuits andSystems*, 21(2):240–247, February 2002.

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

[16] Xiaojian Yang, Bo-Kyung Choi, and Majid Sarrafzadeh. Routability driven white space allocation for fixed-die standard cell placement. In *Proc. Int. Symp. on Physical Design*, pages 42–50, 2002.

[17] M. C. YILDIZ and P. H. Madden. Global objectives for standard cell placement. In *Proc. Great Lakes Symposium on VLSI*, pages 68–72, 2001.

[18] Mehmet Can YILDIZ and Patrick H. Madden. Improved cut sequences for partitioning based placement. In *Proc. Design Automation Conf*, pages 776–779, 2001.