Effective Wire Models for X-Architecture Placement
Tung-Chieh Chen, Yi-Lin Chuang, and Yao-Wen Chang

Abstract—In this paper, we derive the X-half-perimeter wirelength (XHPWL) model for X-architecture placement and explore the effects of three different wire models on X-architecture placement, including the Manhattan-half-perimeter wirelength (MHPWL) model, the XHPWL model, and the X-Steiner wirelength (XStWL) model. For min-cut partitioning placement, we apply the XHPWL and XStWL models to the generalized net-weighting method that can exactly model the wirelength after partitioning by net weight. For analytical placement, we smooth the XHPWL function using log-sum-exp functions to facilitate analytical placement. This paper shows that both the XHPWL and XStWL models can reduce the X wirelength effectively. In particular, our results reveal the effectiveness of the X architecture on wirelength reduction during placement and, thus, the importance of the study on the X-placement algorithms, which is different from the results given in the work of Ono et al. which suggests that the X-architecture placement might not improve the X-routing wirelength over the Manhattan-architecture placement.

Index Terms—Min-cut, net weighting, partitioning, physical design, placement, Steiner tree, X architecture.

I. INTRODUCTION
A. X Architecture

As integrated-circuit (IC) geometries keep shrinking, interconnect delay has become the dominant factor in determining circuit performance. To minimize interconnect delay, the X architecture [3] has been introduced as a new interconnect architecture for the ICs to reduce interconnect length and thus improve circuit performance. The X architecture allows 45\(^\circ\) and 135\(^\circ\) routes, leading to smaller wirelength and, thus, smaller delay and power consumption. Theoretically, the maximum wirelength reduction by using the X architecture can be up to 29\%, as shown in Fig. 1.

The traditional Manhattan architecture has its obvious advantages of easier design, but it incurs significant and needless wirelength over the Euclidean optimum. As reported in [3], the X architecture results in significantly shorter average wirelength than the Manhattan architecture. The X architecture’s pervasive uses of diagonal routing can reduce wirelength. Furthermore, the wirelength reduction makes the circuit design problem easier to solve, resulting in faster timing closure.

Although the via count may increase in the X architecture, some previous studies on X-architecture routing have shown that the via issue is not a significant problem. Koh and Madden [4] pointed out that the increase in via cost is much less than expected, and the wirelength reduction may outweigh the additional via cost. The X-architecture

Manuscript received May 21, 2007; revised August 18, 2007 and October 29, 2007. This work was supported in part by the National Science Council of Taiwan under Grants NSC 94-2215-E-002-030 and NSC 94-2752-E-002-008-PAE. This paper was recommended by Associate Editor P. H. Madden.
T.-C. Chen and Y.-L. Chuang are with the Graduate Institute of Electronics Engineering, National Taiwan University, Taipei 106, Taiwan, R.O.C. (e-mail: donnie@eda.ee.ntu.edu.tw; nicky@eda.ee.ntu.edu.tw).
Y.-W. Chang is with the Department of Electrical Engineering and Graduate Institute of Electronics Engineering, National Taiwan University, Taipei 106, Taiwan, R.O.C. (e-mail: ywchang@cc.ee.ntu.edu.tw).
Digital Object Identifier 10.1109/TCAD.2008.917959
the X bounding box and another two properties for XHPWL for the case when no obstacle is present. Let the size of the X (Manhattan) bounding box for a point set \( P \) be \( S_x(P)/S_M(P) \).

Property 1) \( S_x(P) \leq S_M(P) \) for a point set \( P \).

Property 2) Every optimal X-Steiner tree (with the minimum wirelength) must be within its X bounding box.

Property 3) XHPWL(e) \( \leq \) MHPWL(e) for a multiterminal net \( e \).

Property 4) The XHPWL is a lower bound of the wirelength of a two-pin net for X routing.

