Short Papers

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 weighting. 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° and 135° 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 full-chip routing [5] further shows that the via count for the X architecture is similar to that for the Manhattan architecture.

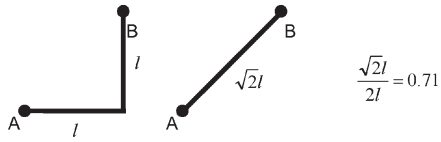

Fig. 1. Maximum wirelength reduction for the X architecture can be up to 29% compared with the Manhattan architecture. This situation occurs when the two pins are on a 45° line.

B. Previous Work

To fully utilize the X architecture, we need to consider both X-architecture-aware placement and routing. Some X-routing algorithms have been proposed in the literature [4]–[6], and their results show that the wirelength can be reduced effectively by using the X architecture. In contrast, not much work on X placement is studied in the literature. In [7], both 45° and 60° wiring metrics were explored. The work was based on some simplified assumptions that all cells are of the unit size and that only five pins and higher degree nets are considered. Furthermore, the simulated annealing method does not scale well.

Based on the partitioning placement framework, Teig and Ganley [8], [9] patented 45°/135° diagonal cutlines (or X cutlines) to partition the placement region to favor diagonal wiring. Very recently, Ono *et al.* [2] conducted a complete study on the patents [8], [9]. They found that X cutlines do not lead to better placement results for the X architecture; the resulting wirelength by using the X cutlines is even worse than that by using traditional Manhattan cutlines. Teig and Ganley [9] also proposed a new wirelength cost metric for X placement. However, we will show later that this wire model may not lead to shorter total X wirelength.

C. Our Contribution

In this paper, we derive a new X-half-perimeter wirelength (XHPWL) model for X-architecture placement. We define the X bounding box as the smallest bounding box formed by the 0°/45°/90°/135° line segments that enclose all terminals of a net. The XHPWL is the half of the perimeter length of the X bounding box. We then incorporate this new wire model into both min-cut partitioning and analytical placement algorithms. Experimental results show that the total X-Steiner wirelength (XStWL) can be reduced effectively.

Without X placement, our experimental results show that X routing reduces the wirelength by only about 7.7% compared with the traditional Manhattan routing. With X placement, the X routing can reduce the wirelength by 11.0% and 11.6% for analytical and min-cut partitioning placements, respectively. 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.

This paper is organized as follows. Section II introduces our new XHPWL model. The XHPWL model is applied to min-cut partitioning and analytical placement algorithms in Sections III and IV, respectively. The experimental results are given in Section V, and Section VI gives the conclusion.

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

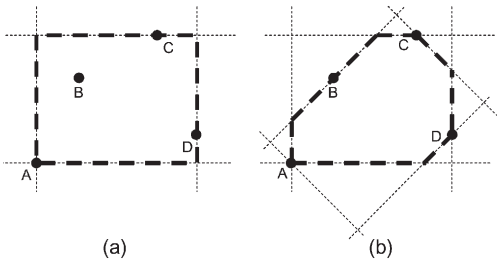

Fig. 2. Manhattan bounding box and X bounding box. (a) Manhattan bounding box. (b) X bounding box.

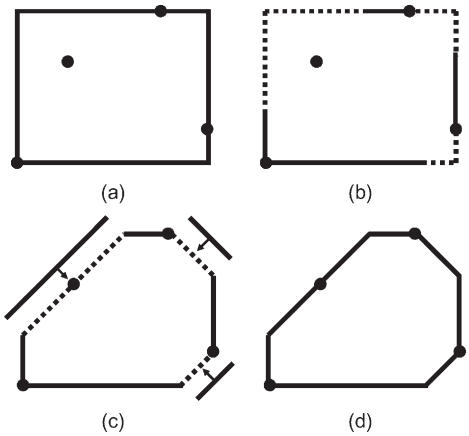

Fig. 3. Procedure of computing the perimeter length of the X bounding box. (a) Compute the Manhattan bounding box. (b) Remove the dotted line segments. (c) Add the oblique line segments. (d) Obtain the resulting X bounding box.

II. XHPWL Model

Traditional placement for the Manhattan architecture is based on the minimization of the Manhattan-half-perimeter wirelength (MHPWL for short, or traditionally called HPWL). An example Manhattan bounding box of a four-terminal net is shown in Fig. 2(a). The MHPWL is the half of the perimeter length of the Manhattan bounding box, and the MHPWL of a net e can be computed by

$$\text{MHPWL}(e) = \max_{v_i,v_j \in e} |x_i - x_j| + \max_{v_i,v_j \in e} |y_i - y_j| \qquad (1)$$

where v_i is a terminal of the net and (x_i, y_i) is the coordinate of v_i.

