A study of routability estimation and clustering in placement

Kalliopi Tsota, Cheng-Kok Koh and Venkataramanan Balakrishnan
School of Electrical and Computer Engineering, Purdue University
West Lafayette, IN, 47907-2035
{ktota,chengkok,ragu}@ecn.purdue.edu

ABSTRACT
This paper studies the effects of clustering as a pre-processing step and routability estimation in the placement flow. The study shows that when clustering and routability estimation are considered, the placer effectively improves the routed wirelength for the circuits of IBM-PLACE 2.0 standard-cell Benchmark Suite [1] and results in the best average routed wirelength when compared against state-of-the-art academic placers.

1. INTRODUCTION
Modern circuits contain millions of components; the placement of the components in a circuit has a strong impact on the performance of the circuit. This work focuses on routability-driven analytical placement. The routed wirelength is estimated during the global placement stage and helps guide the placement process. At the same time, a deterministic hypergraph clustering algorithm reduces the size of the circuit netlist and improves the scalability of the placement algorithm.

Clustering is performed during the first level of the placement framework, where small clusters that typically contain three to five standard-size cells are formed. The placer is then applied to the clustered netlist to minimize the overlap among the clusters. At the second level, the clustered netlist is de-clustered and the cells of the original netlist are assigned to the locations obtained from the first level. The placer minimizes the overlap of the cells and the estimated routed wirelength becomes part of the objective function. The final placement solution is legalized and detailed placed. Then, it is routed by a commercial router.

The main contributions of this paper are: (i) A method for the estimation of routed wirelength in a region; (ii) A deterministic clustering algorithm that clusters nets, rather than cells, based on a more global view of the circuit netlist; (iii) The application of clustering to the placement, in order to obtain higher quality placement solutions; and (iv) A study of the relative importance of clustering and routed wirelength estimation inside the placement framework.

Experimental results show that with routability estimation, the relative gain in the performance of the placer is 0.51% over the baseline software. With clustering, the relative gain in the performance of the placer is 6.21% over the baseline software. Moreover, with both routability estimation and clustering, the relative gain in the performance of the placer is 6.76% over the baseline software.

The placer improves the routed wirelength by 8.62% on average when compared against the placement tools that give the best published results for circuits of the IBM-PLACE 2.0 standard-cell Benchmark Suite [1]. More specifically, the placer improves the routed wirelength by 15.18%, 10.44%, 5.79%, 2.67%, 14.10% and 1.30% when compared against mPL-R+WSA [2], mPL6 [3], CHKS+WSA [4], RUDY [5], APlace2.0 [6] and the placer described in [7].

2. BACKGROUND
Clustering for placement achieves affordable placement runtime. Typically, the clustering algorithms are categorized into: (i) Persistent clustering where the clusters are formed at the beginning of the placement and are further refined only at the end in order to reduce cell overlap [8, 9], and, (ii) Multilevel clustering where clustering is performed repeatedly during placement. Multilevel placers are reported in [10, 3, 11].

The algorithms that have been developed to model routability during placement can be separated into the following two categories: (i) Algorithms that estimate routability during the global placement and include the model of routability into the objective of the placement in order to minimize congestion [12, 13, 14], and, (ii) Algorithms that estimate routability at the end of the global placement and apply post-processing techniques to reduce congestion [15, 5, 16].

3. ROUTABILITY-DRIVEN PLACEMENT
The wire density of a net is defined as
\[
d_n(x, y) = \frac{HPWL_n}{Area_n},
\]
where
\[
HPWL_n = \max_{i \in e} \{x_i\} - \min_{i \in e} \{x_i\} + \max_{i \in e} \{y_i\} - \min_{i \in e} \{y_i\}
\]
is the HPWL of the bounding box of the net \(e\), and
\[
Area_n = (\max_{i \in e} \{x_i\} - \min_{i \in e} \{x_i\}) \cdot (\max_{i \in e} \{y_i\} - \min_{i \in e} \{y_i\})
\]
is the area of the bounding box of the net [5].
Let $Sd_g(x,y)$ denote the estimated routed wirelength inside the bin and $avg_d_g(x,y)$ the average estimated routed wirelength. The expressions of $Sd_g(x,y)$ and $avg_d_g(x,y)$ are given in [7]. Let $D_g(x,y)$ denote the cell potential function. Define $D_g(x,y)$ as a function of the estimated wirelength inside bin $g$ as follows

$$D_g(x,y) = \alpha \cdot (Sd_g(x,y) - D) + \beta \cdot (Sd_g(x,y) - avg_d_g(x,y))$$

where $\alpha$ and $\beta$ are user-defined constants. Note that the formulation in APPlace2.0 [17] defines $D_g(x,y)$ as a constant, while in this work $D_g(x,y)$ is a function of the vector of cell coordinates. The formulation for $D_g(x,y)$, as given in (2), is different from that in [7].