Teig and Ganley [9] also proposed a method to estimate the wirelength. They first use a Manhattan bounding box to enclose all terminals. Then, the wirelength is estimated by \( L + S + (\sqrt{2} - 1) \), where \( L(S) \) is the length of the longer (shorter) side of the bounding box. Although their wire model can also correctly compute the shortest wirelength of a two-pin net for X routing, our experimental results show that their wire model may not lead to shorter total X wirelength. Compared with their wire model, our XHPWL is superior for at least the following two reasons, which will be shown in the later sections:

1) The fidelity of XHPWL for estimating XSiWL is higher than that of their wire model, and 2) XHPWL can easily be applied to analytical placement because of the concept of the “bounding box.”

III. X-ARCHITECTURE MIN-CUT PARTITIONING PLACEMENT

Partitioning placement recursively divides a placement region into several subregions, cuts a netlist into subnetlists, and assigns the subnetlists into regions [10]–[13]. Through the min-cut partitioning, the partitioning placer minimizes the cut size of each cutline, and the total wirelength is minimized indirectly.

To apply X-architecture wire models to min-cut placement, we use the net weighting technique proposed in [1]. Their net-weighting method can be generalized as follows. A circuit is modeled as a hypergraph. Each node in the hypergraph corresponds to a cell inside the region, with the node weight being set to the area of the corresponding cell. A two- or multiterminal net corresponds to one or two hyperedges. The hyperedge weight is set to the value of the wirelength contribution if the hyperedge is cut so that we can map the cut size to the resulting wirelength.

We consider a rectangular region to be vertically or horizontally divided into subregions 1 and 2. Let \( c_1 \) and \( c_2 \) be the centers of the two subregions. A net has multiple terminals \( \{v_1, v_2, \ldots, v_m, t_1, t_2, \ldots, t_n\} \), where \( v_1, v_2, \ldots, v_m \) are movable cells inside the region and \( t_1, t_2, \ldots, t_n \) are fixed terminals outside the region. Let \( w_1 \) be the wirelength when all cells are in subregion 1, \( w_2 \) be the wirelength when all cells are in subregion 2, and \( w_{12} \) be the wirelength when cells are in both subregions. We assume that all cells are placed at the center of the assigned region. See Fig. 4 used in [1] for an illustration of a net with three terminals. We have \( w_1 = \text{wirelength}(\{v_1, t_1, t_2, \ldots, t_n\}) \), \( w_2 = \text{wirelength}(\{v_2, t_1, t_2, \ldots, t_n\}) \), and \( w_{12} = \text{wirelength}(\{c_1, c_2, t_2, \ldots, t_n\}) \), where wirelength(\( \{p_1, p_2, \ldots, p_n\} \)) is the wirelength of the point set \( \{p_1, p_2, \ldots, p_n\} \) based on the given wire model.

We create a hypergraph \( G \) which has two fixed pseudonodes to represent the two subregions and has movable nodes to represent the movable cells. For a net, we introduce two hyperedges \( e_1 \) and \( e_2 \): \( e_1 \) connects all movable nodes and the fixed pseudonode corresponding to the subregion that results in a smaller wirelength; \( e_2 \) connects between all movable nodes. We then assign the weights of the hyperedges as weight(\( e_1 \)) = \( |w_2 - w_1| \) and weight(\( e_2 \)) = \( w_{12} - \max(|w_1|, w_2|) \). If the net has only one movable cell, we do not need to add \( e_2 \) because it is impossible to obtain the case with cells in both regions. \( w_{12} \) is usually larger or equal to \( \max(|w_1|, w_2) \) because separating cells into both regions often results in a larger wirelength. However, if
create the partitioning hypergraph according to the circuit netlist. Then, the weights of hyperedges in the hypergraph are assigned by the aforementioned net-weighting method. After min-cut partitioning, we obtain the resulting subpartitions and corresponding cells within them. If the circuit sizes in those subpartitions are still large, we add these subpartitions into the bin list to be partitioned later. We take each time the first bin from the bin list and bipartition it. The partitioning loop continues until the bin list is empty, and we legalize the placement by removing all overlaps to obtain the final placement.

