Wire Length Prediction based Clustering and its Application in Placement

Bo Hu
Department of Electrical and Computer Engineering
Univ. of California, Santa Barbara, CA 93106, USA
1-805-893-5678
hb@ece.ucsb.edu

Malgorzata Marek-Sadowska
Department of Electrical and Computer Engineering
Univ. of California, Santa Barbara, CA 93106, USA
1-805-893-2721
mms@ece.ucsb.edu

Abstract
In this paper we introduce a metric to evaluate proximity of connected elements in a netlist. Compared to connectivity [8] and edge separability [4], our metric is capable of predicting short connections more accurately. We show that the proposed metric can also predict relative wire length in multi-pin nets. We develop a fine-granularity clustering algorithm based on the new metric and embed it into the Fast Placer Implementation (FPI) framework [10]. Experimental results show that the new clustering algorithm produces better global placement results than the net absorption [10] algorithm, connectivity [8], and edge separability [4] based algorithms. With the new clustering algorithm, FPI achieves up to 50% speedup compared to the latest version of Capo8.5 [19], without placement quality losses.

Categories and Subject Descriptors
J.6 Computer-Aided Engineering - Computer-aided Design (CAD)

General Terms
Algorithms.

Keywords
Placement, Clustering, Wire length prediction

1. Introduction
Due to its importance in automated flow for VLSI designs, the placement problem has been researched for over three decades. As modern VLSI technology advances into nanometer scale, placement becomes even more critical. Whether a design can achieve required objectives, such as performance, routability, and power, is largely determined by the placer's quality. This is so because interconnects dominate nearly all aspects of chip design. A placer decides locations of the cells and thus has a tremendous impact on possible interconnect routes. Interconnect estimation [1][2][7][9][12] attracts a lot of research attention because of the need for early interconnect optimization in the design flow to achieve timing closure. In [2][7][9][12], interconnect lengths are predicted statistically. In [1] wire lengths are estimated individually, based on characterization of typical designs.

Nanometer technology allows us to integrate billions of transistors into a single chip. A modern placer should be able to handle such a design complexity and produce placement solutions meeting the required timing, routability, etc.: It should also do it in a timely fashion to meet the shrinking time-to-market requirements. Due to remarkable advances in circuit partitioning and a need to handle large-scale placement problems, the recently developed placement techniques are mainly based on recursive multi-level partitioning [3][16]. In [10], the authors proposed a Fast Placer Implementation (FPI) framework targeting large-scale placement instances. They applied fine-granularity clustering to reduce the complexity of initial placement problem.

In this paper, we propose a metric to evaluate the proximity of elements connected in a netlist. Compared to the previous connectivity [8] and edge separability [4] metrics, the new metric is capable of predicting short connections more accurately and deciding which groups of nodes should be clustered to achieve good placement. We have developed a deterministic fine-granularity clustering algorithm based on the new metric and integrated it into the Fast Placer Implementation (FPI) framework [10]. Experiments show that the clustering algorithm produces better global placement results than the algorithms based on net absorption [10], connectivity [8], and edge separability [4]. With the new clustering algorithm, FPI achieves up to 50% speedup without placement quality losses compared to the latest version of Capo8.5 [19].

2. Preliminary
A hypergraph \(G(V, E)\) is a graph with a node set \(V\) and a hyperedge set \(E\). Each hyperedge \(i\) in \(E\) is a subset of nodes in \(V\). We denote this subset as \(N(i)\). A node \(v\) is incident to a hyperedge \(i\) if \(v\) is in \(N(i)\). Similarly, a hyperedge (net) \(i\) is incident to a node \(v\) if \(N(i)\) contains \(v\). Degree of a node, denoted by \(d(v)\), is the number of hyperedges incident to \(v\). A degree of a hyperedge \(i\), denoted by \(d(i)\), is the number of nodes incident to \(i\). Each node is associated with an area cost, \(A(v)\).