The technique for the estimation of wirelength inside the bin is based on the fact that the wirelength of a multi-pin net at the interior of a bin is determined from the overlapping area between the two and the wire density of the net. If the bounding box of a net overlaps with a region, it adds to the total wirelength within the region by a factor equal to the product of the wire density of the net and the overlapping area between the region and the bounding box of the net.

The constrained optimization problem is

$$\min WL(x,y)$$

subject to $SD_g(x,y) = D_g(x,y), \forall g$. 

The problem is solved as a sequence of unconstrained optimization problems

$$\min WL(x,y) + \frac{1}{2 \cdot u} \cdot \sum_g (SD_g(x,y) - D_g)^2,$$

where $u$ monotonically decreases in the sequence.

To solve the optimization problem, the publicly available quasi-Newton solver with boundary constraints [18] is used. The implementation of the solver is in the default mode and requires a smooth approximation of the objective function. Instead of using the popular log-sum-exp (LSE) [19] approximation for the wirelength, the authors make use of the two-variable CHKS function [20] that is called recursively to approximate the max value. Instead of using the log-sum-exp (LSE) approximation for the wirelength, the authors make use of the Bell shaped function as in APlace [17].

4. CLUSTERING

The proposed algorithm makes use of large (net degree greater than 90% of the maximum net degree) nets in the circuit to form initial clusters. Each large net forms an initial cluster that contains all cells located inside the net as well as their neighboring cells. The number of initial clusters is equal to the number of large nets. Also, note that a cell may appear in two or more initial clusters. A modified version of FM algorithm [21] is then used to refine the initial clusters. Again, the refined clusters may have overlaps. To remove the overlaps, we assign to each refined cluster a cluster score

$$cl_{sc}(i) = cl_{area}(i) \cdot cl_{nets}(i) \cdot cl_{cell}(i) \cdot sh_{cell}(i),$$

where $cl_{cell}(i)$ is the number of clustered cells inside cluster $i$, $cl_{nets}(i)$ is the number of clustered nets inside cluster $i$, $cl_{area}(i)$ is the area of cluster $i$, $c_{nets}(i)$ is the number of cut nets between cluster $i$ and the remaining refined clusters and $sh_{cell}(i)$ is the number of shared cells between cluster $i$ and the remaining refined clusters.

For each net of the circuit, the net score is initialized to zero and a net score $net_{sc}(j)$ is calculated as

$$net_{sc}(j) = \sum_{i \in cl_j} (cl_{sc}(i))^2 - \sum_{i \in cut_j} (cl_{sc}(i))^2$$

where the net score depends only on the refined clusters that contain net $i$, either entirely or partially [22]. If a refined cluster contains net $i$ entirely, i.e. net $i$ is a clustered net for the cluster, the cluster score is added to the net score. If a refined cluster contains net $i$ partially, i.e. net $i$ is a cut net for the cluster, the score of the cluster is subtracted from the score of net $i$.

The nets are ranked based on their score and for those nets that have a positive score, the following cases are considered [22]: (i) If the net has all its cells unclustered and the area of these cells is smaller than the area constraint, we form a new cluster and place all these cells inside the cluster; (ii) If the net has at least one of its cells clustered and all the clustered cells belong to the same cluster, we examine if the area constraint is satisfied and place all unclustered cells of the net inside the existing cluster; and (iii) If the net has at least two of its cells clustered and they belong to more than one clusters, we examine if the area constraint is satisfied and place all existing clusters as well as unclustered cells into a new large cluster.

5. EXPERIMENTAL RESULTS

The placement algorithm presented in the paper is performed in a two-level framework. The initial clustering of the netlist is obtained from the deterministic hypergraph clustering algorithm. Each cluster is shaped as a square, with its area being the sum of the areas of the cells that belong to the cluster and all cells confined inside the cluster area. We used 0.5 for the user-defined constants $\alpha$ and $\beta$. The wirelength estimation technique is incorporated in the global placement and applied in the second level of the two-level framework. The final placement is legalized using the detailed placer described in mPL-R+WSA [2] and routed using a commercial router. All experiments are performed on an Intel Core2 DUO at 2.66GHz.

5.1 Effects of clustering and routability estimation

Four different variations of the placement algorithm were compared against each other in terms of routability and efficiency: (i) Neither routability estimation nor clustering is considered; (ii) Routability estimation is considered and clustering is not; (iii) Routability estimation is not considered and clustering is; and (iv) Both routability estimation and clustering are considered.

