

On Structure and Suboptimality in Placement

Satoshi Ono^{1,2,3} Patrick H. Madden^{2,3}
FAIS¹ University of Kitakyushu² SUNY Binghamton CSD³

Abstract—Regular structures are present in many types of circuits. If this structure can be identified and utilized, performance can be improved dramatically. In this paper, we present a novel placement approach that successfully identifies regularity, and obtains placements that are superior to other “general purpose” methods. This method has been integrated into our *Feng Shui 2.6* bisection-based placement tool.

On experiments with the PEKO benchmarks, our results are within 32% of optimal for both the large and small suites. The largest example, with 2.1 million cells, can be completed in sixteen hours. The majority of our run time is during detail placement—global placement takes under three hours. The success of our method shows that it can find structure, even when the structure was not expected or intended.

As part of this work, we have made a number of observations related to the nature of suboptimality in placement. These observations have shown that some neglected research areas have great potential, while problems that receive considerable attention are essentially adequately solved.

I. INTRODUCTION

There has been rising interest in placement algorithms; with technology scaling, the importance of interconnect length on performance continues to grow. Where in prior technology generations, there was only a passing interest in wire length minimization, this is now a key concern. Device scaling alone cannot achieve the desired performance improvements: fundamental advances in placement are needed, to reduce wire lengths, thereby reducing delay and power consumption.

Recently, a set of synthetic benchmarks were used to evaluate how far placement tools are from optimal, when pursuing a wirelength minimization objective[5]. For the majority of tools tested, placements were significantly suboptimal: wire lengths were from 50% to 150% higher than optimal, and the quality generally degraded as circuit sizes increased. An earlier set of experiments on a datapath design[8] showed similar behavior; the design that was produced manually (by a designer who understood the structure of the circuit) was far superior to the result produced by commercial tools.

The results reported in [8] motivated much of the work we present here. If structure is present in a circuit, finding the structure and then utilizing it would obviously result in a far better placement. Thus, our focus is not on a general-purpose placement method, but on one tuned for designs with internal structure. Our method can in fact find structure where no structure was intended or expected: for the PEKO benchmarks, we find structure, and can obtain placements that are within 32% of optimal for both the “small” and “large” suites. Our method is also remarkably fast: global placement of a design with 2.1 million cells is done in two and a half hours, while the complete placement run takes roughly sixteen hours.

The algorithmic complexity of our approach is $O(n \log n)$; it is unlikely that any approach will be able to obtain better asymptotic complexity while still producing a good quality placement.

The development of our approach has also resulted in a number of observations regarding the nature of optimal placements for non-synthetic (and non-structured) designs. The PEKO benchmarks, while having a number of aspects that result in their being fundamentally different from “real designs,” have proved useful in understanding how placement methods fail to find good solutions. We also point out a key problem in reaching the next level of performance: rather than improvements in global placement, we see the next barrier being *detail placement*. We also make observations related to architecture-level decisions.

The remainder of this paper is organized as follows. We first discuss more traditional placement methods, and the differences between structured and non-structured designs. We then present our new approach; we assume that a structure is present in the design, and then seek to find and exploit it. Section IV presents our experimental results, with a focus on the PEKO benchmarks. Section V presents a number of observations, and a consideration of the quality difference between *global* placement and *detail* placement. We conclude the paper in Section VI.

II. PREVIOUS WORK

Circuit placement has been widely studied, and is one of the classic problems in VLSI design. Popular methods for placement include annealing[16], [20], [22], analytic methods[17], [9], [14], [21], and recursive bisection[4], [15].

In each of these methods, the locations of circuit elements are improved, to address a wire length (or weighted wire length) minimization objective. The size of modern placement problems makes the task difficult: with millions of components and wires, finding good solutions is non-trivial.

The problem is frequently divided into “global placement” and “detail placement” stages. During global placement, rough positions are found for the components, and there may be a great deal of overlap. The objective during global placement is to position the components close to good locations; by keeping the positioning rough, we reduce computation time considerably. Additionally, the circuit can be clustered, reducing the size and complexity of the placement problem.