IV. X-ARCHITECTURE ANALYTICAL PLACEMENT

The force-directed approach is widely used for the analytical placement. The interconnection between cells provides wire forces to pull cells together and minimize the total wirelength. Considering the wire forces alone, however, cannot always obtain legal placement due to large amounts of overlaps. Consequently, we need to add spreading forces to remove the overlaps between cells. The analytical placement is usually solved in an iterative fashion. The placement process minimizes the total wirelength and gradually adds more spreading forces until cells evenly spread to the whole chip.

For X-architecture analytical placement, we need to minimize the total X wirelength instead of the total Manhattan wirelength. Thus, we shall change the wire model from MHPWL to XHPWL. To facilitate XHPWL optimization, we use log-sum-exp (LSE) functions to smooth the XHPWL function in (2). The smoothed XHPWL function is shown in the following:

\[ \gamma \left( \sqrt{2} - 1 \right) \left( \log \sum_{i=1}^{n} e^{\frac{x_i}{\gamma}} + \log \sum_{i=1}^{n} e^{-\frac{x_i}{\gamma}} \right) + \log \sum_{i=1}^{n} e^{\frac{y_i}{\gamma}} + \log \sum_{i=1}^{n} e^{-\frac{y_i}{\gamma}} \]

- \[ \gamma \left( \sqrt{2} - 2 \right) \left( \log \sum_{i=1}^{n} e^{\frac{x_i+y_i}{\gamma}} + \log \sum_{i=1}^{n} e^{\frac{x_i-y_i}{\gamma}} \right) \]

(3)

\( \gamma \) is the decay factor, which is used to control the smoothness of the function. The smaller the value of \( \gamma \), the more smooth the function will be.

The XHPWL-LSE function is defined as:

\[ \gamma \left( \sqrt{2} - 1 \right) \left( \log \sum_{i=1}^{n} e^{\frac{x_i}{\gamma}} + \log \sum_{i=1}^{n} e^{-\frac{x_i}{\gamma}} \right) + \log \sum_{i=1}^{n} e^{\frac{y_i}{\gamma}} + \log \sum_{i=1}^{n} e^{-\frac{y_i}{\gamma}} \]

Fig. 5. X-architecture min-cut partitioning placement flow.

\[ w_{12} < \text{min}(w_1, w_2) \]

For X-architecture analytical placement, we need to minimize the total X wirelength instead of the total Manhattan wirelength. Thus, we shall change the wire model from MHPWL to XHPWL. To facilitate XHPWL optimization, we use log-sum-exp (LSE) functions to smooth the XHPWL function in (2). The smoothed XHPWL function is shown in the following:

\[ \gamma \left( \sqrt{2} - 1 \right) \left( \log \sum_{i=1}^{n} e^{\frac{x_i}{\gamma}} + \log \sum_{i=1}^{n} e^{-\frac{x_i}{\gamma}} \right) \]

- \[ \gamma \left( \sqrt{2} - 2 \right) \left( \log \sum_{i=1}^{n} e^{\frac{x_i+y_i}{\gamma}} + \log \sum_{i=1}^{n} e^{\frac{x_i-y_i}{\gamma}} \right) \]

(3)

1. Initialize placement;
2. while blocks not spread enough
3. while minimizing \( \alpha W + 3 \beta \);
4. Compute wire forces \( dw/dx \);
5. Compute spreading forces \( d\beta/dx \);
6. Move cells according to the gradient direction;
7. Update \( \alpha \) and \( \beta \);
8. Remove overlaps and output the placement;

Fig. 6. X-architecture analytical placement flow.