In this work, each hyperedge is represented by a clique. Traditionally, a weight of an edge in a clique is given by \(EQ1\):

\[
\text{w}(e) = \frac{1}{d(i) - 1}
\]

A connection \(c(u,v)\) represents all the edges connecting the nodes \(u\) and \(v\). The weight of a connection, \(w(u,v)\), is a sum of edge weights of the edges in \(c(u,v)\). The length of a connection, \(l(u,v)\), is a Manhattan distance between \(u\) and \(v\) after placement. A node \(v\) is node \(u\)'s neighbor if the connection \(c(u,v)\) exists. \(W(u)\) denotes the set of all \(u\)'s neighbors.

Contraction measure for a group of nodes quantifies how strongly those nodes are connected in a circuit. We will derive this measure here and validate its efficiency through experiments.

A good placer tends to place strongly contracted nodes in close proximity. This suggests that a group of nodes with strong contraction is likely to be placed together. Intuitively, we think that a group of nodes is strongly contracted if they share many small-fanout nets. Conversely, strongly contracted elements tend to be placed
together in a good placement. A metric that accurately estimates a node group contraction, can predict interconnect lengths among those nodes, and can be used to make correct clustering decisions. In this work, we focus on deriving such a metric and applying it to placement.

3. Mutual contraction

3.1 Analyzing previous results

In [8], Hauck and Borriello introduced connectivity to measure contraction between two connected nodes \( u \) and \( v \). They developed a connectivity-based clustering algorithm and applied it to a partitioning flow showing better cut-size results than previous works [6], [13] et al. EQ2 expresses connectivity in our terminology.

\[
q(c) = \frac{w(u, v)}{A(u)A(v)[d(u) - w(u, v)]}[d(v) - w(u, v)]
\]  
(EQ2)

In [4], edge separability has been proposed to guide the clustering process. The edge separability, \( \lambda(c) \), is defined for each connection \( c(u, v) \) in a graph \( G \). \( \lambda(c) \) is the minimum cut-size among all the cuts separating nodes \( u \) and \( v \) in \( G \). From this definition we infer that the following equation holds:

\[
w(c) \leq \lambda(c) \leq \min\{d(u), d(v)\}
\]  
(EQ3)

Intuitively, the larger \( \lambda(c) \) is, the more contracted \( u \) and \( v \) are, since separating \( u \) and \( v \) causes a greater cut-size penalty. Since computing \( \lambda(c) \) for each connection \( c \) in a graph is very time-consuming, [4] proposes an algorithm to compute a tight lower bound \( q(c) \) for \( \lambda(c) \). For \( q(c) \), the following equation holds:

\[
w(c) \leq q(c) \leq \lambda(c)
\]  
(EQ4)

Finally, [4] ranks all the connections in a graph using the formula in EQ5, performs greedy clustering based on the ranking, and shows better partitioning results than [11][14][15][17].

\[
r(c) = \frac{q(c)}{\min\{d(u), d(v)\}}
\]  
(EQ5)

Our ultimate objective is to find a good metric to measure contractions of all connections. Based on this metric, a larger contraction should imply a shorter length of the connection in the optimal/good placement solution. Different metrics might give different contraction measurements. Since both [8] and [4] show good results on partitioning, we first investigate if connectivity or edge separability are good metrics for our purpose. To this end we have to verify that they are correlated with wire lengths after placement.

We first modify the annealing schedule of our in-house simulated annealing-based standard cell placer and obtain so-called golden placements. We use IBM-place benchmarks [18] in all the experiments in this paper. The golden placement is obtained with an extremely slow annealing schedule. The placement results in terms of total wire length, are at least 10% better than those obtained by the public-domain academic standard cell placers Capo[3] and Dragon[16]. Theoretically, simulated annealing is able to achieve global optimum if enough time is given. For this reason we choose simulated annealing to produce golden placement results, which form the basis for the following discussion.