During “detail placement,” overlaps are removed with legalization techniques. Many of these techniques are based on dynamic programming (for example, [2]), although a simple greedy method[10] has been shown to be quite effective.

Following legalization, many placement tools apply local optimization[3] to improve results.

An implicit assumption of traditional placement is that the circuit net list is essentially a random graph, onto which order must be applied. For many useful circuits, however, the arrangement of components more closely resembles a mesh: both datapath designs and memories have clear “good” placements, which general purpose tools have difficulty in finding[8].

III. A NOVEL PLACEMENT ALGORITHM

The placement method we describe was developed to find structure when structure is present. We will use “placement distance” to mean the Manhattan distance between two cells, while “graph distance” refers to the shortest path between the cells in the weighted hypergraph netlist. A fundamental assumption of our placement approach is that there is a relationship between the placement distance between a pair of cells, and the graph distance between them.

In Figure 1, we show a graph of the physical distance between a pair of randomly chosen cells in a net list, versus the length of the shortest path between them. We use the circuit PEKO01 and the placement result of mPL[5] for this figure. There is a clear correlation between physical distance (in a good quality placement) and shortest path distance.

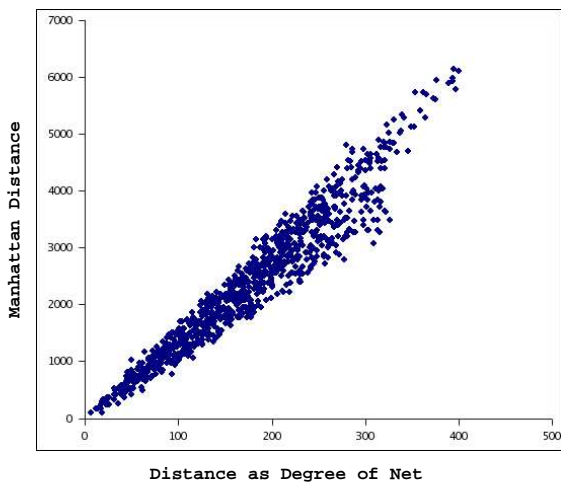


Fig. 1. Manhattan distance in a good quality placement vs. shortest path distance in a weighted hypergraph.

Our tool performs placement in three steps: we first utilize graph distances to obtain relative cell positions, then even out placement density with cell shifting[21], and finally perform legalization and detail placement.

For determining distances in the graph, we weight the hypergraph in the following way: low cardinality nets have low lengths, while high cardinality nets have high length. An easy way to think of this is that lower degree nets should connect cells closer together than higher degree nets do. For example, cells connected by a two-pin net are likely best placed next to each other, while a higher degree net requires more placement area (placement distance) to put all cells connected by the net.

We have experimented with a variety of different weighting functions, and discuss this briefly in the experimental results section.

For circuits with regular structures, the relationship between net degree and distance in a good placement is strong. In a mesh-like topology, many nets are two-pin; adjacent elements of the mesh are connected, with these two pin-nets being quite short. In a datapath circuit, bit lines and control lines are high-fanout, but span the height or width of the block. As we shall show in our experiments, there is even structure in unintended places: the synthetic PEKO benchmarks, widely used to measure the suboptimality of placement approaches, contain a great deal of structure that our approach can detect and utilize.

Our approach uses the relationship between physical and graph distance in the following way. If we can assume that the distance between a pair of cells in a good placement is related to the degree of a net that connects them, we can use this information to weight the net in a shortest-path computation. Using these weights, we can tell the maximum placement distance of *any* two cells. If one of the cells is fixed in place, we can determine the placement area where the other cell could be placed. Having more cells fixed, we have smaller area where the movable cells could be placed.

Buffer insertion and clock tree generation algorithms (for example, [13], [7]) use similar principles: given the location of a buffer or clock driver, there is a “feasible region” in which another buffer or driver might be placed. The intersections of these regions are used to determine locations for other elements. The principle is also similar to techniques used in global positioning and radar systems: the location an object can be determined based on distances from reference points. We refer to our placement tool as “beacon,” as much of our placement approach is modeled around a GPS system—satellites act as signal beacons, providing reference points for triangulation.

