Further Studies on Placement Optimality

Jason Cong, Michalis Romesis, Joseph R. Shinnerl, and Min Xie

UCLA Computer Science Dept.

Abstract

Recent studies of the quality of results produced by leading placement tools on synthetic benchmarks with known optimal placements (PEKO) have generated both interest and skepticism in the physical design community. While the studies show computed wirelengths typically 50–150% above the optimal PEKO wirelengths, it is not yet known whether these figures accurately reflect true optimality gaps obtained on real circuits. In order to identify deficiencies remaining in existing placement algorithms, two simple extensions and one adaptation of the PEKO benchmarks are presented. The extensions incorporate non-uniform cell sizes and non-local nets. A new set of PEKO examples with arguably closer resemblance to standard industrial benchmarks is introduced, for which leading academic tools consistently compute placements with total half-perimeter wirelength less than 5–15% above optimal. Based on these observations, netlist characteristics that make high-quality placement easier are described. The adaptation constructs simple, uniformly accurate global placements of PEKO for use as parametrized detailed-placement benchmarks. The relative impact of global and detailed placement on solution quality are quantified, visualized, and analyzed.

I. Introduction

Placement is a critical step in VLSI circuit design. Interconnect delay dominates system performance, and placement determines the interconnect more than any other step in physical design. The complexity of modern designs, however, makes estimation of bounds for optimality difficult.

Recent studies on simplified, synthetic benchmarks with known optimal-wirelength placements (PEKO, [7]) suggest that many leading tools may produce solutions with excess wirelength of up to 50–150% or more. This result has generated wide interest in both industry [15], [14], [13] and academia [17], [10], [6], [9], [21], [22]. However, these studies have well-known limitations. Although the PEKO benchmarks' cell count, net count, and net degrees match the corresponding quantities in standard industrial benchmarks [2], the PEKO circuits are simplified in two key ways, in order to guarantee known optimal solutions, First, all cells are squares of the same size. Second, all nets in an optimal PEKO placement are local — the netlist of a PEKO circuit is defined first by arranging cells in a regular array and then grouping adjacent cells into nets such that each net has the minimum possible wirelength. Subsequent studies [17] derived useful *upper bounds* on the optimal placements of circuits with both local and non-local nets (PEKU). The gaps observed for leading tools on these benchmarks were considerably smaller, though still significant. However, it is not known how close the PEKU bounds are to the minimum possible wirelengths for the PEKU circuits.

Recent progress in placement [34], [8], [19], [4], [1], [25], [12] has also raised questions concerning the traditional division between *global* and *detailed* placement. Generally, both global and detailed placement proceed by iterative improvement. The goal of global placement is to position each cell within some relatively small neighborhood of its final position, while eventually obtaining a sufficiently uniform distribution of cell area over the entire chip. Typically, large sets of cells are moved simultaneously under some relaxed or incremental formulation of area density control — scalable algorithms do not strictly enforce pairwise nonoverlap constraints during this stage. The goal of detailed placement is, given a sufficiently good global placement, determine the final positions of all cells so that (i) no two cells overlap and (ii) a given objective, e.g., total wirelength, is minimized. Typically, detailed placement proceeds by a sequence of refinements made one at a time on small, contiguous subregions [11], [3].

How much of the optimality gap left by contemporary methods should be attributed to deficiencies in global-placement algorithms, and how much to detailed? In practice, global placement is terminated when iterations are observed to make little or no reduction in the objective and the cell-area distribution is sufficiently uniform. On a real circuit, there is no way of knowing how far a cell is from its nearest optimal location, at any stage. On the PEKO circuits, however, it is possible to precisely determine the quality of either a global or detailed placement engine in isolation from its detailed or global counterpart. Thus, the PEKO circuits provide a more precise means of quantifying the relative effectiveness and the relative difficulty of the two stages.

The goal of the work described here is to further elucidate the nature of the placement optimality gap. Three sets of experiments are described and analyzed. In the first, a simple modification of a PEKO solution yields another PEKO solution for a circuit with varying cell widths. These PEKO circuits with non-uniform cell sizes are called “PEKO-NU” (Section 2). In the second, placement examples with *both* known optimal solutions and non-local nets are constructed. The key idea is to organize all non-local nets into disjoint chains of nets connecting well separated fixed terminals (pads) along the chip boundary. Optimal wirelength is guaranteed by constructing a placement in which the nets in each chain have bounding boxes which overlap minimally and which are ordered monotonically along the two directions defined by the chip boundary (Section 3). In the third set of experiments, simple aggregation of cells in an optimal plain-PEKO solution into the centers of bins of user-specified size yields “perfect” global placements of PEKO, parametrized by the bin size. These benchmarks are called

"PEKO-DP." They are useful for gauging the difficulty of detailed placement with respect to the precision of the global placement (Section 4).

II. PEKO-NU

The PEKO algorithm [7] takes an integer $n > 0$ and a vector $D_{|e|}$ of net degrees and outputs a placement example which has (i) n cells, (ii) net-degree profile $D_{|e|}$, and (iii) known optimal half-perimeter wirelength. The original PEKO study [7] showed that, in early 2003, leading placement algorithms were 50% to 150% away from the optimal solutions on examples thus constructed. However, the cells of the original PEKO are of uniform size. How much of the PEKO gap might be attributed to a placer's ability to exploit the uniform-cell-size simplification?

To address this question, we extended the PEKO algorithm by introducing non-uniform sized cells into PEKO. The new algorithm is named PEKO-NU. In addition to the number of cells n and the net-degree vector $D_{|e|}$, PEKO-NU also takes a second integer, m, specifying the maximum width of any cell as a multiple of the width of the smallest cell, and a second vector, D_w, specifying the desired distribution of cell widths in the benchmark. Each element of D_w is a pair consisting of a cell width and the fraction of cells that should have that width in the circuit; hence, $\sum_i (D_w)_i = 1$. Just as in PEKO, PEKO-NU outputs a circuit matching the requested net-degree profile $D|e|$ whose minimum attainable *pin-to-pin* wirelength is known. The cell-width distribution D_w cannot generally be matched exactly, but it is used as a guide in the construction of non-uniformly sized cells. Nets are generated to have locally optimal half-perimeter pin-to-pin wirelength, just as in PEKO.[1] Unlike PEKO, however, pins are no longer placed at cell centers, and there is no claim that the PEKO-NU wirelength remains optimal if all pins are moved to cell centers, as is commonly done by academic placers. However, intuition and experiments suggest that this center-to-center wirelength is very close to optimal.

The PEKO-NU construction starts from an optimal $\lceil\sqrt{n}\rceil \times \lceil n/\lceil\sqrt{n}\rceil\rceil$ PEKO placement, in which all cells are unit squares, and proceeds as follows. The cells are visited one by one, starting from one corner of the chip. When examining cell c_i, we select an integer j, $1 \le j \le m$, and try to group cells from c_i to c_{i+j}. The group size j is selected to improve the cell-width distribution. For the grouping to succeed, the wirelengths of the nets containing cells in the group must remain optimal. Nets that connect only the cells within the group will be removed. For a net that also connects cells outside the group, the centers of the unit cells forming the group cell are examined as candidate pin positions one by one. Some pin choices may result in suboptimal nets, as shown in Figure 1. If, however a pin position is found for which the net length is minimal, it is retained,

[1] It is not generally possible to match both $D_{|e|}$ and D_w exactly, when every net is individually required to have minimal wirelength. The PEKO-MC method described in Section 4, however, *does* match both $D_{|e|}$ and D_w exactly, by carefully incorporating non-local nets.

and the group becomes a single cell. Otherwise, the candidate group is disbanded, and attempted grouping repeats at the next cell. In the end, the total cell area is preserved, but grouping reduces the number of cells from the original n by about 10–20%.