We use connectivity in EQ2 and edge separability in EQ5 to compute the contraction value for each connection in a hypergraph. Meanwhile, we obtain the lengths of all connections from the golden placement results. In Figure 1, we show the typical length vs. contraction plots for connectivity and edge separability. These results are for the benchmark ibm03. In the plots, x-axis represents contraction while y-axis gives the length of connections. Each data point corresponds to one connection in the graph.

We observe that the data points collected for connectivity measure form several disjoint island-style groups. The number of connections with the same or similar contraction values is quite large, making the metric less accurate for wire length predictions. It is not very surprising to see these island-style groups. In EQ2, \( w(u, v) \) is divided by the following term \( A(u)A(v)[d(u) - w(u, v)]\).

In a typical standard cell netlist, the degree of a cell is usually less than or equal to 4, and the area distribution of standard cells is also quite limited. Furthermore, due to the connectivity of a netlist, the number of connections with the same weight may be large. All these factors contribute to the formation of island-style groups.

We observe that there is a general trend for edge separability, that larger contractions, as measured by \( r(c) \) in EQ5, correspond to shorter connection lengths, but the distribution is not under firm control. The contraction values computed by \( r(c) \) are not well correlated with interconnect lengths.

3.2 Intrinsic contraction of a connection

To motivate our derivation of a good contraction metric, we start with an example. In Figure 2, we show two nodes, \( u \) and \( x \), and their neighbors. For clarity, in all the following figures, we draw multi-pin nets as stars instead of cliques. The node \( u \) has four neighbors connected via a 5-pin net and one neighbor via a 2-pin net, while node \( x \) connects to its neighbors via 2-pin nets. We ask whether connections like \( (u,v) \) and \( (x,y) \) are treated the same by placers. In Figure 2, \( i \) and \( j \) are both 2-pin nets. It is known that placers minimizing total wire lengths try hard to optimize small-fanout nets. Such nets constitute a majority of all the nets and are more sensitive to locations of the nodes. Therefore, we expect that a placer will try to shorten the length of \( (u,v) \) and \( (x,y) \). But the question is whether a connection like \( (u,v) \) really is treated the
same as a connection like \((x, y)\). In the following, we first show that a placer does indeed treats \((u, v)\) and \((x, y)\) differently but neither connectivity nor edge separability based metrics reflect this. Next, we discuss our new metric, which is able to capture this difference. From the golden placement results, we collect all the nodes whose degree is two, and one of the two incident nets is a 2-pin net. We denote the incident 2-pin net as \(\alpha\) and the other net as \(\beta\). We ask how \(\beta\) affects the length of \(\alpha\). Figure 3 shows the plot of the \(\alpha\)-nets length distribution over the \(\beta\)-nets degree for all the nodes we collected. The y-axis gives the lengths of \(\alpha\)-nets and the x-axis shows the degrees of \(\beta\)-nets. We observe that the larger the degree of \(\beta\), the shorter the \(\alpha\)-net length in the golden placement. Based on this observation, we conclude that in Figure 2, connection \((u,v)\) is expected to be shorter than \((x,y)\) after placement.

![Fig. 3: length distribution of 2-pin nets](image)

The observation seems also intuitively plausible. As mentioned before, a placer puts more optimization effort on smaller-fanout nets. So small-fanout nets tend to have stronger contraction than large-fanout ones. The final locations of the nodes can be thought of as an effect of contractions of all the connections. To achieve short length, a connection has to compete with other connections under the physical constraints of non-overlapping nodes. In Figure 2, the connection \((u,v)\) is competing with a 5-pin net while \((x,y)\) is competing with another 2-pin net. Intuitively it is obvious that \((u,v)\) is more likely to win this competition and be shorter than \((x,y)\).

We can also view this problem from the traditional force-directed placement perspective, as for example described in [5]. In force-directed placement approach, nodes are modeled as particles and nets as strings. The placement of particles is determined by the contraction forces of the strings. Small-fanout nets usually exert stronger contraction forces on their connected nodes than large-fanout nets do.