Steps in our approach are the following.

- We identify good candidates for “corner” cells, and fix them in place.
- We determine the distance of every other cell from the placed corners, using the relationship between placement distance and graph distance.
- From each corner, the possible location for the placement of a cell is a Manhattan circle (diamond). The intersections of these circles for a given cell determine the possible locations for placement.
- Each cell is placed in the center of its corresponding Manhattan circle; we then apply cell shifting, traditional legalization, and detail placement.

Corner Finding

When pads are present, finding a “corner” is trivial; for benchmarks which do not have pads, we perform the following steps.

We first pick one cell randomly, then run the Dijkstra’s algorithm with the net weights to find the farthest cell from the random chosen cell. We assume the farthest cell, A , is one

of the corners. Next, we run Dijkstra’s algorithm starting from cell A , and we set the farthest cell, B , as the diagonal corner. Using the same technique, but with cells A and B as sources, we identify the third corner, C . With the source cell as C , we find the fourth corner, D . Depending on the ratio A to C and A to D , we chose a configuration that gives acceptable distance ratios, and have determined four “corner” cells. The locations of these cells are fixed at the corners of the placement region, and these four become the “beacons” that are used in the rest of our placement approach.

Triangulation

Using the graph distance from four corners, we place the remainder of the cells. We run Dijkstra’s algorithm four times—once from each of the four corners. Dijkstra’s algorithm gives distances to *all* cells—for each cell, we have four distances, and we use these to triangulate the possible locations for the cell. Again, the principle is similar to methods commonly in use with buffer insertion and clock tree generation.

Cell Shifting

In Figure 2, we show the shape of the placement prior to legalization. As graph distance is not *exactly* equal to physical distance, there can be overlap, resulting in higher density in the center of the placement region.

To remove overlap, we apply the “cell shifting” technique presented in [21]. The placement area is divided into horizontal or vertical stripes, and then by applying iterative bin adjustments, the cells are evenly distributed across the placement region, while relative positions are kept in tact. Figure 2 also shows wire length changes during placement shifting by color coding the cells.

The color coding uses the following scheme: as the PEKO benchmarks have known optimal configurations, we know the total length of nets connected to any given cell in an optimal placement. We compare the net lengths in this ideal case to the lengths for each cell in our placement: higher lengths are shaded a darker color, allowing easy visualization of placement quality. From the figure, it is easy to see that prior to cell shifting, net lengths are quite close to their optimal values. After shifting, lengths increase slightly, but are still close to the desired values.

Legalization and Detailed Placement

Our approach is integrated with our *Feng Shui* 2.6[2] tool, and we use this to perform legalization and detail placement. Version 2.6 provides a simple greedy heuristic based on [10], and branch-and-bound driven detail placement. In our experiments, we include run times for both the “beacon” portion of a run, and the time consumed by detail placement. Run times are in fact dominated by detail placement, with global placement contributing only around 25% of the total run time.

In the experiments we report here, the detail placement window uses a size of 4; increasing the size of the window improves results slightly, with an increase in run time.