**TABLE I**

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Total X Half-Perimeter Wirelength (XHPWL)</th>
<th>Min-Cut Partitioning</th>
<th>Analytical</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MHPWL</td>
<td>XHPWL</td>
<td>XSWL</td>
</tr>
<tr>
<td>ibm01</td>
<td>4.506 e6</td>
<td>0.993</td>
<td>1.027</td>
</tr>
<tr>
<td>ibm02</td>
<td>1.331 e8</td>
<td>0.997</td>
<td>1.034</td>
</tr>
<tr>
<td>ibm07</td>
<td>3.006 e8</td>
<td>0.990</td>
<td>1.012</td>
</tr>
<tr>
<td>ibm08</td>
<td>3.182 e8</td>
<td>0.993</td>
<td>1.029</td>
</tr>
<tr>
<td>ibm09</td>
<td>2.735 e8</td>
<td>0.983</td>
<td>1.005</td>
</tr>
<tr>
<td>ibm10</td>
<td>5.257 e8</td>
<td>0.992</td>
<td>1.008</td>
</tr>
<tr>
<td>ibm11</td>
<td>4.049 e8</td>
<td>0.996</td>
<td>1.017</td>
</tr>
<tr>
<td>ibm12</td>
<td>7.022 e8</td>
<td>0.984</td>
<td>1.016</td>
</tr>
<tr>
<td>Average</td>
<td>1.000</td>
<td>0.991</td>
<td>1.018</td>
</tr>
</tbody>
</table>
This function has similar property to the MHPWL-LSE function: When $\gamma$ is sufficiently small, the XHPWL-LSE wirelength is close to the XHPWL.

The wire-force direction is given by the gradient direction of the wirelength function. The wire forces are along the gradient directions toward the interior of the bounding box. Therefore, compared with the XHPWL function, the XHPWL-LSE function can effectively reduce the size of the X bounding box and obtain smaller total X wirelengths for the X-architecture placement.

We adopt the analytical placement framework proposed in [14] to implement our X-architecture analytical placer. Fig. 6 shows the flow, which contains two loops. The inner loop uses the conjugate gradient method to minimize the objective function, $\alpha W + \beta O$, where $W$ is the wirelength function, $O$ is the overlap function, and $\alpha$ and $\beta$ are the corresponding weights. The outer loop updates $\alpha$ and $\beta$ to remove overlaps gradually (we use the same method described in [14] to update $\alpha$ and $\beta$). To optimize the X wirelength, we use the XHPWL-LSE in (3) for $W$. The placement loop continues until all cells are spread enough or the overlaps are small enough. Then, we legalize the placement by removing all overlaps to obtain the resulting placement.

V. EXPERIMENTAL RESULTS

We applied different wire models to both the min-cut partitioning placer NTUplace[12] and the analytical placer NTUplace3 [14]. For the min-cut partitioning placer, we have four wire models: MHPWL, XHPWL, XStWL, and TGXWL [9]. For the analytical placer, we have two wire models: MHPWL and XHPWL. We do not use XStWL and TGXWL for analytical placement because they cannot be applied directly. All experiments were performed on an AMD Opteron 2.6-GHz machine.

We used the benchmark “IBM version 2.0,” which is widely used in academia [15]. Although not reported here, the results on the ISPD’05 and ISPD’06 Placement Contest Benchmarks [16], [17] are similar. We used FLUTE [18] as our Steiner-tree estimator because it is very accurate and fast. For X-architecture Steiner trees, we first used FLUTE to find Steiner points to determine the Steiner-tree topology. Then, X routing was applied to compute the minimum distance for each net segment. Although this approach is not X-architecture aware, it can still provide good estimation of the XStWL. Our placers are comparable to other state-of-the-art academic placement tools in wirelength, including APace 2.0 [19], Capo 10.2 [11], Feng Shui 5.1 [13], and mPL6 [20].

Total XHPWL

In this paper, we show how much XHPWL can be reduced by using the XHPWL model. Table I gives the total XHPWL for different placement algorithms/wire models. As shown in the table, the XHPWL model can reduce the average total XHPWL by 0.9% for the min-cut and 3.2% for the analytical placement. The reductions are consistent for all circuits. Note that the XStWL model increases wirelength by 1.8% because this objective function does not optimize the total XHPWL directly.

