Zero-Change Netlist Transformations: A New Technique for Placement Benchmarking

Andrew B. Kahng, *Member, IEEE*, and Sherief Reda

***Abstract*—In this paper, the authors introduce the concept of zero-change netlist transformations (ZCNTs) to: 1) quantify the suboptimality of existing placers on artificially constructed instances and 2) "partially" quantify the suboptimality of placers on synthesized netlists from arbitrary netlists by giving lower bounds to the suboptimality gap. Given a netlist and its placement from a placer, a class of netlist transformations that synthesizes a different netlist from the given netlist is formally defined, but yet the new netlist has the same half-perimeter wire length (HPWL) on the given placement. Furthermore, and more importantly, the optimal HPWL value of the new netlist is no less than that of the original netlist. By applying the transformations and reexecuting the placer, any deviation in HPWL as a lower bound to the gap from the optimal HPWL value of the new synthesized netlist can be interpreted. The transformations allow us to: 1) increase the cardinality of hyperedges; 2) reduce the number of hyperedges; and 3) increase the number of two-pin edges, while maintaining the placement HPWL constant. It is developed here methods that apply ZCNTs to synthesize netlists having typical netlist statistics. Furthermore, an approach to estimate the suboptimality of other metrics, such as rectilinear minimum-spanning tree (RMST) and minimum-Steiner tree, is extended. Using these transformations, the suboptimality of some of the existing academic placers (FengShui, Capo, mPL, Dragon) is studied on synthesized netlists from the IBM benchmarks with instances ranging from $10k$ to $210k$ placeable instances. The results show that current placers exhibit suboptimal behavior to ZCNTs with varying degree according to the placer. Systematic suboptimality deviations in HPWL and RMST are displayed on the synthesized netlists from IBM (version 1) benchmarks. The specific nature of the transformations points out troublesome netlist structures and possible directions for improvement in the existing placers.**

***Index Terms*—Algorithms, benchmarking, performance, placement, suboptimality, wirelength.**

I. Introduction

In THE physical design stage of every digital integrated circuit, we are concerned with placing or assigning all components of the circuit to specific locations on the layout area such that no two components overlap. Given the limited routing resources and stringent performance constraints of state-of-the-art designs, it is necessary to assign the components to minimize the total net wire length. Minimizing the wire length overall reduces the demand on routing resources as well as total timing and power.

Wire length is measured by the sum of Steiner tree length of the various nets. Routed wire length is typically slightly larger than the Steiner-minimum-tree (SMT) length since contention on routing resources by different nets might lead to detours, eventually increasing the wire length. Given that the Steiner tree problem is NP-hard [33], placers typically minimize and report other metrics that are faster and easier to compute. Half-perimeter wire length (HPWL) is the most widely used placement objective and reported metric. The HPWL of a net is equal to half of the perimeter of the smallest bounding box enclosing all nodes of a net. The popularity of HPWL is no surprise given that it is equivalent to the SMT cost for two-pin and three-pin nets, and it is well correlated with the SMT cost for multipin (≥ 4) nets [8]. Placers minimize the HPWL heuristically by using, for example, min-cut partitioners [4], [34], [35], quadratic or analytical solvers [13], [21], or simulated annealing [32]. The HPWL is also the typically reported metric when comparing results of different placers on various benchmarks [2], [6], [9].

Given a benchmark circuit and a placer, placement benchmarking, or placer suboptimality evaluation, is the problem of finding how close the placer's result is to the optimal result for the given benchmark. We use the term exact suboptimality quantification to refer to calculating the exact difference between the placer's result and the unknown optimal result, and the term partial suboptimality quantification to refer to calculating a lower bound on the difference between the placer's result and the unknown optimal result. The placement problem or the HPWL minimization is notoriously hard since: 1) It is NP-hard [26]; 2) it has no polynomial constant approximation algorithms [25]; and 3) it has no approximation schemes [26].[1] Given these theoretical results, researchers must rely on heuristic methods to solve the problem. Lack of placement benchmarking can lead to frustration since there is no direct way to assess whether existing heuristics are sufficiently close to optimal for arbitrary instances.

In this paper, we propose a new technique for placement benchmarking. We introduce the concept of zero-change netlist transformations (ZCNTs) and devise a set of such transformations to: 1) exactly quantify the suboptimality of existing placers on artificially constructed instances and, more importantly, 2) partially quantify their suboptimality on synthesized netlists from arbitrary given netlists. Given a netlist and its placement

Manuscript received April 10, 2005; revised August 12, 2005 and October 18, 2005. This paper was recommended by Associate Editor J. Lillis.