This is in contrast to connectivity as computed from EQ2. If nodes \(v\) and \(y\) have the same degrees and \(u, v, x, y\) are of similar size, \(c_e(u, v)\) is the same or very close to \(c_e(x, y)\). In other words, connectivity does not provide enough resolution to distinguish between \((u,v)\) and \((x,y)\). In case of edge separability, suppose \((u,v)\) and \((x,y)\) are the min-cuts separating \(u\) from \(v\) and \(x\) from \(y\), respectively. It is clear that in such a case, \(\lambda(u, v)\) is equal to \(\lambda(x, y)\) since \(m\) and \(n\) are both 2-pin nets.

To provide a finer resolution for situations similar to that discussed above, we first redefine the weight of the edges in a clique formed by a net \(i\) as follows:

\[
 w'(e) = \frac{2}{(d(i) - 1)d(i)} \tag{EQ6}
\]

Compared to the classical edge weight in a clique as stated in EQ1, we reduce the weight by a factor of \(\frac{d(i)}{2}\). The weight of a connection is computed as before, using the new weights \(w'(e)\).

A node incident to a net \(i\) of degree \(d\) has \(d - 1\) edges of weight \(w'(e)\) connecting to the other nodes in \(i\). Therefore, such a node contributes to the other nodes in \(i\), a total weight of \(\frac{2}{d}\). This means that the larger the degree of a net is, the less weight it contributes to its connected nodes. In Figure 2, \(u\) connects to four neighbors via a 5-pin net \(m\). The net \(m\) introduces 4 connections incident to \(u\). Each connection has a weight \(\frac{2}{(5 - 1) \cdot 5} = 0.1\) computed from EQ6. The total weight on 4 connections incident to \(u\) and introduced by \(m\) is 0.4.

To capture the relative weight of connections incident to a node \(u\), we define the relative weight of a connection as follows:

\[
 w_r(u, x) = \frac{w'(u, x)}{\sum_x w'(u, x)} \tag{EQ7}
\]

In EQ7, \(w'(u, x)\) is a weight of the connection \((u, x)\); the summation is over all nodes \(x\) adjacent to \(u\).

For example, referring to Figure 2, \(w_r(u, v) = \frac{1}{1 + 0.4} = 0.71\), while \(w_r(u, x) = \frac{1}{1 + 1} = 0.5\). Since \(w_r(u, v)\) is larger than \(w_r(u, x)\), it indicates that the connection \((u,v)\) plays a bigger role in determining the placement of node \(u\) than the connection \((x,y)\) does in determining the placement of node \(x\).

**Definition 1:** The best neighbor of a node \(u\) is the node incident to \(u\), \(v_b\), which has the largest \(w_r(u, v)\) among all the incident nodes.

**Definition 2:** A node pair \((x, y)\) is a best neighbor pair if \(x\) is \(y\)'s best neighbor and \(y\) is \(x\)'s best neighbor.

EQ7 takes a node-centric point of view. It can be used to indicate which connection/neighbor has the strongest impact on the placement of the center node, but as the following example shows, this might not be sufficient to predict the placement of a node.

In Figure 4, we show two connections \((u,v)\) and \((x,y)\). From EQ7, \(w_r(u, v) = w_r(x, y) = 0.71\); and \(v\) is the best neighbor of \(u\) and \(y\) is the best neighbor of \(x\). But in the figure, node \(v\)'s best neighbor is node \(q\). It indicates that \(v\) is likely to be placed closer to \(q\) instead of to \(u\). In contrast, \(x\) and \(y\) form a best-neighbor pair. As a result, a placer is more likely to put \(x\) and \(y\) together.

This observation suggests that we should rely on mutual connectivity relationship among nodes in order to predict their relative placement.