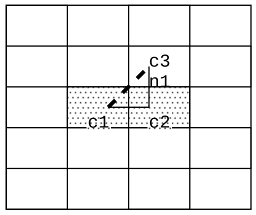


Fig. 1. Attempted grouping of adjacent cells to form a larger cell in a benchmark circuit. Originally a 3-pin net n_1 (solid line segment) connects uniform-sized cells c_1, c_2, and c_3. After c_1 and c_2 are grouped together, n_1 becomes a 2-pin net joining a pin at the center of c_3 to a single pin in the center of either the left half or the right half of the grouped cell. As shown, the new 2-pin net (dashed line) is not of minimal length. Moving the group-cell's pin to the center of its right-hand half will restore optimal netlength and render the grouping successful.

A. PEKO-NU Experiments

Table I shows the characterstics of the new PEKO-NU benchmarks, their optimal wirelengths and maximum cell width ratios. Results of test runs of Capo 8.7, Feng Shui 2.6, Dragon 3.01, mPL4, and mPL3 on PEKO-NU and PEKO are shown in Figure 2. With the exception of mPL3, the results on PEKO-NU do not differ significantly from the results on plain PEKO. This suggests that, for most tools, the uniform cell width of the PEKO circuits does not by itself make PEKO circuits easier or harder to place than the industrial test cases on which they are based. The large gap between mPL3's and mPL4's results can be attributed to mPL's development history; versions before mPL4 were tested primarily on benchmarks with uniform cell sizes.

mPL4 produces superior quality ratios on nearly all the PEKO benchmarks. The reason for this is believed to lie in the ability of the aggregation-based approach to correctly position cells in local nets for optimal wirelength.[2]

III. PEKO-MC

In contrast to the PEKO [7] and PEKO-NU circuits described in Section 2 the PEKO-MC (PEKO-Monotone Chains) examples contain non-local nets. The PEKO circuits have previously been extended to contain non-local nets in the PEKU circuits [17].

[2]An alternative method of correlating placement distance with netlist distance, based on triangulation of cell positions from fixed corner cells, has recently been developed by Ono and Madden as an enhancement to Feng Shui for circuits with "regular structure" [22]. However, as of this writing, the implementation of that enhancement is not yet effective on designs with non-uniform cell sizes.

Circuit	#cells	#nets	C-C HPWL	P-P HPWL	$\frac{w_{\max}}{w_{\min}}$
Peko-nu01	13123	16920	1.25E+06	952448	17
Peko-nu02	21450	23476	1.74E+06	1.50E+06	16
Peko-nu03	23542	32868	2.35E+06	1.75E+06	18
Peko-nu04	26974	38345	2.88E+06	2.01E+06	21
Peko-nu05	31990	34126	2.57E+06	2.33E+06	11
Peko-nu06	33928	41781	3.07E+06	2.42E+06	18
Peko-nu07	45874	57732	4.56E+06	3.33E+06	19
Peko-nu08	53930	60588	4.64E+06	3.69E+06	21
Peko-nu09	55775	73071	5.38E+06	4.27E+06	20
Peko-nu10	74349	90224	6.63E+06	5.63E+06	15
Peko-nu11	69027	97736	7.61E+06	5.42E+06	21
Peko-nu12	76960	92680	6.90E+06	5.96E+06	18
Peko-nu13	86426	119590	8.78E+06	6.87E+06	18
Peko-nu14	143218	183316	1.47E+07	1.03E+07	21
Peko-nu15	171848	223914	1.64E+07	1.36E+07	18
Peko-nu16	196544	228045	1.76E+07	1.48E+07	20
Peko-nu17	203832	227483	1.80E+07	1.60E+07	15
Peko-nu18	214083	242279	2.00E+07	1.53E+07	21

TABLE I

CHARACTERISTICS OF THE PEKO-NU CIRCUITS. OPTIMAL PIN-TO-PIN HALF-PERMETER WIRELENGTH (HPWL) AND CORRESPONDING CENTER-TO-CENTER HPWL ARE DISPLAYED IN COLUMNS 4 AND 5. THE LAST COLUMN DISPLAYS THE MAXIMUM RATIO OF ANY TWO CELL WIDTHS IN THE CIRCUIT.

PEKU, however, employs only a limited number of randomly generated non-local nets and provides only an upper bound on optimal wirelength, rather than a known optimal solution. The PEKO-MC examples generate over 50% non-local nets in such a way that an optimal-wirelength placement is known. The key idea is to ensure that the non-local nets can be grouped into net-wise disjoint monotone chains of nets, as discussed below. In addition, the PEKO-MC netlists resemble the industrial netlists from which they are derived much more closely than “plain” PEKO [7]. The PEKO circuits match the real benchmarks only in cell count and net-degree distribution. The PEKO-MC examples match not only net-degree distribution and cell-area distribution exactly, but they also leave over 60% of the nets of the original netlist completely unmodifi ed.

A. Monotone Hyperpaths

A *monotonic* path of a planar graph is a path whose total length is equal to the Manhattan distance of its two terminal vertices.

Figure 3 shows an example of a monotonic path. If (x_i, y_i) are the coordinates of node n_i, then it is easy to see that :
$|x_3 - x_0| + |y_3 - y_0| = |x_1 - x_0| + |y_1 - y_0| + |x_2 - x_1| + |y_2 - y_1| + |x_3 - x_2| + |y_3 - y_2|$
Thus, the path in Figure 3 is monotonic.

As a direct result of the definition, a path with n edges (e_1,..., e_n) connecting $n + 1$ vertices (v_0,...,v_n), vertex v_i with coordinates (x_i, y_i) , and edge e_i connecting vertices v_{i-1}, v_i with $x_n \geq x_0$ and $y_n \leq y_0$ is monotonic if and only if for every i, $1 \leq i \leq n$: $x_i \geq x_{i-1}$ and $y_i \leq y_{i-1}$. Similar conditions can be derived for different relations between the coordinates of the terminal vertices.

Here we extend the definition of a monotonic path for hyperpaths of a planar hypergraph. In a planar hypergaph, the length of a hyperedge is the half-perimeter of the minimum bounding box that encloses that hyperedge, and the total length of the hyperpath is the sum of the lengths of its hyperedges. An edge e is called the equivalent edge of a hyperedge h of a planar hypergraph, if it connects two of the vertices of h, such that e's minimum bounding box is the same as h's minimum bounding box. Note that a hyperedge may not have an equivalent edge, or may have one or multiple equivalent edges. Figure 4 shows examples of hyperedges and their equivalent edges.