A. B. Kahng is with the Departments of Computer Science and Engineering and Electrical and Computer Engineering, University of California at San Diego, La Jolla, CA 92093-0114 USA, and also with Blaze DFM, Inc., Sunnyvale, CA 94089 USA (e-mail: abk@ucsd.edu; abk@blaze-dfm.com).

S. Reda is with the Department of Computer Science and Engineering, University of California at San Diego, La Jolla, CA 92093-0114 USA (e-mail: sreda@cs.ucsd.edu).

Digital Object Identifier 10.1109/TCAD.2006.882473

[1] Assuming $P \neq NP$.

from a placer, ZCNTs alter the given netlist while keeping its HPWL constant, resulting in a zero change to its HPWL. More importantly, the optimal placement HPWL of the new netlist has a value no less than the original netlist's optimal HPWL. Thus, by executing the placer on the new netlist, we can interpret any deviation of the new HPWL from the original HPWL as a lower bound on the deviation from optimal results for the new netlist. Since our transformations might change basic netlist statistics, compromising the realism of the produced netlist, we propose extensions to our approach to keep the basic netlist statistics intact. Furthermore, we also propose how to apply our technique with respect to other metrics such as rectilinear minimum spanning tree (RMST) or rectilinear SMT (RSMT) costs. We support our techniques with extensive empirical results. Our results show that existing placers fail to reproduce their original HPWL results and incur serious deviations. The measured suboptimality gaps also increase as our transformations increase in magnitude.

The organization of this paper is as follows. Section II classifies benchmarking approaches, summarizes previous work on the placement benchmarking problem, and gives a number of preliminaries and definitions essential to understand this paper. Section III formally introduces the concept of ZCNTs and their use in placement benchmarking. Section IV gives several ZCNTs. Experimental results from the application of different transformations to various netlists are given in Section V. Finally, Section VI summarizes the implications of this paper and gives directions for future work.

II. Background and Preliminaries

Before describing our approach, we first outline the possible general benchmarking methods and previous work on the placement problem in Section II-A. Second, we give the necessary notations and definitions for understanding this paper in Section II-B.

A. Benchmarking Approaches

Numerous heuristics have been proposed to solve the placement since its introduction four decades ago [28]. It is a fundamental question to ask how close a placement heuristic's (or placer's) suboptimal result is to the optimal result for a given benchmark instance. The relationship between a suboptimal result w and the optimal result w^* is schematically shown in Fig. 1. The suboptimality gap $w - w^*$ is an indicator of the performance of the placement heuristic with respect to an optimal algorithm. Assuming that $N \neq NP$, the optimal result w^* is computationally infeasible to calculate except for extremely small instances. Thus, to estimate the suboptimality gap, it is necessary to rely on other approaches. We can taxonomize these approaches as follows.

1) Algorithms with guaranteed performance: In this approach, an algorithm is proven to give results that do not exceed a certain upper bound on suboptimality from the optimal result. Unfortunately, for the placement problem, it has been proven that unless $P = NP$, there does not exist any polynomial constant ratio approximation algorithms or even ε-approximation schemes [25], [26]. For example, Sahni and Gonzalez [26] show that if the placement problem has a polynomial ε-approximation algorithm, then it is possible to decide whether or not a given graph has a Hamiltonian cycle–an NP-complete problem–in polynomial time.

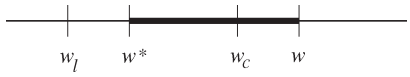

Fig. 1. Relationship between various wire-length quantities. w is a placer's suboptimal wire-length result. w^* is the optimal result. w_l is a lower bound. w_c is a precalculated wire length. $w - w^*$ is the suboptimality gap.