**Definition 3:** A mutual contraction of a connection \((x,y)\) is a tuple \((w_r(x,y), w_r(y,x))\).

For example, if \(x\) and \(y\) form a best-neighbor pair, then both \(w_r(x,y)\) and \(w_r(y,x)\) are large, and the connection \((x,y)\) is likely...
In Figure 6, we show two 3-pin nets, \( \text{net } i \) and \( \text{net } j \), and two node groups: \( U_1(1, 2, 3) \) and \( U_2(4, 5, 6) \). If we compute \( c_g(U_1) \) and \( c_g(U_2) \) based on EQ9 and EQ10, we can see that \( c_g(U_1) = 0.4^2 = 0.16 \) and \( c_g(U_2) = 0.5^2 = 0.25 \). Obviously, \( c_g(U_2) > c_g(U_1) \). It means that, by our metric, the group \( U_2 \) has a stronger contraction than \( U_1 \). This is so because in Figure 6, net \( i \) has to compete with three 2-pin nets to be short, whereas net \( j \) does that with three 3-pin nets which have less contraction than the 2-pin nets. Since a placer usually places strongly contracted nodes, or nodes with strong contraction, in a close proximity, we predict that the net \( j \) will be shorter than \( i \).

Fig. 6: Mutual contraction for a group

4. Mutual contraction-based clustering algorithm

In section 3.2, we have established a good correlation between the contraction of a connection \((x, y)\), measured by the mutual contraction metric, and the length of \((x, y)\). Especially, as shown in Figure 5, the connections with strong contraction most likely will end up having very short lengths. It is thus possible to develop a deterministic clustering algorithm exploiting the good predictability of the new metric. We will use a pass-based pair-wise clustering strategy. At each pass, we only visit each node once. A priority queue is created for connections in the hypergraph. A connection with the largest contraction is on the top of the queue. We pick the connection \((x, y)\) from the top of the queue to see whether grouping \( x \) and \( y \) is feasible or not. Grouping \( x \) and \( y \) is called feasible if the grouping does not violate cluster size constraint and both \( x \) and \( y \) are unvisited in the current pass. If the grouping is not feasible, we discard this connection and continue the process. Otherwise, we create a new node \( z \) which represents both \( x \) and \( y \), mark \( x \) and \( y \) visited, and update the connectivity information for \( z \). The cluster size constraint is given before clustering. In this work, we perform fine-granularity clustering as in [10]. Each fine cluster consists of 2-3 nodes of average size. We also set a target cluster number. The clustering process terminates when no more clustering is possible due to the size constraints or when the cluster number limit has been reached. In general, since we only create fine-grained clusters, 2 or 3 passes are enough to finish the clustering process.

5. Experiments

We implemented the new clustering algorithm in the FPI [10] framework. The experiments are conducted on 1GHz Pentium 4 Intel processor and the operating system GNU/Linux Mandrake 9.0. We obtained the IBM-Place benchmarks from [18] with no channels between standard cell rows. Sizes of the benchmarks used in the experiments range from 22207 to 70558 nodes.

5.1 Connection length prediction

We evaluate the mutual contraction based metric \( c_g(x, y) \) by comparing it with the edge separability[4] and connectivity metrics[8] in the following experiment. For each metric, we sort all the connections in descending order based on our contraction value and compute the total length of the first \( N \) connections. We collect the first \( N \) shortest connections predicted by each metric and add them up as total length \( L \) of those connections.

In Figure 7, we show a typical \( L \) vs. \( N \) plot for each metric using the golden placement results. The x-axis represents \( N \), the number of connections, while y-axis gives \( L \), the total length of connections. Since we are dealing with a fine-granularity clustering problem, it is meaningful to look only at the connections with strong contraction. So we set the largest \( N \) to be roughly the number of nodes in the hypergraph. It can be seen that the first \( N \) connections with the
largest contraction identified by mutual contraction are shorter than those by edge separability and connectivity. We observe also that connectivity predicts short wires better than edge separability, although it is shown in [4] that edge separability produces better partitioning results than the connectivity based algorithm [8].