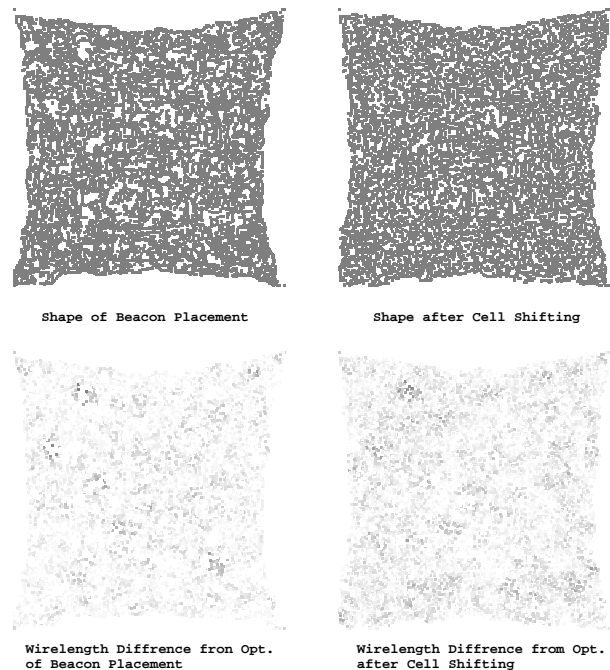


Fig. 2. Cell shifting, evening cell density across the placement region. Cells are color coded based on the lengths of connected nets.

IV. EXPERIMENTAL RESULTS

For our experiments, we have tried a variety of net weighting schemes. We find that for designs with regularity, an extremely simple method works quite well: we equate hypergraph edge length to the cardinality of a net minus 1 (scaled for cell size). Thus, a two pin net has a length of “1”, a three pin net “2,” and so on.

To evaluate the performance of our placement approach, we first applied it to regular mesh structures. Not surprisingly, results were quite good. For a more interesting and challenging test, we next considered the PEKO benchmark circuits[5]; these have been widely used to evaluate placement performance, have known optimal configurations, and are large enough to test the scalability of placement tools.

In Tables I and II, we show the run times of different portions of our placement approach on PEKO suite 1 and 2. The largest design contains roughly 2.1 million cells, and can be completed on a 2.4ghz Pentium-based PC running Linux, in roughly sixteen hours. The bulk of the run time is consumed by detail placement.

In Table III, we compare quality ratios of a number of current academic and commercial tools on both PEKO suite 1 and 2. We use some results reported in [5]—many of the tools have new versions, with improved results. Most placement tools are at least 50% away from optimal in terms of wire length, with quality degrading as the problem sizes increase. Run time and memory become an issue with the larger benchmarks. Direct run time comparisons are not possible, due to differences in the computing platforms.

Our tool is able to complete all designs in less than 24 hours.

circuit	Total runtime(s)	Beacon %	Cell Shifting %	Rest %
Peko01	29	1.62	24.21	74.17
Peko02	53	1.62	27.19	71.19
Peko03	65	1.57	26.22	72.22
Peko04	80	1.50	25.45	73.05
Peko05	85	1.60	25.12	73.28
Peko06	105	1.63	29.30	69.08
Peko07	169	1.30	28.05	70.65
Peko08	197	1.45	26.37	72.18
Peko09	201	1.47	26.83	71.70
Peko10	298	1.40	29.00	69.60
Peko11	301	1.30	29.44	69.27
Peko12	311	1.44	28.58	69.97
Peko13	294	1.79	37.74	60.47
Peko14	874	1.10	29.07	69.83
Peko15	1062	1.21	26.26	72.53
Peko16	1236	1.15	27.32	71.53
Peko17	1281	1.18	26.88	71.94
Peko18	1481	1.06	26.40	72.53
Avg.		1.41	27.75	70.84

TABLE I
RUN TIME OF BEACON ON PEKO SUITE1.

circuit	Total runtime(s)	Beacon %	Cell Shifting %	Rest %
Peko01x10	699	1.23	26.83	71.94
Peko02x10	1308	1.15	26.62	72.22
Peko03x10	1671	1.11	25.45	73.45
Peko04x10	2260	0.91	27.72	71.37
Peko05x10	2254	1.05	26.69	72.26
Peko06x10	2695	0.92	25.63	73.45
Peko07x10	4687	0.79	23.18	76.04
Peko08x10	5693	0.78	24.27	74.95
Peko09x10	6134	0.76	24.09	75.15
Peko10x10	9171	0.70	21.94	77.37
Peko11x10	9690	0.43	21.95	77.63
Peko12x10	9821	0.70	21.67	77.63
Peko13x10	12901	0.62	20.21	79.17
Peko14x10	34178	0.40	17.21	82.39
Peko15x10	39106	0.44	16.24	83.31
Peko16x10	49161	0.40	15.52	84.08
Peko17x10	48528	0.42	15.61	83.97
Peko18x10	60373	0.35	16.10	83.55
Avg.		0.73	22.05	77.22

TABLE II
RUN TIME OF BEACON ON SUITE2

Solution quality remains relatively flat—across both suites, wire lengths are roughly 32% higher than optimal.