A path P is called the *equivalent path* of a hyperpath H on a planar hypergraph, if there is a one-to-one correspondence between the edges of P and the hyperedges of H, such that every edge of P is the equivalent edge of its corresponding

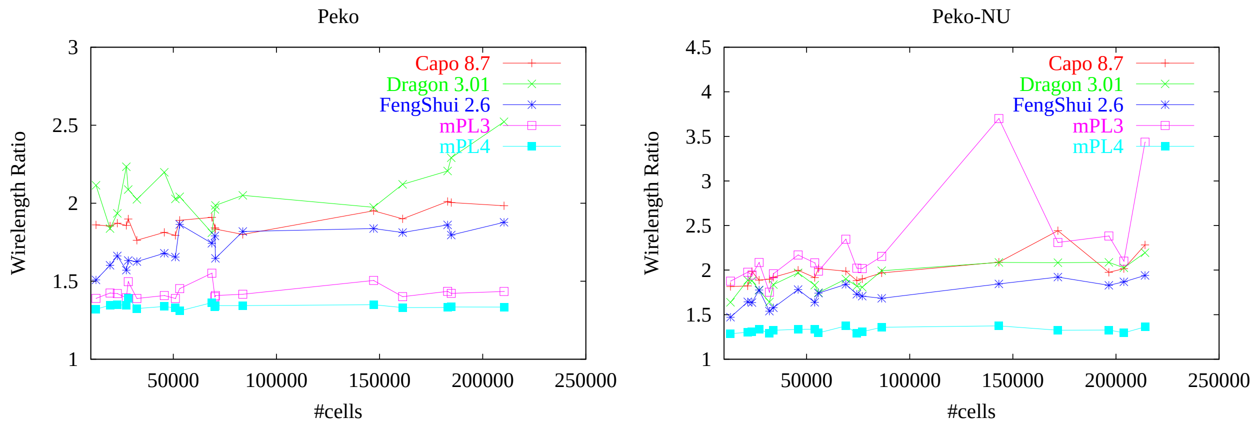


Fig. 2. Wirelength results for leading academic placers on the PEKO (top) and PEKO-NU (bottom) benchmark circuits.

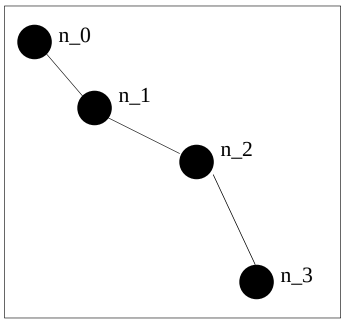


Fig. 3. Example of a monotonic path.

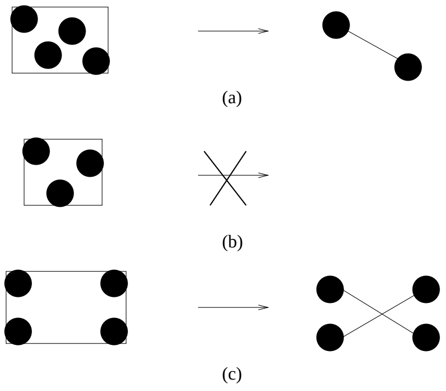


Fig. 4. Examples of hyperedges and their equivalent edges. For every hyperedge on the left, its minimum bounding box is shown. (a) A hyperedge that has one equivalent edge, (b) A hyperedge that does not have an equivalent edge, (c) A hyperedge with two equivalent edges.

hyperedge in H. An example is shown in Figure 5.

Note that not all hyperpaths have an equivalent path even in the case of all their hyperedges having equivalent edges, as shown in Figure 6. We define as *monotonic hyperpath*, the hyperpath which has an equivalent path that is monotonic. An example of a monotonic hyperpath is shown in Figure 7. If a hyperedge has more than one equivalent edges, one of them has to be selected for the creation of the equivalent path. It is easy to see that if a hyperpath is monotonic, this selection is unique for every hyperedge of that hyperpath (assuming that no vertices can occupy the exact same location on the plane and that the hyperpath has more than one hyperedge). It is also easy to see that two neighboring hyperedges in a monotonic hyperpath have exactly one vertex in common (again assuming that no vertices can occupy the exact same location) and these common vertices form the equivalent monotonic path. The terminal vertices of a monotonic hyperpath are the terminal vertices of its equivalent path.

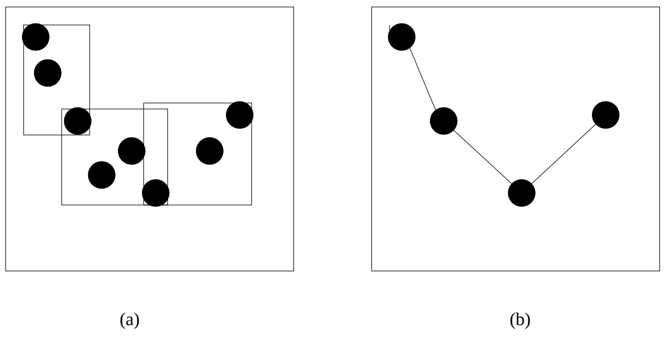


Fig. 5. A hyperpath (a) and its equivalent path (b). The minimum bounding boxes of the hyperedges of the hyperpath are shown in (a).

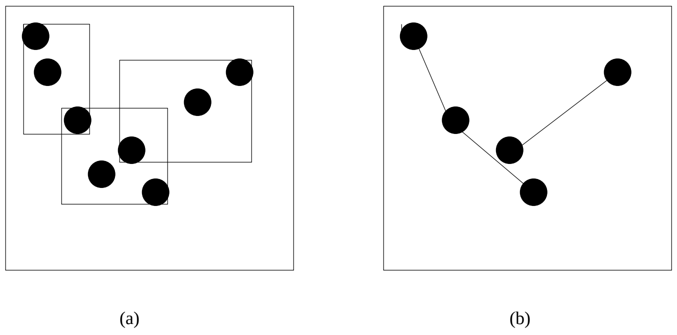


Fig. 6. (a) A hyperpath that does not have an equivalent path. (b) The equivalent edges of the hyperedges of the hyperpath are not forming a path.

After the above definitions, we proceed with the main theorem of this document:

Theorem 1: If the terminal vertices of a monotonic hyperpath $P = (h_1, h_2, ..., h_n)$ of a planar hypergraph G are fixed on the plane, there is no other embedding of hypergraph G that can make the length of hyperpath P shorter.

Proof: By definition, a monotonic hyperpath has an equivalent monotonic path with total length equal to the Manhattan distance k of its terminal vertices, which in this case is fixed. Since the length of a hyperpaths is equal to the length of its equivalent path, we conclude that the length of P is also k. We will now show that there is no other embedding of the hypergraph on the plane, without changing the location of the terminal vertices, that can make the length of P less than k. Since P is monotonic, we know that neighboring hyperedges have exactly one vertex in common. We define a sequence of vertices v_i with coordinates (x_i, y_i), $0 \leq i \leq n$, with v_0, v_n as the terminal vertices that are members of hyperedges h_1 and h_n respectively and v_i, $1 \leq i \leq n-1$ as the common vertex of hyperedges h_i and h_{i+1}.