Total Steiner-Wirelength Comparisons

We use the total Steiner wirelength to evaluate the quality of the placement. Compared with the half-perimeter wirelength, the Steiner wirelength is much closer to the routed wirelength. The results are shown in Table II. The left part of the table reports the total Manhattan-Steiner wirelengths (MStWLs), whereas the right part shows the total X-Steiner ones. We also compare the total XStWL using the wire model of Teig and Ganley (TGXWL). The average values are normalized to the respective placement algorithms using the MHPWL model. For the total X-Steiner, compared with the MHPWL model, the XStWLs are reduced by 0.7% and 4.7% for min-cut partitioning placement using the XHPWL and XStWL models, respectively. For analytical placement, compared with the MHPWL model, the XStWL is reduced by 2.8% on average. Note that TGXWL failed to generate any placement with a shorter total XStWL for all cases.

CPU Time

The XHPWL and XStWL models need more computation efforts than the MHPWL one. For the min-cut partitioning placement, the XHPWL and XStWL models incur the runtime overheads of about 7.6% and 21.9% on average, respectively. The XStWL model requires the highest CPU time because of its Steiner-tree construction. For the analytical placement, the XHPWL model incurs about 15.1% runtime overhead on average.

Wire-Model Accuracy and Fidelity

To compare the accuracy of the wire models, we show in Fig. 7 the ratios of MHPWL, TGXWL, and XHPWL to XStWL for nets with the pin counts ranging from 3 to 23 using the circuit ibm01. Unlike MHPWL, both TGXWL and XHPWL provide good lower bound estimations for XStWL; TGXWL and XHPWL are consistently no larger than XStWL, and their distributions are very similar. However, XHPWL gives a more accurate estimation to XStWL than TGXWL.

We further compute the correlation between different wire models. Table III gives the correlations of MHPWL, TGXWL, XHPWL, and XStWL w.r.t. MStWL and XStWL. Note that XStWL has the highest correlation because it is the exact XStWL. As shown in the table, XHPWL has higher correlations to both MStWL and XStWL than MHPWL and TGXWL. In contrast, the correlations of TGXWL to both MStWL and XStWL are even lower than those of MHPWL. It provides insights into why TGXWL does not lead to smaller total XStWL, as reported in Table II.
In those conditions, partitioning and analytical placement algorithms, respectively, reduce the wirelength by 11.6% and 11.0% on average for min-cut X-architecture placement, in contrast, the X-architecture routing can alone reduces the wirelength by only 7.7%–8.0% on average. With our MHPWL for most cases; this may again explain why TGXWL does not lead to shorter XStWLs, as shown in Table II.

Table III
WIRELENGTH CORRELATION BETWEEN WIRE MODELS IN THE CIRCUIT IBM01

<table>
<thead>
<tr>
<th>Wire Model</th>
<th>MHSWL</th>
<th>XSWL</th>
</tr>
</thead>
<tbody>
<tr>
<td>MHPWL</td>
<td>0.965</td>
<td>0.951</td>
</tr>
<tr>
<td>TGXWL</td>
<td>0.954</td>
<td>0.942</td>
</tr>
<tr>
<td>XHPWL</td>
<td>0.969</td>
<td>0.961</td>
</tr>
<tr>
<td>XSWL</td>
<td>0.997</td>
<td>1.000</td>
</tr>
</tbody>
</table>

Fig. 7. Comparison of the accuracy of XSWL estimators. (Left lines) MHPWL, (middle) TGXWL (Teig and Ganley’s model), and (right) XHPWL for nets with 3–23 pins in the circuit ibm01.

VI. CONCLUSION

We have proposed and studied the XHPWL and the XSWL models for min-cut partitioning and analytical placement. Experimental results have shown that using the XHPWL or XSWL model in placement can lead to shorter XSWLs than traditional Manhattan placement. The results reveal the effectiveness of the X architecture on wirelength reduction during placement and, thus, the importance of the study on the X-placement algorithms.

REFERENCES