In Table 1, under column “Baseline Software”, “Routability Estimation”, “Clustering” and “Proposed Placer (Routability Estimation + Clustering)” we report the results of cases (i), (ii), (iii), and (iv), respectively. For each case, the first column, labeled “r-WL”, shows the routed wirelength on each benchmark after routing the placed design. A “*” next to the number means that there are some routing violations. The second column, labeled “p/r-CPU”, reports the total runtimes for placement and routing. The total CPU time for placement includes clustering, global placement, legalization and detailed placement.

As shown in Table 1, the relative gain in the performance,
### Table 1: Routability Results on IBM vs. 2 Standard Cell Placement Benchmark Suite

<table>
<thead>
<tr>
<th>Baseline Software</th>
<th>Routability Estimation</th>
<th>Clustering</th>
<th>Proposed Places (Wirelength Estimation + Clustering)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>r-WL (m)</td>
<td>p/r-CPU (s)/(s)</td>
<td>r-WL (m)</td>
</tr>
<tr>
<td>ibm01e</td>
<td>0.69</td>
<td>106/2384</td>
<td>0.69</td>
</tr>
<tr>
<td>ibm01h</td>
<td>0.69</td>
<td>108/8607</td>
<td>0.69</td>
</tr>
<tr>
<td>ibm02e</td>
<td>1.84</td>
<td>188/596</td>
<td>1.82</td>
</tr>
<tr>
<td>ibm02h</td>
<td>1.90</td>
<td>178/4150</td>
<td>1.89</td>
</tr>
<tr>
<td>ibm07c</td>
<td>3.76</td>
<td>417/828</td>
<td>3.74</td>
</tr>
<tr>
<td>ibm07h</td>
<td>3.89</td>
<td>393/572</td>
<td>3.86</td>
</tr>
<tr>
<td>ibm08e</td>
<td>3.81</td>
<td>605/339</td>
<td>3.84</td>
</tr>
<tr>
<td>ibm08h</td>
<td>4.02</td>
<td>496/415</td>
<td>4.01</td>
</tr>
<tr>
<td>ibm09e</td>
<td>3.34</td>
<td>735/472</td>
<td>3.41</td>
</tr>
<tr>
<td>ibm09h</td>
<td>3.04</td>
<td>677/490</td>
<td>3.02</td>
</tr>
<tr>
<td>ibm10c</td>
<td>5.90</td>
<td>930/601</td>
<td>5.86</td>
</tr>
<tr>
<td>ibm10h</td>
<td>5.87</td>
<td>972/815</td>
<td>5.85</td>
</tr>
<tr>
<td>ibm11e</td>
<td>4.45</td>
<td>829/965</td>
<td>4.42</td>
</tr>
<tr>
<td>ibm11h</td>
<td>4.59</td>
<td>854/830</td>
<td>4.57</td>
</tr>
<tr>
<td>ibm12e</td>
<td>8.80</td>
<td>806/1084</td>
<td>8.81</td>
</tr>
<tr>
<td>ibm12h</td>
<td>8.90</td>
<td>708/1924</td>
<td>8.88</td>
</tr>
<tr>
<td>% improv.</td>
<td></td>
<td>0.51</td>
<td></td>
</tr>
</tbody>
</table>

i.e. average routed wirelength, of the placer is 0.51% over the baseline software when only routability is considered. At the same time, the relative gain in the performance of the placer is 6.21% over the baseline software when only clustering is considered. Moreover, when both routability estimation and clustering are considered, the relative gain in the performance of the placer is 6.76% over the baseline software.

#### 5.2 Comparison with other placement tools

Table 2 shows the results obtained from the placer for the circuits of IBM-PLACE 2.0 standard-cell Benchmark Suite [1]. For each placement tool, the first column, named “r-WL”, shows the routed wirelength on each benchmark after routing the placed design. A “∗” next to the number means that there are some routing violations. Note that the results presented for RUDY [5] were routed using a different routing tool than the one used in this paper. The results reported for RUDY in Table 2 have better wirelengths than those in [5]. The second column, labeled “p/r-CPU”, reports the total runtimes for placement and routing.

Compared with [7], the placer achieves a better average routed wirelength. At the same time, the placer described in [7] makes use of the randomized algorithm hMetis [8], whereas the proposed placer is based on deterministic clustering. Measuring the quality of the routing solution, the placer combining wire density and clustering results in the best routed wirelength compared to all other placers. Specifically, the proposed placer obtains a reduction in the routed wirelength by 15.18%, 10.44%, 5.79%, 2.67%, 14.10% and 6.76% when compared against mPL-R+WSA [2], mPL6 [3], CHK5+WSA [4], RUDY [5], APlace2.0 [6] and the placer described in [7].