For any embedding of the hypegraph, the total length of P is the sum of the lengths of hyperedges h_i :

$length(P) = \sum_{i=1}^{n} length(h_i) \geq \sum_{i=1}^{n}(|x_i - x_{i-1}| + |y_i - y_{i-1}|) \geq |x_n - x_0| + |y_n - y_0| = k.$

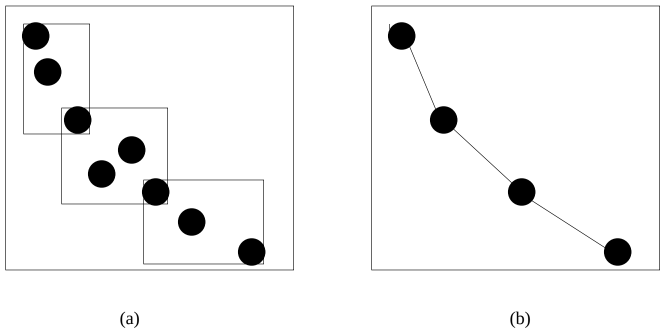


Fig. 7. A monotonic hyperpath (a) and its monotonic equivalent path (b).

So, the length of P can never be less than k, no matter what the values of (x_i, y_i) are. As a conclusion, the embedding of hypergraph G that makes hyperpath P monotonic is optimal in terms of the length of P, given that x_0, y_0, x_n, y_n are fixed.

We use Theorem 1 for the creation of placement examples with known optimal wirelength in the presence of non-local nets. We define here as a local net, a hyperedge of a planar hypergraph whose length is the minimum possible given some spacing constraints between the vertices. Otherwise we define a hyperedge as a non-local net. Theorem 2 shows under what conditions the embedding of a hypergraph is optimal in terms of the sum of the lengths of its hyperedges.

Theorem 2: An embedding of a hypergraph G in the plane is optimal in terms of the sum of the lengths of its hyperedges, if all of its non-local nets can be partitioned into disjoint groups, where each group corresponds to a monotonic hyperpath with fixed terminal vertices.

Proof: For the embedding of G, each of G's hyperedges is either a local net or can be assigned to exactly one of a set S of monotonic hyperpaths with fixed terminal vertices (note that it can be a member of multiple monotonic hyperpaths with fixed terminal vertices, as long as only one of them belongs to S).

Assume that G has n hyperedges $(h_1, h_2, ..., h_n)$, E_1 is the set of its non-local nets, and E_2 is the set of its local nets. The total length of the hyperedges is :

$$\sum_{i=1}^{n} length(h_i) = \sum_{h \in E_1} length(h) + \sum_{h \in E_2} length(h) = \sum_{P \in S} length(P) + \sum_{h \in E_2} length(h)$$

We see that the total length of the nets is the sum of the lengths of monotonic paths with fixed terminal vertices and the lengths of local nets. From Theorem 1 and the definition of local nets, it is easy to see that none of the terms of these sums can be further minimized for any other possible embedding of that hypergraph, so the total net length for this embedding is optimal.

We can use Theorem 2 to generate benchmarks with a known wirelength-optimal placement solution. Starting from a placement of a real circuit, we can modify the non-local nets of the circuit with the purpose of generating disjoint monotonic hyperpaths with fixed terminal vertices. The nets that are already local do not need any modification. By modifying the netlist as little as possible, we can generate examples that are both realistic and with known optimal solution.

B. The PEKO-MC Algorithm

The methodology for the creation of the examples is based on the concept of monotone hyperpaths that was presented in the previous section. The idea is, starting from the placement of the real benchmark, to identify sets of nets that can be grouped together into monotone hyperpaths (or monotone chains as we will call them for the rest of the paper). Initially, these chains have some empty regions, which are later filled by other nets that have to be modified from the original netlist.

The algorithm consists of 5 steps: placement generation, net categorization, chain generation, chain removal, and gap covering.

Placement generation: The PEKO-MC algorithm needs a placement of the cells of the original hypergraph in order to identify the monotone chains. This placement stays fixed throughout the algorithm, while the connectivity of the nets changes. Starting from a random placement is possible, but our experiments showed that starting from a placement output from a placement tool increases the final similarity of the original and the derived circuits. The main reason for this increase is the conservation of the locally optimal nets of the original circuit, since as shown in Theorem 2 of the previous section, these do not have to be members of the monotone chains. Obviously, a placement output from a tool will generate considerably more locally optimal nets than a random placement.

Net categorization: The nets of the original hypergraph can be divided into three different categories depending on the placement of their pins. The first category are the locally optimal nets, as defined in the previous section. Assuming that all the pins are at the centers of their corresponding cells, it is possible to compute the minimum possible wirelength for any hyperedge. In our experiments, we skipped the identification of the locally optimal nets among the nets of degree four or higher, since the computations can be time consuming, and the percentage of nets of such degrees is small compared to the two and three-pin nets in most real circuits. The second category are the nets that do not have equivalent edges, as defined in the previous section. These nets cannot be members of monotone chains, and they have to be modified in the final netlist. The final category are the nets that have equivalent edges and can be members of monotone chains. These nets are further divided into two sets, depending on the direction of the chains that they can be members of. The monotone chains can have two possible directions : from the lower left part of the chip to the upper right, and from the lower right to the upper left. A net with an equivalent edge can be a member of a chain with the same direction as the equivalent edge. For example the net shown in Figure 4a can only be a member of chains with direction from lower right to upper left, while the net in Figure 4c can be a member of any chains. For the nets that can be members of chains of any direction, the algorithm will primarily consider them for chains of one direction, but these nets can still become members of chains of the other direction if some conditions are satisfied. The details for these conditions are omitted, since they do not change the essence of the algorithm.

Chain generation: During this step, sets of nets that can be members of the same chain are identified, as well as sets of empty regions that are needed to be filled later by nets in order to complete the monotone chains. All the nets that belong to the third category from the previous step are assigned to chains. Initially, the chains of the direction from lower left to upper right are created. Nets are selected sequentially and they have to satisfy three conditions: First, the net must be eligible to be a member of chains of that direction, second, the lower left corner $A(x_1, y_1)$ of its bounding box and the upper right corner $B(x_2, y_2)$ of the bounding box of the previous net of the chain (we assume the point B(0,0) when selecting the first net of the chain) must satisfy the conditions : $x_1 \geq x_2$ and $y_1 \geq y_2$, and, third, the manhattan distance between A and B is the minimum among the candidate nets. If that manhattan distance is more than 0, the nets are disconnected (we assume that all the pins of a cell are in the same location) and an empty region whose bounding box is defined by the corners A and B is inserted in the chain. If no nets can satisfy the conditions and the chain has not been completed, then an empty region that covers the upper right part of the chip is inserted. After one chain is complete, the procedure is repeated for a new chain until all the candidate nets are assigned. A similar procedure is followed for the chains of the other direction. An example of a chain generation is shown in Figure 8.

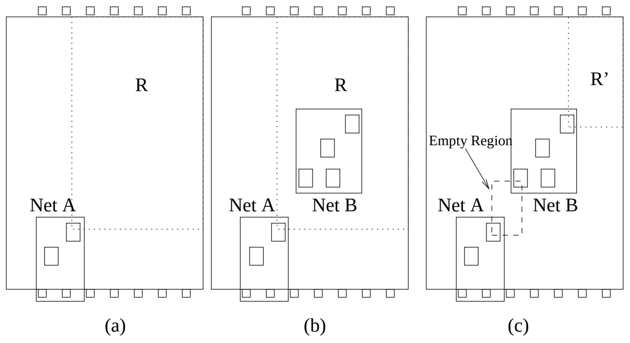


Fig. 8. Example of chain generation. (a) Net A has already been added to the chain. A search for a new net takes place in region R. (b) Net B is selected to be added to the chain. (c) An empty region is inserted between nets A and B, that will be covered later by a new net. A new search is initiated for nets in region R'.