The MHPWL does not consider the 45°/135° routes of the X architecture. We thus propose a new XHPWL model for the X architecture. We define the X bounding box as the smallest bounding box formed by 0°/45°/90°/135° line segments. Fig. 2(b) shows an example X bounding box of the four terminals.

To compute the perimeter length of the X bounding box, we can use the procedure shown in Fig. 3, and the XHPWL of a net e can be computed by

$$\begin{aligned}\text{XHPWL}(e) = (\sqrt{2}-1)&\left(\max_{v_i,v_j \in e} |x_i - x_j| + \max_{v_i,v_j \in e} |y_i - y_j|\right) \\ -(\sqrt{2}/2-1)&\left(\max_{v_i,v_j \in e} |x_i + y_i - x_j - y_j| \right. \\ &\left. + \max_{v_i,v_j \in e} |x_i - y_i - x_j + y_j|\right) \qquad (2)\end{aligned}$$

where v_i is a terminal of the net and (x_i, y_i) is the coordinate of v_i. Based on the triangle inequality, the total length of the added oblique line segments in Fig. 3(c) is always smaller than that of the removed line segments in Fig. 3(b). We have the following two properties for 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 XStWL 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 connected to the 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 $w_1 = \text{wirelength}(\{c_1, t_1, t_2, \ldots, t_n\})$, $w_2 = \text{wirelength}(\{c_2, t_1, t_2, \ldots, t_n\})$, and $w_{12} = \text{wirelength}(\{c_1, c_2, t_1, t_2, \ldots, t_n\})$, 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 $\text{weight}(e_1) = |w_2 - w_1|$ and $\text{weight}(e_2) = w_{12} - \max(w_1, w_2)$. 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

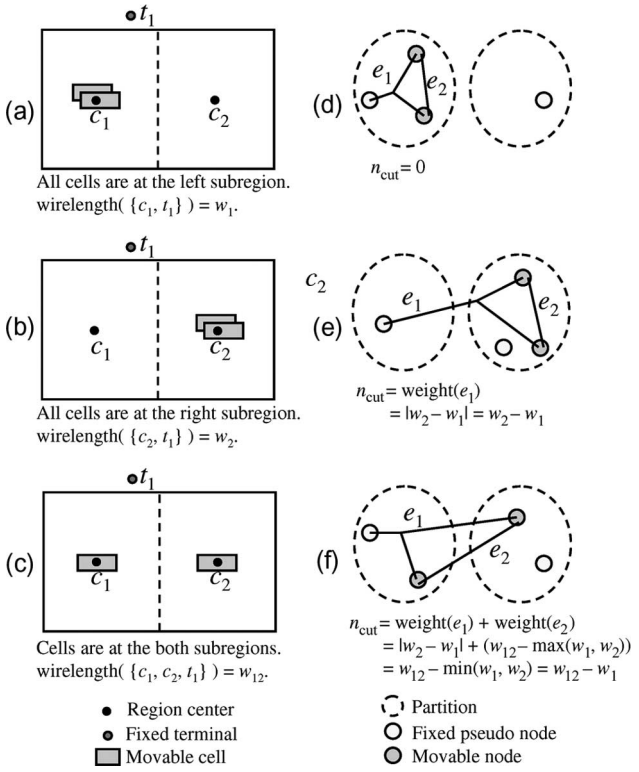

Fig. 4. Example of determining a net weight [1]. (a)–(c) are three possible partitioning results. (d)–(f) are corresponding partitioning hypergraphs.

1. Create the first partition;
2. **while** the bin list not empty
3. Select a bin from the bin list;
4. Create the partitioning graph;
5. Assign net-weights using generalized net-weighting;
6. Find a min-cut bi-partitioning result;
7. Add large sub-partitions into the bin list;
8. Remove overlaps and output the placement;

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

$w_{12} < \max(w_1, w_2)$, we may make weight$(e_2) = 0$ to avoid negative edge weights for which some hypergraph partitioners cannot handle. It is shown in [1] that the aforementioned net-weighting method can exactly map the resulting total wirelength (based on the given wire model) to the min-cut cost.

Partitioning the resulting hypergraph gives the partition to which the cell belongs and the cut size n_{cut}. We have the following two theorems [1].

Theorem 1: With the generalized net weighting, the wirelength l of a net with the cut size n_{cut} is given by $l = \min(w_1, w_2) + n_{\text{cut}}$.