#### 6. CONCLUSION

The paper proposes a global placement flow guided by an efficient routed wirelength estimation method. The placement is performed in a two-level framework in which the size of the original circuit netlist is reduced using an efficient deterministic hypergraph clustering algorithm. When combined with both clustering and routability estimation, the placer obtains the best average routed wirelength in the circuits of IBM-PLACE 2.0 standard-cell Benchmark Suite [1]. The routability estimation is shown to improve the quality of the placement solution, while the clustering reduces the placement runtime.

#### 7. ACKNOWLEDGMENT

This research was supported in part by SRC (Task ID 1822.001).

#### 8. REFERENCES


Table 2: Routability Results on IBM vs.2 Standard Cell Placement Benchmark Suite

<table>
<thead>
<tr>
<th></th>
<th>mPL-R+WSA</th>
<th>mPLB</th>
<th>CHRS+WSA</th>
<th>RDFY</th>
<th>APace2.0</th>
<th>Placed in [7]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>r-WL (m)</td>
<td>p/r-CPU (s)/(s)</td>
<td>r-WL (m)</td>
<td>p/r-CPU (s)/(s)</td>
<td>r-WL (m)</td>
<td>p/r-CPU (s)/(s)</td>
</tr>
<tr>
<td>ibm10e</td>
<td>0.77</td>
<td>129/3517</td>
<td>0.84*</td>
<td>56/12607</td>
<td>0.71</td>
<td>143/1500</td>
</tr>
<tr>
<td>ibm10h</td>
<td>0.74*</td>
<td>125/10032</td>
<td>0.77*</td>
<td>80/35006</td>
<td>0.75</td>
<td>137/3928</td>
</tr>
<tr>
<td>ibm10c</td>
<td>2.08</td>
<td>237/462</td>
<td>2.05</td>
<td>111/483</td>
<td>1.88</td>
<td>238/369</td>
</tr>
<tr>
<td>ibm02h</td>
<td>2.35*</td>
<td>286/1702</td>
<td>2.04</td>
<td>106/5855</td>
<td>1.95</td>
<td>233/227</td>
</tr>
<tr>
<td>ibm07e</td>
<td>3.85</td>
<td>505/490</td>
<td>3.97</td>
<td>257/7495</td>
<td>3.86</td>
<td>506/480</td>
</tr>
<tr>
<td>ibm07h</td>
<td>4.00</td>
<td>491/591</td>
<td>4.06*</td>
<td>254/4584</td>
<td>3.65</td>
<td>538/1831</td>
</tr>
<tr>
<td>ibm08e</td>
<td>4.47</td>
<td>744/1020</td>
<td>4.24</td>
<td>315/6664</td>
<td>3.89</td>
<td>638/649</td>
</tr>
<tr>
<td>ibm08h</td>
<td>4.20</td>
<td>736/717</td>
<td>4.20</td>
<td>301/5951</td>
<td>3.89</td>
<td>743/583</td>
</tr>
<tr>
<td>ibm09e</td>
<td>3.66</td>
<td>508/418</td>
<td>3.05</td>
<td>304/401</td>
<td>3.01</td>
<td>723/437</td>
</tr>
<tr>
<td>ibm09h</td>
<td>4.01</td>
<td>561/466</td>
<td>3.05</td>
<td>286/447</td>
<td>3.10</td>
<td>625/401</td>
</tr>
<tr>
<td>ibm10e</td>
<td>6.56</td>
<td>932/990</td>
<td>6.01</td>
<td>462/1034</td>
<td>5.89</td>
<td>819/631</td>
</tr>
<tr>
<td>ibm10h</td>
<td>6.59</td>
<td>999/1221</td>
<td>6.00</td>
<td>451/658</td>
<td>5.80</td>
<td>871/770</td>
</tr>
<tr>
<td>ibm11e</td>
<td>5.34</td>
<td>727/613</td>
<td>4.61</td>
<td>425/634</td>
<td>4.39</td>
<td>819/582</td>
</tr>
<tr>
<td>ibm11h</td>
<td>4.88</td>
<td>705/583</td>
<td>4.56</td>
<td>422/517</td>
<td>4.35</td>
<td>758/717</td>
</tr>
<tr>
<td>ibm12h</td>
<td>10.40*</td>
<td>1100/6127</td>
<td>8.96</td>
<td>512/1775</td>
<td>8.56</td>
<td>809/1631</td>
</tr>
<tr>
<td>% improv.</td>
<td>15.18</td>
<td>10.44</td>
<td>5.79</td>
<td>2.67</td>
<td>14.10</td>
<td>1.30</td>
</tr>
</tbody>
</table>