Chain removal: After the third step, a set of empty regions has been generated that has to be covered by the nets of the second category. However, in all the examples that we tried, the number of empty regions is higher than the number of nets, which means that some of the chains that were generated earlier have to be removed in order to reduce the number of empty regions and increase the number of available nets. In this step, the generated chains are ordered in decreasing order of the metric $Empty\_Regions/Nets$, which is highest for the chains with only one net, and two empty regions that connect that net to the borders of the chip. The chains are visited in that order and are removed one by one until the number of empty regions becomes equal or larger to the number of available nets, which is gradually increased by the addition of nets of the removed chains.

Gap covering: In the final step, the empty regions are covered by modifying the available nets into new ones that maintain the same degree and can complete the monotone chains. When a net is selected to cover an empty region, it must include the pins that were used to define the empty region (see Step 3 of the algorithm), while the remaining pins are selected from the cells that lie within the bounding box of the region. The cells, whose degree in the current netlist, is smallest compared to their original degree, are given priority. By doing this, the cell degree distribution of the new netlist becomes very close to the distribution of the original circuit. If an empty region reaches the borders of the chip, then a pad within that region which will serve as an anchor for the monotone chain is added to the net. Most empty regions are covered by one net, but there are a few that have to be covered by two nets, since we cannot guarantee the number of empty regions being exactly equal to the number of available nets. An example of the covering of an empty region with two nets is shown in Figure 9. In theory, the covering of all the regions is not guaranteed. In our experiments, by carefully selecting the matching of empty regions with nets (larger regions for larger degree nets), we were able to complete all the monotone chains for all cases.

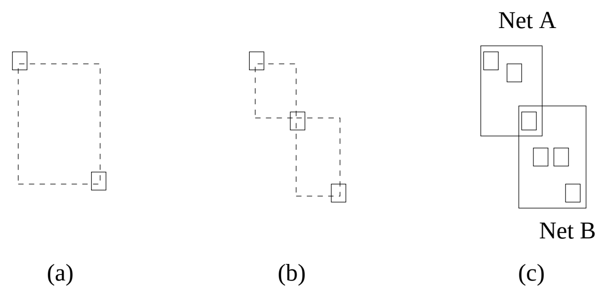


Fig. 9. Covering of an empty region with two nets. (a) Region that has to be covered. A four-pin net and a three-pin net are selected to cover this region. (b) A cell within the region is selected as the link between the two nets. Two new smaller regions are generated. (c) One more pin is selected to form net A and two more for net B.

C. PEKO-MC Experiments

All PEKO-MC benchmarks used in our experiments are generated from the FastPlace [32], [8] versions of the 2002 IBM/ISPD benchmarks [29], [2]. The FastPlace benchmarks modify the original IBM benchmarks by replacing macros with standard cells. Because these examples include pads and do not have macros, they are more accessible to many academic placement tools. However, the PEKO-MC algorithm can also be applied to examples with macros for the generation of mixed-size placement examples with known optimal solutions.

The PEKO-MC benchmark generator described in the previous section requires as input both a netlist and an initial "seed" placement of that netlist. Thus, results obtained on a PEKO-MC circuit may depend not just on the netlist used but also on how the initial placement used to seed the generator is obtained. In particular, it seems likely that a PEKO-MC example may be solved best by the program that generates the seed placement. For a fair comparison, two separate sets of PEKO-MC examples were generated: mPL-PEKO-MC and Dragon-PEKO-MC. Programs mPL4 [33] and Dragon 3.01 [30] generated the seeds for the respective suites bearing their names. Both suites match the FastPlace-IBM benchmarks exactly in number of cells, cell areas, number of nets, and net-degree distribution. Unlike PEKO and PEKO-NU, PEKO-MC maintains an unambiguous one-to-one correspondence between cells in the original, "real" benchmark and cells in the synthetic benchmark it produces. In terms of this correspondence, roughly 60–70% of the nets in the original and synthetic benchmarks are the same. Moreover, the cell degree distributions of the original and synthetic benchmarks are very similar. Figure 10 shows the distribution of the cell degree differences between the cells of mPL-PEKO-MC01 and their corresponding cells from FastPlace-ibm01. Almost 80% of the cells have a pin degree difference at most 1 from their corresponding cells. Figure 11 shows the strong similarity between the wirelength distributions of the nets of FastPlace-ibm01 as placed by mPL4 and the nets in their optimal placement of mPL-PEKO-MC01.

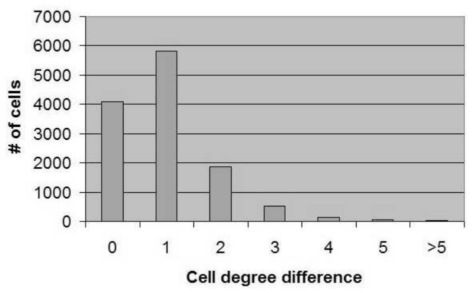


Fig. 10. The cell degree difference (in absolute values) distribution between the cells of mPL-PEKO-MC01 and their corresponding cells in FastPlace-ibm01.

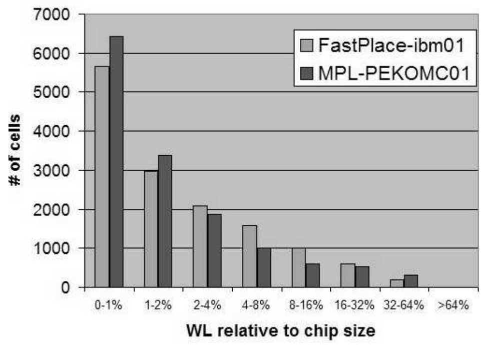


Fig. 11. The wirelength distribution (relative to the chip half-perimeter) of the nets in FastPlace-ibm01 (as placed by mPL4) and the nets in mPL-PEKO-MC01 (in their optimal plcements).

Although no new pads are explicitly inserted, most existing pads are connected to several nets each. Programs Feng Shui 2.6 [31], mPL4 [33], Dragon 3.01 [30], and Capo 8.8 [28] were tested on the two PEKO-MC suites. Figures 12 and 13 show the results obtained on the Dragon-PEKO-MC and mPL-PEKO-MC examples, respectively.

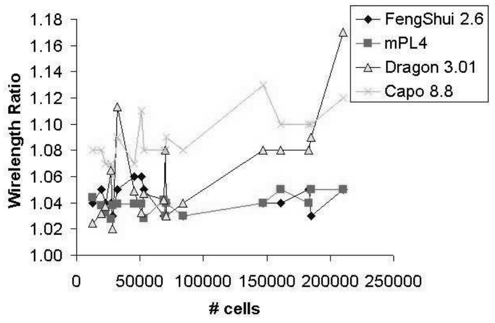


Fig. 12. Results of leading academic tools on PEKO-MC Circuits seeded by Dragon 3.01 placements of FastPlace-IBM Benchmarks.