V. OBSERVATIONS ON STRUCTURE AND SUBOPTIMALITY

As part of this work, we have made a number of observations that are relevant to more general-purpose placement applications. In this Section, we summarize these observations, and present figures that illustrate the main points.

Detail Placement is a Key Problem

In Figure 3, we show a color-coding of the placements of a number of tools, on the benchmark PEKO01. The coding is as follows: the higher the Manhattan distance of a cell from a known optimal location, the darker it’s color. The coding drops to black if a cell is more than 30 steps from an optimal location, and is close to white if it is precisely on the optimal location.

It should not be surprising that the “beacon” placement has most cells close to optimal positions. A number of other placement tools, however, also find placements that are close to optimal. In particular, the results of mPL, Feng Shui, and Dragon place almost all cells within 20 steps of their known optimal locations. We note that both Capo and Kraftwerk position many cells correctly, and have portions that are simply shifted or rotated away from optimal locations.

From these graphs, we would suggest that *global placement is performed well by a number of different tools*. If all cells are within 20 steps of an ideal location, it seems unlikely that improvements to global placement algorithms could produce much more benefit.

To further this point, we present Figure 4, which uses the lengths of nets connected to a particular cell to perform the color coding. If the net lengths connected to a cell are close to the lengths in an optimal placement, they are light-colored. We show graphs for placements from before and after legalization

and detail placement, for *Beacon* and *Feng Shui*. Prior to detail placement, total wire lengths are similar—and the physical locations of cells from both tools are close to the optimal locations. During detail placement, however, slightly better relative ordering available with *Beacon* placements results in a far better final placement.

Study of these graphs has resulted in a shift in the focus of our research group to better methods of detail placement. In terms of solution quality, we see relatively little opportunity for gain at the “global” placement level, but a great deal of potential at the “detail” placement level. The placements produced by Beacon, Dragon, mPL, and Feng Shui have the bulk of cells within 20 steps of an ideal location, but have wire lengths that are a minimum of 30% higher than optimal—clearly, if we are to recover this wire length loss, it must come during detail placement optimization.

Irregularity of “Real” Circuits

We have focused on circuits with regularity—and in fact have shown that the PEKO benchmarks contain regularity that was neither intended or expected. The implementation we describe here does not perform well on circuits without internal regularity—the correlation between physical distance and graph distance is not as strong.

The lack of regularity is a significant difference between “real” circuits and the benchmarks. Further, the regularity (and basic construction) of the PEKO benchmarks may lead to pessimism about how far current placement tools are from optimal. All nets in the optimal placements for PEKO benchmarks are contained within minimum area bounding boxes—for small cardinality nets, this means that there are only few configurations that meet the wire length target, and any deviation results in a substantial wire length jump.

“Real” circuits, which lack regularity, do not have optimal placements in which all nets are within minimum area bound-