5.2 Multi-pin net length prediction

In section 3.3, we derived $c_g(U)$ to evaluate the contraction of a node group. Here we describe an experiment establishing correlation between $c_g(U)$ and the length of multi-pin nets. We collect all the nets of degree $d$ and compute contraction $c_g$ for each node group formed by them. We also obtain the net lengths from the golden placement. Net lengths are estimated by half-perimeter of their bounding boxes. In Figure 8, we show length vs. $c_g$ plots for all 3-pin and 4-pin nets in the benchmark ibm07. Each point in the plots represents one net. We observe that $c_g$ demonstrates a good correlation with net length. Especially, the nets with large $c_g$ are most likely to be short, and long wires are likely to have small $c_g$.

5.3 Comparing clustering algorithms

In this experiment, we compare our mutual contraction-based clustering algorithm with those connectivity-based [8] and net-absorption-based algorithms [10]. The algorithm in [8] visits nodes in a random order and always clusters the neighbor with the largest connectivity into the currently visited node, whereas the algorithm in [10] tries to maximize the net absorption based cost function. We have implemented the original algorithm in [8], and its deterministic version which always picks the connection with the largest connectivity for clustering. To compare with edge separability [4] (since we don’t have their implementation details), we implemented a deterministic version of that algorithm, always picking the connection with the largest rank. The experiments were conducted in the Fast Placer Implementation (FPI) framework [10]. FPI flow consists of 3 stages: (S1) fine-granularity clustering; (S2) global placement of the clustered netlist; and (S3) fast simulated annealing-based refinement. We applied each algorithm at stage 1 and compared the total wire length results after stage 2 (global placement), since we do not want simulated annealing-based refinement to shield the differences at stage 1.

In Table 1, we denote by rcon, FGC, and MC the original connectivity [8], net absorption [10], and mutual contraction-based algorithms, respectively. We also denote by dcon and ESC the deterministic connectivity and edge-separability-based algorithms we implemented. We observe that dcon performs much better than rcon, probably because of the random clustering process. The algorithm in [8] randomly visits nodes in a graph and picks the neighbor with maximum connectivity for clustering. As a result, it may choose some connections with relatively small contractions. Not surprisingly, among the five algorithms, Mutual Contraction gives the overall best result. As shown in Figure 7, MC can predict better short connections and thus has a better chance to make correct clustering decisions. It is interesting that ESC produces slightly better (1%) wire length results than dcon although connectivity shows better short wire prediction in Fig 7. We conjecture that there are two possible reasons: (1) the results given in Table 1 are from FPI flow while those in Fig 7 are based on golden placement results which are at least 10% better. Thus, it is possible that current FPI flow might not take full advantage of a good clustering solution. (2) Since placement quality from FPI is mainly decided by the second stage, min-cut based global placer, ESC based clustering, which aims at min-cut objective, might be more suitable than that based on simple connectivity.

<table>
<thead>
<tr>
<th>bench</th>
<th>rcon</th>
<th>dcon</th>
<th>ESC</th>
<th>FGC</th>
<th>MC</th>
</tr>
</thead>
<tbody>
<tr>
<td>ibm03</td>
<td>12.6</td>
<td>0.77</td>
<td>0.80</td>
<td>0.83</td>
<td>0.76</td>
</tr>
<tr>
<td>ibm04</td>
<td>14.9</td>
<td>0.88</td>
<td>0.89</td>
<td>0.92</td>
<td>0.86</td>
</tr>
<tr>
<td>ibm05</td>
<td>39.7</td>
<td>0.89</td>
<td>0.90</td>
<td>0.93</td>
<td>0.86</td>
</tr>
<tr>
<td>ibm06</td>
<td>17.0</td>
<td>0.94</td>
<td>0.92</td>
<td>0.97</td>
<td>0.90</td>
</tr>
<tr>
<td>ibm07</td>
<td>418</td>
<td>0.90</td>
<td>0.89</td>
<td>0.93</td>
<td>0.86</td>
</tr>
<tr>
<td>ibm08</td>
<td>430</td>
<td>0.91</td>
<td>0.90</td>
<td>0.94</td>
<td>0.90</td>
</tr>
<tr>
<td>ibm09</td>
<td>366</td>
<td>0.92</td>
<td>0.90</td>
<td>0.96</td>
<td>0.88</td>
</tr>
<tr>
<td>ibm10</td>
<td>710</td>
<td>0.90</td>
<td>0.88</td>
<td>0.95</td>
<td>0.87</td>
</tr>
<tr>
<td>ibm11</td>
<td>550</td>
<td>0.92</td>
<td>0.92</td>
<td>0.94</td>
<td>0.90</td>
</tr>
<tr>
<td>Ave</td>
<td>1.00</td>
<td>0.89</td>
<td>0.88</td>
<td>0.93</td>
<td>0.86</td>
</tr>
</tbody>
</table>