The overall results show very good performance by all tools on all the benchmarks, regardless of which tool generates the initial placement used to seed the benchmark construction. Although mPL4 and Dragon do indeed perform better on examples generated from their own initial solutions, the advantage appears slight. Dragon’s average quality ratio improves from 1.07 on mPL-PEKO-MC to 1.06 on Dragon-PEKO-MC, while mPL’s quality ratio improves from 1.04 on Dragon-PEKO-MC to 1.03 on mPL-PEKO-MC. The worst reported quality ratio by any of the placers on any benchmark is 1.17; on average, quality ratios range between 1.03 and 1.09. The contrast between these small gaps and gaps observed on the PEKO examples [7] and the PEKO-NU examples in Section 2 is striking. Reasons for this “gap between the gaps” are discussed next.

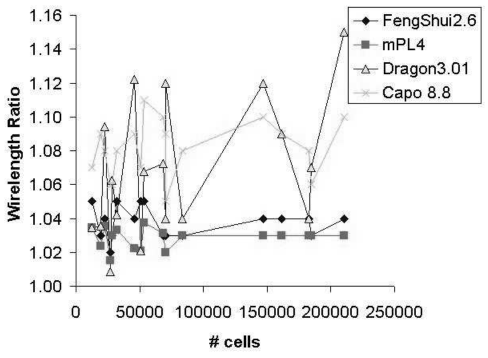


Fig. 13. Results of leading academic tools on PEKO-MC Circuits seeded by mPL4 placements of FastPlace-IBM Benchmarks.

D. Interpretation of the Results

The tight quality ratios observed for the PEKO-MC circuits come as a surprise, compared to the large quality ratios observed on PEKO and PEKO-NU. We believe this "gap between the gaps" can be attributed to three distinct netlist differences between the benchmark sets: (i) the absolute sizes of the optimal wirelengths, (ii) relative sensitivities to cell displacements, and (iii) connectivity to fixed pads. This conclusion is explained in more detail below.

First, a comparison of the optimal wirelengths for the PEKO-MC circuits and PEKO-NU circuits shows that, although the cell and net counts differ by only 10–20%, the optimal half-perimeter wirelengths (HPWL) differ by roughly 2–4×. Thus, comparable displacements of cells from optimal locations will result in a larger optimality gap on PEKO or PEKO-NU than on PEKO-MC. I.e., the much shorter wirelengths of the plain PEKO examples make them more sensitive to changes in their cells' positions.

Second, as has been previously observed [22], the sensitivity of PEKO's and PEKO-NU's optimal wirelengths to cells' displacements is further amplified by the fact that all nets in their optimal solutions are local. When a cell in a local net is moved, the relative increase in that net's HPWL is greater than when a cell in a non-local net is moved. In the case of a cell forming the overlapping corners of successive nets in a monotone chain, the cell can in fact be moved anywhere within the bounding box of *both* nets with *zero* increase in total wirelength. We refer to the union of all possible locations for a cell in all optimal placements as the "optimality region" of a cell. Because most nets in PEKO-MC01 belong to some monotone chain, most cells' optimality regions are larger in PEKO-MC01 than in PEKO or PEKO-NU. Calculations for PEKO-MC01 (not shown) suggest that the average area of an optimality region of a cell is approximately 0.7% of the total area of the chip — an area approximately 100× larger than the average cell area. This figure is actually reduced significantly (5–10×) by the approximately 20% of cells in PEKO-MC that belong only to locally optimal nets. Thus, (i) for most cells, it is relatively easier to find them good locations in PEKO-MC than in PEKO or PEKO-NU, and (ii) displacing cells from optimal locations in PEKO-MC has far less relative wirelength impact than displacing cells from optimal locations in PEKO or PEKO-NU by a comparable distance. Average displacement-from-optimal-location distances for computed solutions to the PEKO-MC and PEKO(-NU) circuits were observed to agree within $2X$.

Third, the netlist of PEKO-MC supports the construction of chains by a significant increase in the number of direct connections between nets and pads. In fact, in order to produce sufficiently many monotone chains, PEKO-MC connects roughly 5–15 distinct nets to each pad, rendering its benchmarks both somewhat unrealistic and perhaps easier to place than industrial circuits. Evidence for this view can be drawn from experiments with *unconstrained* HPWL minimization, which can be solved by linear programming (LP) to global optimality for a lower bound on the constrained optimal HPWL. An LP formulation of HPWL minimization is well known as rectilinear-distance facilities location (RDFL) [16]. On the real IBM/ISPD98 test cases, the RDFL lower bound underestimates wirelengths of computed placements, and presumably optimal placements as well, by a factor well over $6\times$. However, on the PEKO-MC examples, the RDFL is consistently only 1–5% less than the true optimum. Examination of the placements generated (not shown) indicates clearly that additional connections to well distributed pads automatically spread cells out, producing placements much closer to overlap-free configurations.

We believe that the surprisingly small wirelength gaps observed on the PEKO-MC examples may have important implications for circuit design. The concept of a monotone *path* for a circuit signal has been used recently in performance-driven logic synthesis [26], [23], coupled timing-driven placement and logic synthesis [27], and the analysis of wirelength models in timing-driven placement [24]. Although the PEKO-MC study in this section is concerned only with optimal wirelength and not with timing delay, it supports the general conclusion reached by these earlier studies: circuits designed to support placements with most or all signals following monotone or nearly monotone paths from point to point can be expected to have near-optimal placements and therefore superior performance. Our study shows that existing wirelength-driven placement tools appear quite adept at arranging nets of cells to follow monotone chains, when they exist. However, the ability to correctly organize the chains probably depends considerably on the presence of many separate strong connections to fixed, well distributed I/O pads. However, we caution that a low gap on the PEKO does not necessarily indicate a low gap on real benchmarks. “Pad sharing” by the PEKO-MC nets is not generally feasible in real circuits, and it appears to make the PEKO-MC benchmarks easier to place.[3]

[3]Interestingly, recent work of Chu and Viswanathan [8] shows that even well chosen *artificial* pads can also be used to improve a placer’s run-time and solution quality.

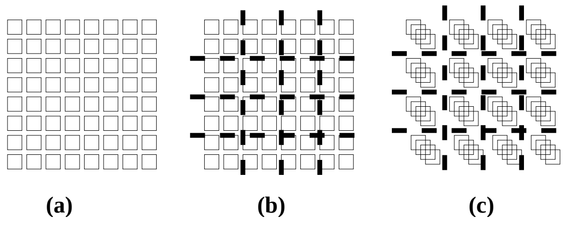


Fig. 14. Sample construction of a PEKO-DP benchmark from an optimal PEKO solution. Cells in an optimal PEKO placement are moved to the centers of uniform bins of user-specified dimensions. (For simplicity, nets are not shown.)

IV. PEKO-DP

The PEKO-DP construction is illustrated in Figure 14. Starting from any optimal PEKO placement, a uniform rectangular bin grid of user-specified dimensions is superimposed. Cells in the same bin are moved to the bin center, where they are placed concentrically (not quite shown in part (c) of the figure). The result is a global placement which is optimal for the given circuit, but only up to the bin size. Because the global placement is used to test the quality of a detailed placer, it is called PEKO-DP. Each PEKO circuit generates multiple PEKO-DP circuits, one for each bin size and shape. By running a detailed placer on such a set of PEKO-DP circuits, the rate of degradation in the quality of the detailed-placement with respect to the bin size is observed.