Theorem 2: The generalized net-weighting method exactly maps the resulting total wirelength (based on the given wire model) to the min-cut cost.

The theorems can be proved by the similar method used in [1].

Two models can be used in the generalized net-weighting method to minimize the total XStWL for min-cut partitioning placement: 1) the XHPWL model and 2) the XStWL model. For the XHPWL model, we can use (2) as the wirelength function to evaluate w_1, w_2, and w_{12} for each net and assign net weights according to the aforementioned method. For the XStWL model, we need to construct X-Steiner trees to evaluate w_1, w_2, and w_{12} for each net.

We adopt the min-cut partitioning placement framework proposed in [12] to implement our X-architecture partitioning placer. Fig. 5 shows the flow. It contains a loop of recursive bipartitioning. First, all cells are located at the top-level partition (the whole chip). We

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

Fig. 6. X-architecture analytical placement flow.

TABLE I
COMPARISON OF THE RESULTING TOTAL XHPWL BASED ON DIFFERENT PLACEMENT ALGORITHMS/WIRE MODELS

	Total X Half-Perimeter Wirelength (XHPWL)				
Algorithm	Min-Cut Partitioning			Analytical	
Wire Model	MHPWL	XHPWL	XStWL	MHPWL	XHPWL
ibm01	4.506 e7	0.994	1.027	4.324 e7	0.994
ibm02	1.331 e8	0.997	1.034	1.205 e8	0.964
ibm07	3.006 e8	0.990	1.012	2.893 e8	0.984
ibm08	3.182 e8	0.993	1.029	2.919 e8	0.971
ibm09	2.735 e8	0.983	1.005	2.432 e8	0.962
ibm10	5.237 e8	0.992	1.008	4.944 e8	0.948
ibm11	4.049 e8	0.996	1.017	3.801 e8	0.958
ibm12	7.022 e8	0.984	1.016	6.656 e8	0.961
Average	1.000	0.991	1.018	1.000	0.968

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:

$$\begin{aligned}\text{XHPWL-LSE}(e) &= \gamma(\sqrt{2}-1)\left(\log\sum_{v_i\in e} e^{\frac{x_i}{\gamma}} + \log\sum_{v_i\in e} e^{\frac{-x_i}{\gamma}} + \log\sum_{v_i\in e} e^{\frac{y_i}{\gamma}} + \log\sum_{v_i\in e} e^{\frac{-y_i}{\gamma}}\right) \\ &\quad - \gamma(\sqrt{2}/2-1)\left(\log\sum_{v_i\in e} e^{\frac{x_i+y_i}{\gamma}} + \log\sum_{v_i\in e} e^{\frac{-x_i-y_i}{\gamma}} + \log\sum_{v_i\in e} e^{\frac{x_i-y_i}{\gamma}} + \log\sum_{v_i\in e} e^{\frac{-x_i+y_i}{\gamma}}\right). \quad (3)\end{aligned}$$

TABLE II
COMPARISON OF THE RESULTING TOTAL MSTWL AND TOTAL XSTWL BASED ON DIFFERENT PLACEMENT ALGORITHMS/WIRE MODELS

	Total Manhattan-Steiner Wirelength (MStWL)					Total X-Steiner Wirelength (XStWL)					
Algorithm	Min-Cut Partitioning			Analytical		Min-Cut Partitioning				Analytical	
Wire Model	MHPWL	XHPWL	XStWL	MHPWL	XHPWL	MHPWL	XHPWL	XStWL	TGXWL	MHPWL	XHPWL
ibm01	6.185 e7	1.010	1.000	5.699 e7	1.026	5.716 e7	0.994	0.967	1.049	5.272 e7	0.993
ibm02	1.813 e8	1.014	0.991	1.615 e8	0.985	1.683 e8	0.997	0.953	1.021	1.502 e8	0.963
ibm07	3.894 e8	1.014	0.996	3.658 e8	1.027	3.596 e8	0.990	0.950	1.023	3.345 e8	0.992
ibm08	4.351 e8	1.013	0.991	3.986 e8	1.002	4.019 e8	0.991	0.948	1.029	3.663 e8	0.975
ibm09	3.608 e8	1.006	0.977	3.248 e8	1.013	3.321 e8	0.988	0.939	1.012	2.908 e8	0.974
ibm10	6.803 e8	1.017	0.995	6.285 e8	0.991	6.266 e8	0.995	0.953	1.035	5.805 e8	0.956
ibm11	5.118 e8	1.027	1.012	4.740 e8	1.000	4.716 e8	0.999	0.959	1.018	4.324 e8	0.963
ibm12	9.056 e8	1.014	1.004	8.484 e8	1.000	8.349 e8	0.988	0.956	1.027	7.782 e8	0.963
Average	1.000	1.014	0.996	1.000	1.006	1.000	0.993	0.953	1.027	1.000	0.972

This function has similar property to the MHPWL-LSE function: When γ 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 α and β are the corresponding weights. The outer loop updates α and β to remove overlaps gradually (we use the same method described in [14] to update α and β). 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 NTUplace1 [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 APlace 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.