2) Placement lower bounds: In this approach, it would be possible to calculate a lower bound $w_l \leq w^*$ on the optimal placement of a given instance. The amount $w - w_l$ is then an upper bound on the suboptimality gap, as shown in Fig. 1. Naturally, the tighter w_l is to w^*, the better is our estimate of the suboptimality gap. Placement lower bounds is one of the least tackled issues in the placement literature. Donath [12] probabilistically calculated the expected lower bounds for only random graphs and showed that existing placement algorithms at this time exhibit large suboptimality gaps. However, attempts by the authors to replicate his calculations with current placers and instances sizes yield extremely large suboptimality gaps. This might suggest that such lower bounds are perhaps quite loose for modern instances.
3) Instances with precalculated wire length: In this approach, a benchmark is constructed with either known optimal wire length or precalculated suboptimal wire length. In the case of optimal constructions, a recent paper by Chang *et al.* [6] uses an overlooked construction method by Hagen *et al.* [15] to optimally construct a number of benchmarks (PEKO) with known optimal HPWL. We, however, note that this construction method was proposed very early in the placement literature [16], more than 30 years ago. Despite the meticulous efforts in [6] to generate benchmarks with typical netlist statistics, the benchmarks are considered unrealistic since only local signals are considered. For example, a two-pin net can only be connected to spatially adjacent objects. To overcome this drawback, Cong and co-workers [5], [9] added global hyperedges and established placement upper bounds. In essence, such upper bounds are precalculated suboptimal wire lengths for the generated benchmarks. Another method of generating instances with known precalculated wire length is through scaling a given instance and comparing the results of the placement heuristic on the scaled instance against a precalculated value from the unscaled instance [15]. If the precalculated wire-length instance is w_c, then the amount $w - w_c$ is a lower bound on the suboptimality gap as shown in Fig. 1.

In addition to these approaches, a number of papers in the benchmarking literature deal with general placement methodologies or the effect of netlist structures on the suboptimality gap. For example, results of different placers on various benchmarks are given in [2], where HPWL, timing, and

routability results of different placers are tabulated. The results show that placers exhibit different efficiencies on different benchmark families. Placer efficiency with respect to various netlist structures is studied by Liu and Marek-Sadowska [23]. Using the PEKO benchmark generator and existing placers, the effects of net degree distribution, net count, and Rent's exponent of the netlist are studied and tabulated. Practical conclusions are also given to justify the performance of different placers. Furthermore, Kahng and Mantik [17] study the mismatch between incremental optimizers, e.g., partitioners and engineering-change order (ECO) placers, and instance perturbations. In another effort, the stability of different runs of the academic place Capo [4] on the same benchmark is studied [1], where trying a number of randomly selected cells to their regions is proposed to stabilize results from different Capo runs. Another recent paper [24] studies the reason for the suboptimal behavior of placers on the PEKO benchmarks, and concludes that poor detailed placement is the likely culprit.

B. Preliminaries

A circuit netlist is a hypergraph $H = (V, E)$, where V is a set of vertices representing the circuit cells, and E is the set of hyperedges representing the circuit wires. A hyperedge $e \in E$ is a set of vertices $e \in 2^V$, where $|e|$ gives the cardinality or degree of hyperedge e. The placement area is composed of a number of sites that cells can legally occupy. Each cell may occupy a number of sites depending on its width. A placement of a hypergraph is defined as follows.

Definition 1: Given an enumeration of possible placement sites, a two-dimensional placement π of a given hypergraph $H(V, E)$ is a mapping π: $V \rightarrow Z^+$ assigning a placement site to every netlist cell such that no two cells overlap, i.e., no two cells share a common site. If a cell occupies more than one site, then the mapping gives the first occupied site.

Definition 2: The HPWL of a hyperedge e in a given placement π is the length of half the perimeter of the smallest bounding box that includes all vertices of e.[2] Such HPWL of a hyperedge is denoted by $l(e, \pi)$. The total HPWL (or wire length) is $L(H, \pi) = \sum_{e \in E} l(e, \pi)$.

A placement π_* of a hypergraph H that has minimum total HPWL is called an optimal placement. The HPWL of an optimal placement is called optimal wire length or HPWL.[3] A suboptimal placement is a placement with a total HPWL larger than the optimal HPWL.

Definition 3: A netlist or a hypergraph transformation applied to an input hypergraph $H_1 = (V, E)$ produces a new hypergraph $H_2 = (V, E')$ with the same set of vertices as H_1 but with a different set of hyperedges, i.e., a netlist transformation changes the connectivity.

With these basic definitions, we are ready to introduce the concept of ZCNTs.

[2]We always assume that hyperedges/nets are connected to cells via pins at the center of the cells. Another alternative is to measure half the perimeter of the bounding box completely enclosing the cells of a net.

[3]There can be more than one optimal placement yielding the same optimal HPWL.

III. Suboptimality Evaluation Using ZCNTs

In this section, we propose and define the concept of ZCNTs and examine how it can be used in placement benchmarking.