Experiments are summarized in Figures 15 (square bins) and 16 (nonsquare bins). The PCL algorithm [20] is used by Feng Shui [19], [31]; a variant of it is also used by mPL4 [4], [5], [33]. Dragon's detailed placer [25], [30] requires that bins contain cells in the same standard-cell row only; thus, all bins in Figure 16 have only one row. The results of these experiments show that the quality of detailed placement deteriorates fairly rapidly as the bin size increases, even for these uniformly accurate global placements.

To understand the relative capabilities of given global and detailed placement algorithms on the PEKO circuits, the global placement's displacement from optimal must also be considered. Figure 17 shows the displacement field for the global placement and corresponding detailed placement generated by mPL4 on PEKO 01 (each line segment indicates a cell's displacement from its optimal location). Statistics of cells' displacements from their optimal locations are also listed. It is clear from these figures that, with very few exceptions, cells are placed quite close to their optimal PEKO locations by mPL4's global placer. Ono and Madden observe similar results for Dragon, mPL4, and FengShui [22]. They conclude that, for most algorithms, little room for improvement in global placement exists, and most of the PEKO optimality gap can be attributed to deficiencies in detailed placement [22, page 4].

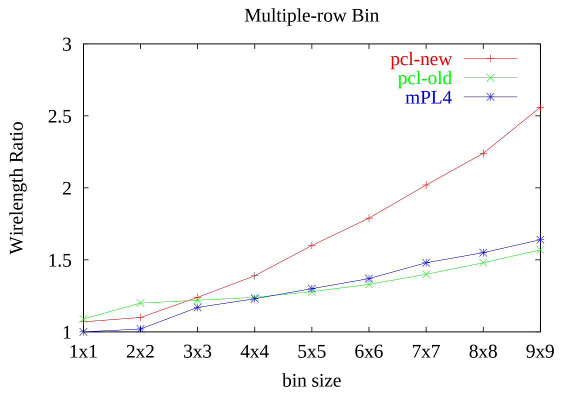


Fig. 15. Average results of PCL-old, PCL-new, and mPL4 on PEKO-DP01–03 with square bins.

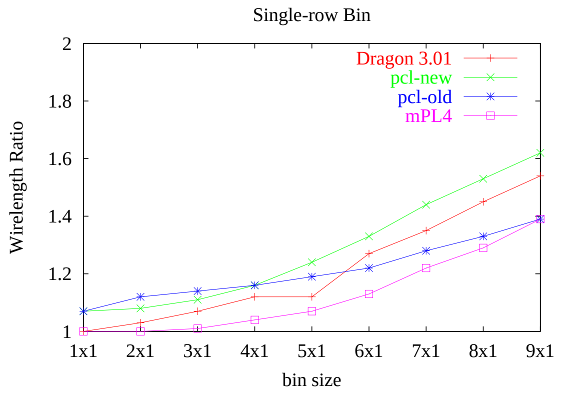
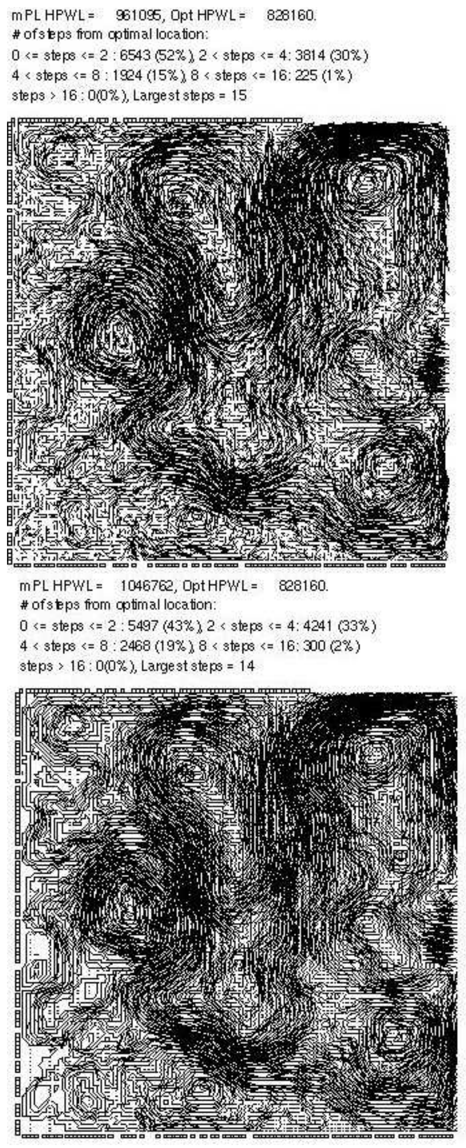


Fig. 16. Average results of Dragon, PCL, and mPL4 on PEKO-DP01–03 with single-row bins.

However, we draw a somewhat different conclusion, for a reason illustrated both in Figure 17 and the example in Figure 18. Examination of the placement in Figure 18 shows that, even when every cell is within just one step of its globally optimal location, no strictly local refinement of cells in this placement will improve the wirelength. That is, no overlap-free optimization on any restricted subset of cells will improve the placement. However, the simple global operation of simultaneously shifting every cell in the chain one step (one “step” equals one cell width plus white space) in the counterclockwise direction brings the placement to the global optimum. While this example might seem contrived, we observe similar structure in the displacement fields of mPL’s placements, both global and detailed, as shown in Figure 17. Similar displacement fields are produced by other tools. From these results, we conclude that, even when each cell in a global placement is very close to (one of) its optimal location(s), further reduction in the objective can often only be achieved by additional global placement, wherein large



```
mPL HPWL =    961095, Opt HPWL =    828160.
# of steps from optimal location:
0 <= steps <= 2 : 6543 (52%), 2 < steps <= 4: 3814 (30%)
4 < steps <= 8 : 1924 (15%), 8 < steps <= 16: 225 (1%)
steps > 16 : 0(0%), Largest steps = 15
```

```
mPL HPWL =    1046762, Opt HPWL =    828160.
# of steps from optimal location:
0 <= steps <= 2 : 5497 (43%), 2 < steps <= 4: 4241 (33%)
4 < steps <= 8 : 2468 (19%), 8 < steps <= 16: 300 (2%)
steps > 16 : 0(0%), Largest steps = 14
```

Fig. 17. Displacement fields generated by mPL4's global (top) and corresponding detailed (bottom) placements on PEKO 01. Line segments indicate the displacement between each cell and its location in a known optimal solution. Statistical summaries of displacement distances are also shown.

subsets of cells can be moved simultaneously, albeit by small amounts. We also infer that techniques based on analytical or force-directed methods [12], [5], [18], [34] will be the most successful in modeling, capturing, and exploiting the apparent "flow" structure in the displacement field exhibited in Figure 17.