PEKO bench	Opt. WL	Suite 1 Quality Ratio						Suite 2 Quality Ratio					
		Dragon	Qplace	Capo	mPL 4	FS2.0	Beacon	Dragon	Qplace	Capo	mPL 4	FS2.0	Beacon
01	8.14E+05	1.79	1.69	2.24	1.25	1.53	1.30	2.15	1.77	2.39	1.28	1.92	1.32
02	1.26E+06	1.93	1.82	2.22	1.25	1.62	1.34	2.53	1.94	2.61	1.30	1.88	1.33
03	1.50E+06	1.95	1.9	2.28	1.27	1.76	1.33	2.53	1.83	2.44	1.31	1.88	1.32
04	1.75E+06	2.21	1.9	2.28	1.30	1.78	1.32	2.05	1.85	2.48	1.31	1.99	1.33
05	1.91E+06	1.98	1.95	2.27	1.25	1.64	1.33	2.02	1.84	2.53	1.29	1.98	1.34
06	2.06E+06	2.11	1.75	2.35	1.28	1.70	1.34	2.42	1.93	2.45	1.33	1.86	1.33
07	2.88E+06	2.17	1.85	2.44	1.27	1.77	1.32	> 24h	1.85	2.58	1.30	1.82	1.31
08	3.14E+06	2.16	1.78	2.27	1.29	1.77	1.35	> 24h	1.9	2.42	1.34	2.09	1.37
09	3.64E+06	2.12	1.91	2.38	1.28	1.93	1.31	> 24h	1.84	2.59	1.32	2.04	1.32
10	4.73E+06	1.79	1.88	2.42	1.29	2.02	1.33	> 24h	1.9	2.62	1.35	2.00	1.33
11	4.71E+06	1.94	1.96	2.38	1.28	1.80	1.31	> 24h	1.93	2.58	1.33	2.04	1.30
12	5.00E+06	1.93	1.8	2.5	1.29	1.86	1.34	> 24h	1.82	2.63	1.34	1.97	1.32
13	5.87E+06	2.15	1.78	2.36	1.28	1.86	1.31	> 24h	1.86	2.67	1.37	2.14	1.31
14	9.01E+06	2.01	1.76	2.42	1.31	1.85	1.33	> 24h	1.93	mem	1.45	2.01	1.32
15	1.15E+07	2.17	1.85	2.4	1.29	1.99	1.33	> 24h	1.96	mem	1.91	2.19	1.32
16	1.25E+07	2.26	1.83	2.5	1.27	1.94	1.31	> 24h	1.87	mem	1.71	2.10	1.31
17	1.34E+07	2.5	1.87	2.53	1.29	2.00	1.32	> 24h	mem	mem	2.21	> 24h	1.32
18	1.32E+07	2.38	1.84	2.53	1.28	1.94	1.32	> 24h	mem	mem	mem	> 24h	1.33
Avg.		2.09	1.84	2.38	1.28	1.82	1.32	2.28	1.88	2.54	1.44	2.00	1.32

TABLE III

QUALITY RATIO COMPARISON ON PEKO SUITE 1 AND 2. SOME RESULTS ARE TAKEN FROM PUBLISHED RESULTS; RESULTS FOR FENG SHUI 2.0, MPL4 AND OUR BEACON APPROACH ARE CURRENT. NEWER VERSIONS OF SOME TOOLS HAVE BETTER PERFORMANCE. THE TESTED PLACEMENT TOOLS INCLUDE DRAGON, QPLACE, CAPO, MPL, FENG SHUI, AND OUR BEACON PLACEMENT TOOL. RUN TIMES ARE NOT DIRECTLY COMPARABLE DUE TO DIFFERENT COMPUTING PLATFORMS.

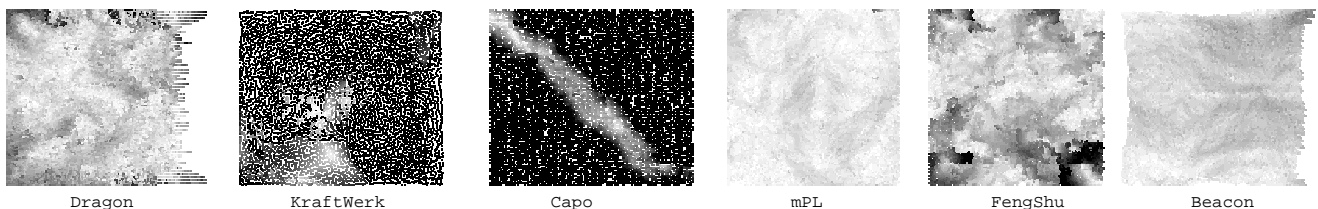


Fig. 3. Color-coding of distance from optimal configurations. Each cell in placements by the different tools were color coded based on Manhattan distance of the cell from an optimal location. Several of the tools produced results that were close to optimal from a global perspective—almost all cells were within 20 steps of an optimal position.

ing boxes. While optimal placements for these designs are unknown (and likely cannot be determined), it would be safe to assume that the optimal placement would have many nets with increased bounding box area, resulting in more configurations that allow deviation from optimal without as severe of a penalty in total wire length.