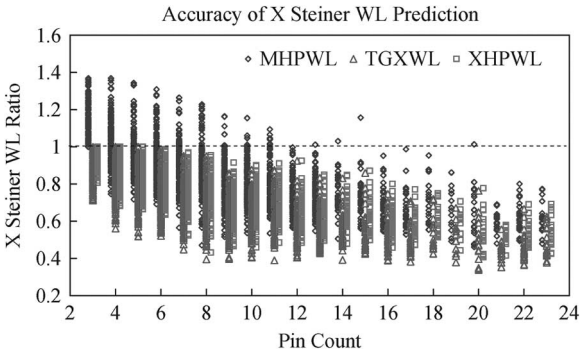

Fig. 7. Comparison of the accuracy of XStWL estimators. (Left lines) MHPWL, (middle) TGXWL (Teig and Ganley's model), and (right) XHPWL for nets with 3–23 pins in the circuit ibm01.

TABLE III
WIRELENGTH CORRELATION BETWEEN WIRE MODELS IN THE CIRCUIT IBM01

Wire Model	Correlation	
	MStWL	XStWL
MHPWL	0.965	0.951
TGXWL	0.954	0.942
XHPWL	0.969	0.961
XStWL	0.997	1.000

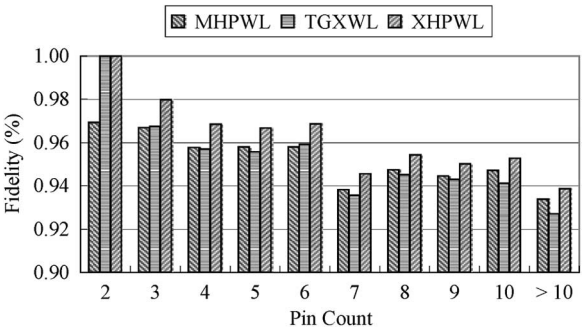

Fig. 8. Fidelity of MHPWL, TGXWL, and XHPWL for estimating XStWLs based on nets with different pin counts.

We also evaluate the wire models by their fidelity. If a wire model f has high fidelity to the XStWL model, it satisfies the following two conditions.

1) $f(P_1) > f(P_2)$ if $\text{XStWL}(P_1) > \text{XStWL}(P_2)$.
2) $f(P_1) = f(P_2)$ if $\text{XStWL}(P_1) = \text{XStWL}(P_2)$.

In those conditions, P_1 and P_2 are two sets of net pins with the same pin count. Fig. 8 shows the fidelity of the three wire models for the XStWL estimation based on the IBM benchmarks. The fidelity is given by the percentage of the point sets satisfying the aforesaid conditions. As shown in the figure, our XHPWL has the highest fidelity for all pin counts. Note that the fidelity of TGXWL is smaller than that of MHPWL for most cases; this may again explain why TGXWL does not lead to shorter XStWLs, as shown in Table II.

Wirelength Using Different Architectures

Without our X-architecture placement, the X-architecture routing alone reduces the wirelength by only 7.7%–8.0% on average. With our X-architecture placement, in contrast, the X-architecture routing can reduce the wirelength by 11.6% and 11.0% on average for min-cut partitioning and analytical placement algorithms, respectively.

VI. CONCLUSION

We have proposed and studied the XHPWL and the XStWL models for min-cut partitioning and analytical placement.

Experimental results have shown that using the XHPWL or XStWL model in placement can lead to shorter XStWLs 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

[1] T.-C. Chen, Y.-W. Chang, and S.-C. Lin, "IMF: Interconnect-driven multilevel floorplanning for large-scale building-module designs," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des.*, San Jose, CA, Nov. 2005, pp. 159–164.

[2] S. Ono, S. Tilak, and P. H. Madden, "Bisection based placement for the X architecture," in *Proc. IEEE/ACM Asia South Pacific Des. Autom. Conf.*, Yokohama, Japan, Jan. 2007, pp. 153–158.

[3] S. L. Teig, "The X architecture: Not your father's diagonal wiring," in *Proc. Syst. Level Interconnect Prediction Workshop*, San Diego, CA, Apr. 2002, pp. 33–38.