V. Conclusion

Three new sets of synthetic benchmark circuits with known optimal placements have been presented and analyzed. The first, PEKO-NU, is a straightforward extension of the original PEKO benchmarks to circuits with non-uniformly sized standard

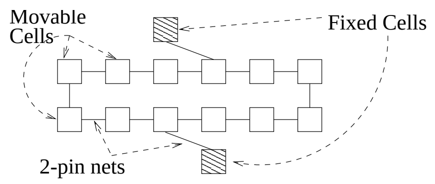


Fig. 18. A nearly optimal global placement which cannot be improved by localized refinement.

cells. The second, PEKO-MC, contains both local and non-local nets and is derived by embedding all its non-local nets within spatially monotone chains of nets between pairs of well separated fixed pads. The third, PEKO-DP, is a simple detailed-placement benchmark constructed by gathering cells in optimal PEKO placements into local bins. Experiments on these new benchmarks yield interesting results. First, standard-cell circuits with all cell widths bounded within a small range are not significantly more difficult to place than similar circuits with all cells of uniform width. Second, the presence of net-wise disjoint chains of nets linking pairs of numerous, well-distributed, fixed I/O pads appears to make placement by contemporary methods considerably less difficult. Circuits designed to ensure the existence of monotone paths for all signals [26], [23], [27], [24] can reasonably be expected to have wirelengths far closer to optimal than what leading placement tools are able to achieve on the original PEKO benchmarks. Third, detailed placement by sequences of local-window optimizations may be ineffective, even when each cell's position in a given global placement is very nearly optimal.

REFERENCES

[1] S. N. Adya and I. L. Markov. Consistent placement of macro-blocks using floorplanning and standard-cell placement. In *Proc. Int'l Symp. on Phys. Design*, pages 12–17, April 2002.

[2] C.J. Alpert. The ISPD98 circuit benchmark suite. In *Proc. Int'l Symp. on Phys. Design*, pages 80–85, 1998.

[3] U. Brenner, A. Pauli, and J. Vygen. Almost optimum placement legalization by minimum cost flow and dynamic programming. In *Proc. Int'l Symp. on Phys. Design*, pages 2–8, 2004.

[4] T.F. Chan, J. Cong, T. Kong, and J. Shinnerl. Multilevel circuit placement. In *Multilevel Optimization in VLSICAD*, chapter 4. Kluwer, Boston, 2003.

[5] T.F. Chan, J. Cong, T. Kong, J. Shinnerl, and K. Sze. An enhanced multilevel algorithm for circuit placement. In *Proc. Int'l Conf. on Computer-Aided Design*, San Jose, CA, 2003.

[6] C. Chang, J. Cong, M. Romesis, and M. Xie. Optimality and scalability study of existing placement algorithms. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, pages 537–549, 2004.

[7] C.C. Chang, J. Cong, and M. Xie. Optimality and scalability study of existing placement algorithms. In *ASPDAC*, pages 325–330, Kitakyushu, Japan, Jan 2003.

[8] C. Chu and N. Viswanathan. FastPlace: Efficient analytical placement using cell shifting, iterative local refinement, and a hybrid net model. In *Proc. Int'l Symp. on Phys. Design*, pages 26–33, April 2004.

[9] J. Cong, G. Nataneli, M. Romesis, and J. Shinnerl. An area-optimality study of floorplanning. In *Proc. Int'l Symposium on Physical Design*, pages 78–83, 2004.

[10] J. Cong, M. Romesis, and M. Xie. Optimality and scalability study of timing-driven placement algorithms. In *Proc. Int. Conf. on Computer Aided Design*, pages 472 – 478, 2003.

[11] K. Doll, F.M. Johannes, and K.J. Antreich. Iterative placement improvement by network flow methods. *IEEE Transactions on Computer-Aided Design*, 13(10), Oct 1994.

[12] H. Eisenmann and F.M. Johannes. Generic global placement and floorplanning. In *Proc. 35th ACM/IEEE Design Automation Conference*, pages 269–274, 1998.

[13] R. Goering. FPGA placement performs poorly, study says. *EE Times*, 2003. `http://www.eedesign.com/story/OEG20031113S0048`.

[14] R. Goering. IC placement benchmarks needed, researchers say. *EE Times*, 2003. `http://www.eedesign.com/story/OEG20030410S0029`.

[15] R. Goering. Placement tools criticized for hampering IC designs. *EE Times*, 2003. `http://www.eedesign.com/story/OEG20030205S0014`.

[16] S.-W. Hur and J. Lillis. Mongrel: Hybrid techniques for standard-cell placement. In *Proc. IEEE International Conference on Computer Aided Design*, pages 165–170, San Jose, CA, Nov 2000.

[17] M. Romesis J. Cong and M. Xie. Optimality, scalability and stability study of partitioning and placement algorithms. In *Proc. International Symposium on Physical Design*, 2003.

[18] A.B. Kahng and Q. Wang. Implementation and extensibility of an analytic placer. In *Proc. Int'l Symp. on Phys. Design*, pages 18–25, 2004.

[19] A. Khatkhate, C. Li, A. R. Agnihotri, S. Ono, M. C. Yildiz, C.-K. Koh, and P. H. Madden. Recursive bisection based mixed block placement. In *Proc. Int'l Symp. on Phys. Design*, 2004.

[20] C. Li and C-K. Koh. On improving recursive bipartitioning-based placement. Technical Report TR-ECE 03-14, Purdue University, 2003.

[21] Q. Liu and M. Marek-Sadowska. A study of netlist structure and placement effi ciency. In *Proc. Int'l Symp. on Phys. Design*, pages 198–203, 2004.

[22] S. Ono and P.H. Madden. On structure and suboptimality in placement. In *Asia South Pacific Design Automation Conf.*, Jan 2005.

[23] R. Otten, and R. Brayton. Planning for performance. In *Proc. of the Design Automation Conference*, pages 122 – 127, 1998.

[24] S. Ramji, and N. Dhanwada. Design topology aware physical metrics for placement analysis. In *Proc. of the Great Lakes Symposium on VLSI*, pages 186 – 191, 2003.

[25] M. Sarrafzadeh, M. Wang, and X. Yang. *Modern Placement Techiques*. Kluwer, Boston, 2002.

[26] W. Gosti, A. Narayan, R. Brayton, and A. Sangiovanni-Vincentelli. Wireplanning in logic synthesis. In *Proc. of the International Conference on Computer-Aided Design*, pages 26 – 33, 1998.

[27] W. Gosti, S. Khatri, and A. Sangiovanni-Vincentelli. Addressing the timing closure problem by integrating logic optimization and placement. In *Proc. of the International Conference on Computer-Aided Design*, pages 224 – 231, 2001.

[28] http://vlsicad.eecs.umich.edu/bk/pdtools/tar.gz/.

[29] `http://vlsicad.eecs.umich.edu/BK/ISPD02bench/`

[30] `http://er.cs.ucla.edu/Dragon/`

[31] `http://vlsicad.cs.binghamton.edu/software.html`

[32] `http://www.public.iastate.edu/˜nataraj/ISPD04_Bench.html`

[33] `http://cadlab.cs.ucla.edu/cpmo/mpl/`

[34] Z. Xiu, J.D. Ma, S.M. Fowler, and R.A. Rutenbar. Large-scale placement by grid warping. In *Proc. Design Automation Conf.*, pages 351–356, June 2004.