TABLE 1. Global placement comparison
In this paper, we have proposed a new metric to measure the contraction of the connections in the netlist. We show that the new metric can be used to predict relative interconnect lengths, and that it performs very well in a deterministic clustering algorithm. We demonstrate that the fine-grain clustering based on the contraction metric improves the performance of the state of the art standard cell placer without quality losses.

### 5.4 Comparison with Capo8.5[19]

In FPI 1.0[10], simulated annealing-based refinement has been applied at stage 3 to recover the placement quality loss caused by stage 1. We implemented FPI 2.0 by integrating Mutual Contraction-based clustering algorithm at stage 1. In Table 1 we have already shown that a mutual contraction based clustering algorithm performs better than the FGC algorithm[10] because of its ability to predict short connections, so we leave stage 3 as an option in FPI 2.0.

In this experiment, we compare FPI 2.0 to Capo8.5, which is the latest release of[19]. In FPI 2.0, we use Capo8.5 at stage 2. By doing this, we want to demonstrate that integrating an existing fast placer into our framework boosts performance of the placer without quality losses. We attribute this improvement to the fact that our metric can accurately predict which connections will be short in a good placement. Clustering the nodes on short connections assures that the placement quality does not suffer.

In Table 2, we report the total wire length in meters (\(wl\)) achieved by Capo8.5, and state the corresponding CPU run times in seconds. FPI 2.0 results are normalized with respect to those of Capo. We observe that FPI usually takes only a half of Capo’s cpu time to get comparable total wire lengths. By finding a good metric to predict and then cluster the very short connections in the netlist, the placement problem becomes smaller and the performance of the existing placer can be dramatically improved.

### 6. Conclusions

In this paper, we have proposed a new metric to measure the contraction of the connections in the netlist. We show that the new metric can be used to predict relative interconnect lengths, and that it performs very well in a deterministic clustering algorithm. We demonstrate that the fine-grain clustering based on the contraction metric improves the performance of the state of the art standard cell placer without quality losses.

<table>
<thead>
<tr>
<th>bench</th>
<th>Capo8.5</th>
<th>FPI 2.0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(wl)</td>
<td>(cpu)</td>
</tr>
<tr>
<td>ibm03</td>
<td>10.1</td>
<td>163</td>
</tr>
<tr>
<td>ibm04</td>
<td>13.1</td>
<td>218</td>
</tr>
<tr>
<td>ibm05</td>
<td>35.0</td>
<td>299</td>
</tr>
<tr>
<td>ibm06</td>
<td>14.9</td>
<td>278</td>
</tr>
<tr>
<td>ibm07</td>
<td>370</td>
<td>346</td>
</tr>
<tr>
<td>ibm08</td>
<td>395</td>
<td>383</td>
</tr>
<tr>
<td>ibm09</td>
<td>327</td>
<td>408</td>
</tr>
<tr>
<td>ibm10</td>
<td>632</td>
<td>534</td>
</tr>
<tr>
<td>ibm11</td>
<td>489</td>
<td>558</td>
</tr>
<tr>
<td>Ave</td>
<td>1.00</td>
<td>1.00</td>
</tr>
</tbody>
</table>

**TABLE 2.** placement comparison

In Table 1, we didn’t show CPU time results for the clustering algorithms because they are of the same order of magnitude and are marginal compared to that consumed by placement.

### 7. Acknowledgment

This work was supported by MARCO/DARPA Gigascale Silicon Research Center (GSRC).

### Reference