To illustrate this point: if a two pin net must be placed in a pair of adjacent locations to obtain wire length equal to that of an optimal configuration, the combinations that meet this objective are limited. Any deviation from the ideal configuration incurs a minimum of a 100% increase in length for that net—length will jump from 1 to at minimum 2. If the optimal configuration for a net is in fact larger than the minimum bounding box size, the number of configurations that meet the constraint is larger (making this easier to accomplish), and any deviation from optimality will incur a much smaller percentage increase in wire length.

The “minimum bounding box” construction of the PEKO benchmarks makes them in some respects pessimistic. There is also, however, some optimism. Detail placement optimization

on these benchmarks is easy—all cells are of equal size, simplifying the task of finding legal reconfigurations. The “bin packing” nature found in most placement problems has been removed, giving detail placement algorithms greater freedom.

This issue underscores the importance of detail placement. Given an “easy” problem, there is clearly a wire length loss of at least 30%. With cells of different sizes, the problem becomes harder—it would be reasonable to assume that the wire length loss would be increased.

Architecture-level Decisions

Placement is clearly a difficult problem, and current tools fall far short of optimality. If the ideal structure of a circuit can be determined, however, better placements can be obtained. This leads to an architecture-level design decision: the architecture can be made regular, losing optimization potential but gaining a better placement, or we may have an irregular structure, and hope that an acceptable placement can be found.

Some optimizations, such as individual device sizing, can degrade regularity. Obtaining the benefits of carefully tuned devices may preclude the use of a regular placement structure.

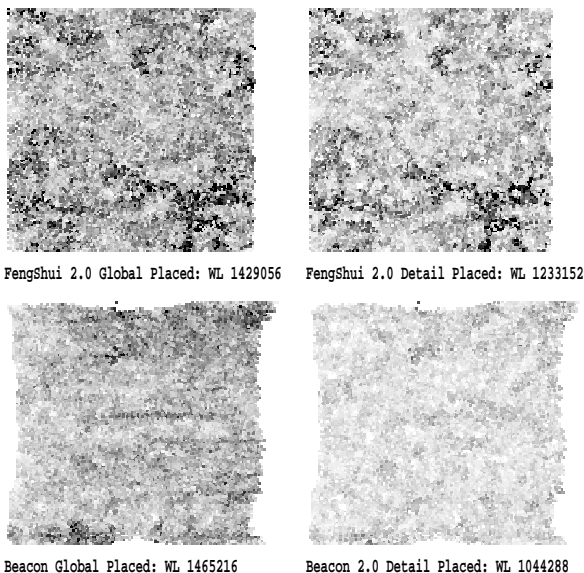


Fig. 4. Wire length difference from optimal before and after detail placement.

Without the structure, there is the inherent performance loss caused by the difficulty of placement.

The rise in interest in “fabric-based” design (for example [6], [11], [19]) reflects these issues. By mapping an architecture onto a regular structure, we lose optimization potential, but gain far better placement results. In [12], it was suggested that regular layouts were not required—with the rise in interconnect delay, and the clear suboptimality of irregular circuit placement, this may no longer be the case.

VI. CONCLUSION AND FUTURE WORK

Over the past few years, there have been great advances in placement research. A convergence on metrics and benchmarks[18], [1] has allowed direct comparison of results. The PEKO benchmarks[5] showed how far current tools were from optimal, and how much performance gain was possible through better placement.

In this work, we have presented a placement approach that finds and exploits structure in circuits. We use the strong relationship between physical distance and graph distance to develop a new method that works exceptionally well on a widely used set of benchmarks. While the PEKO benchmarks may not accurately reflect “real” circuits, and results have a mix of optimism and pessimism, their impact on research focus has been profound.