[4] C.-K. Koh and P. H. Madden, "Manhattan or non-Manhattan? A study of alternative VLSI routing architectures," in *Proc. ACM Great Lakes Symp. VLSI*, Chicago, IL, Mar. 2000, pp. 47–52.

[5] T.-Y. Ho, C.-F. Chang, Y.-W. Chang, and S.-J. Chen, "Multilevel full-chip routing for the X-based architecture," in *Proc. ACM/IEEE Des. Autom. Conf.*, Anaheim, CA, Jun. 2005, pp. 597–602.

[6] Z. Cao, T. Jing, Y. Hu, Y. Shi, X. Hong, X. Hu, and G. Yan, "DraXRouter: Global routing in X-architecture with dynamic resource assignment," in *Proc. IEEE/ACM Asia South Pacific Des. Autom. Conf.*, Yokohama, Japan, Jan. 2006, pp. 618–623.

[7] H. Chen, C.-K. Cheng, A. B. Kahng, I. Mandoiu, and Q. Wang, "Estimation of wirelength reduction for λ-geometry vs. Manhattan placement and routing," in *Proc. Syst. Level Interconnect Prediction Workshop*, Monterey, CA, Apr. 2003, pp. 71–76.

[8] S. L. Teig and J. L. Ganley, "Partitioning placement method and apparatus," U.S. Patent 6 848 091, Jan. 25, 2005.

[9] S. L. Teig and J. L. Ganley, "Partitioning placement method using diagonal cutlines," U.S. Patent 6 516 455, Feb. 4, 2003.

[10] T. Taghavi, X. Yang, B.-K. Choi, M. Wang, and M. Sarrafzadeh, "Dragon2006: Blockage-aware congestion-controlling mixed-size placer," in *Proc. ACM Int. Symp. Phys. Des.*, San Jose, CA, Apr. 2006, pp. 209–211.

[11] J. Roy, D. Papa, A. Ng, and I. Markov, "Satisfying whitespace requirements in top-down placement," in *Proc. ACM Int. Symp. Phys. Des.*, San Jose, CA, Apr. 2006, pp. 206–208.

[12] T.-C. Chen, T.-C. Hsu, Z.-W. Jiang, and Y.-W. Chang, "NTUplace: A ratio partitioning based placement algorithm for large-scale mixed-size designs," in *Proc. ACM Int. Symp. Phys. Des.*, San Francisco, CA, Apr. 2005, pp. 236–238.

[13] A. R. Agnihotri, S. Ono, and P. H. Madden, "Recursive bisection placement: Feng Shui 5.0 implementation details," in *Proc. ACM Int. Symp. Phys. Des.*, San Francisco, CA, Apr. 2005, pp. 230–232.

[14] T.-C. Chen, Z.-W. Jiang, T.-C. Hsu, and Y.-W. Chang, "A high-quality mixed-size analytical placer considering preplaced blocks and density constraints," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des.*, San Jose, CA, Nov. 2006, pp. 187–192.

[15] S. N. Adya, M. C. Yildiz, I. L. Markov, P. G. Villarrubia, P. N. Parakh, and P. H. Madden, "Benchmarking for large-scale placement and beyond," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 23, no. 4, pp. 472–487, Apr. 2004.

[16] *ISPD 2005 Placement Contest*. [Online]. Available: http://www.sigda.org/ispd2005/contest.htm

[17] *ISPD 2006 Placement Contest*. [Online]. Available: http://www.sigda.org/ispd2006/contest.html

[18] C. Chu and Y.-C. Wong, "Fast and accurate rectilinear Steiner minimal tree algorithm for VLSI design," in *Proc. ACM Int. Symp. Phys. Des.*, 2005, pp. 28–35.

[19] A. B. Kahng and Q. Wang, "A faster implementation of APlace," in *Proc. ACM Int. Symp. Phys. Des.*, San Jose, CA, Apr. 2006, pp. 218–220.

[20] T. Chan, J. Cong, J. Shinnerl, K. Sze, and M. Xie, "mPL6: Enhanced multilevel mixed-size placement," in *Proc. ACM Int. Symp. Phys. Des.*, San Jose, CA, Apr. 2006, pp. 212–214.

[21] J. A. Roy, J. F. Lu, and I. L. Markov, "Seeing the forest and the trees: Steiner wirelength optimization in placement," in *Proc. ACM Int. Symp. Phys. Des.*, San Francisco, CA, Apr. 2005, pp. 78–85.