Our experiments have also highlighted an important problem in physical design: detail placement, perhaps considered effectively “solved,” is clearly responsible for a large percentage of the optimality gap on modern placements. Improving detail placement methods is essential to reaching the next plateau of performance.

Current work in our group focuses on detail placement, and on finding “good” lengths for nets in netlists which do not have regular structure. Preliminary experiments indicate that our “beacon” approach can be extended to a wider class of

circuits, while maintaining the speed and quality which we have obtained on regular circuits. Details of this work will be presented when it is mature.

Acknowledgements: This work was supported in part by an IBM Faculty Partnership Award, SRC Project 947.1, an equipment grant from Intel, NYSTAR MDC, and by funds from the Japanese Ministry of ECSST via Kitakyushu and Fukuoka knowledge-based cluster project.

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] 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.
- [3] A. E. Caldwell, A. B. Kahng, and I. L. Markov. Optimal partitioners and end-case placers for standard cell layout. In *Proc. Int. Symp. on Physical Design*, pages 90–96, 1999.
- [4] A. E. Caldwell, A. B. Kahng, and I. 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, Y. Fan, X. Yang, and Z. Zhang. Architecture and synthesis for multi-cycle communication. In *Proc. Int. Symp. on Physical Design*, pages 190–196, 2003.
- [7] J. Cong and C.-K. Koh. Interconnect layout optimization under higher-order RLC model. In *Proc. Int. Conf. on Computer Aided Design*, pages 713–720, 1997.
- [8] W. J. Dally and A. Chang. The role of custom design in ASIC chips. In *Proc. Design Automation Conf.*, pages 643–647, 2000.
- [9] H. Eisenmann and F. M. Johannes. Generic global placement and floorplanning. In *Proc. Design Automation Conf.*, pages 269–274, 1998.
- [10] D. Hill. US patent 6,370,673: Method and system for high speed detailed placement of cells within an integrated circuit design, 2002.
- [11] B. Hu, H. Jiang, Q. Liu, and M. Marek-Sadowska. Synthesis and placement flow for gain-based programmable regular fabrics. In *Proc. Int. Symp. on Physical Design*, pages 197–203, 2003.
- [12] P. Inne and A. Griessing. Practical experiences with standard-cell based datapath design tools: do we really need regular layouts? In *Proc. Design Automation Conf.*, pages 396–401, 1998.
- [13] A. B. Kahng and C.-W. A. Tsao. Low-cost single-layer clock trees with exact zero Elmore delay skew. In *Proc. Int. Conf. on Computer Aided Design*, pages 213–218, 1994.
- [14] A. B. Kahng and Q. Wang. Implementation and extensibility of an analytic placer. In *Proc. Int. Symp. on Physical Design*, pages 18–25, 2004.
- [15] A. Khatkhate, C. Li, A. R. Agnihotri, M. C. Yildiz, S. Ono, C.-K. Koh, and P. H. Madden. Recursive bisection based mixed block placement. In *Proc. Int. Symp. on Physical Design*, pages 84–89, 2004.
- [16] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, May 1983.
- [17] 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 and Systems*, 10(3):356–365, 1991.
- [18] P. H. Madden. Reporting of standard cell placement results. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 21(2):240–247, February 2002.
- [19] D. D. Sherlekar. Design considerations for regular fabrics. In *Proc. Int. Symp. on Physical Design*, pages 97–102, 2004.
- [20] W. Swartz and C. Sechen. Timing driven placement for large standard cell circuits. In *Proc. Design Automation Conf.*, pages 211–215, 1995.
- [21] N. Viswanathan and C. C.-N. Chu. Fastplace: Efficient analytical placement using cell shifting, iterative local refinement and a hybrid net model. In *Proc. Int. Symp. on Physical Design*, pages 26–33, 2004.
- [22] M. Wang, X. Yang, and M. Sarrafzadeh. Dragon2000: Standard-cell placement tool for large industry circuits. In *Proc. Int. Conf. on Computer Aided Design*, pages 260–263, 2000.