Definition 4: Given a placement π_1 of some hypergraph H_1, a netlist transformation that synthesizes H_2 from H_1 is a zero change if the following two properties are satisfied.

1) Quiescency property: $L(H_1, \pi_1) = L(H_2, \pi_1)$, i.e., the transformation results in zero change to HPWL with respect to the input placement π_1.
2) Hardness property: For any other placement π_k: $L(H_1, \pi_k) \leq L(H_2, \pi_k)$.

The composition of ZCNTs is also a ZCNT. If $\mathcal{Z}(H_1, \pi_1)$ denotes an arbitrary ZCNT that takes as inputs a hypergraph H_1 and a placement π_1 and outputs a new transformed netlist, then the composition of m ZCNTs can be expressed as

$$H_2 = \mathcal{Z}(H_1, \pi_1), H_3 = \mathcal{Z}(H_2, \pi_1), \ldots, H_m = \mathcal{Z}(H_{m-1}, \pi_1) \tag{1}$$

such that for the given placement π_1

$$L(H_1, \pi_1) = L(H_2, \pi_1) = \cdots = L(H_m, \pi_1) \tag{2}$$

and for any other placement π_k

$$L(H_1, \pi_k) \leq L(H_2, \pi_k) \leq \cdots \leq L(H_m, \pi_k). \tag{3}$$

From the hardness property, it is possible to establish a relationship between the optimal placement π_1^* of the original hypergraph H_1 and the optimal placement π_2^* of hypergraph H_2 obtained from the application of one or more ZCNTs.

Theorem 1: Given an original hypergraph H_1, a hypergraph H_2 generated from ZCNTs applied to H_1 has an optimal HPWL no less than that of the original hypergraph, i.e., $L(H_2, \pi_2^*) \geq L(H_1, \pi_1^*)$.

Proof: Toward a contradiction, assume that $L(H_2, \pi_2^*) < L(H_1, \pi_1^*)$. Using π_2^* for H_1 gives a placement with HPWL $L(H_1, \pi_2^*) \leq L(H_2, \pi_2^*)$ from the hardness property. Consequently, $L(H_1, \pi_2^*) < L(H_1, \pi_1^*)$, contradicting the assumption that π_1^* is the optimal placement of H_1. ■

From the previous theorem, it is easy to prove the following.

Corollary 1: If the given placement π_1 is optimal, i.e., $\pi_1 = \pi_1^*$, then the π_1^* is also optimal for the new hypergraph. ■

The zero-change properties as well as the results of Theorem 1 and Corollary 1 can be visually represented by a hypothetical plot as shown in Fig. 2.

A. Using ZCNT to Quantify the Exact Suboptimality Gap of Placers on Special Instances

Corollary 1 leads us to a discussion of special cases where ZCNTs can be used to quantify the entire suboptimality gap. We consider a trivial instance netlist with a known optimal placement wire length, namely a clique $\mathcal{C}$, which has the same wire length for any placement. We then execute ZCNTs on $\mathcal{C}$ using any reference placement π^* (note that we can choose any random placement as the reference placement) to produce a new netlist $\mathcal{C}'$. $\mathcal{C}'$ is not trivial; nevertheless, by Corollary 1, we have that π^* is still optimal for $\mathcal{C}'$. On the other hand, $\mathcal{C}'$ does

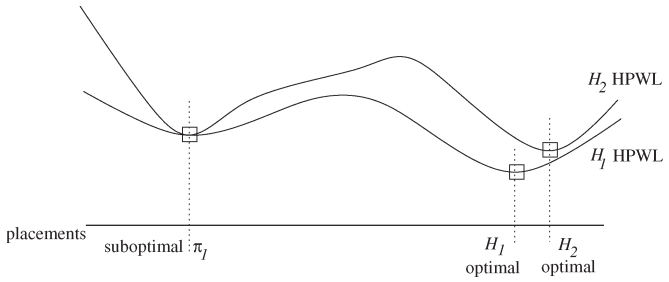

Fig. 2. Hypothetical plot showing the relationship between the HPWL of H_2 and H_1 placements. The horizontal axis represents the various placements, while the vertical axis gives the HPWL values. We can see that for any placement π_k: $L(H_2, \pi_k) \geq L(H_1, \pi_k)$ and for π_1: $L(H_1, \pi_1) = L(H_2, \pi_1)$.

not have the property that any arbitrary placement is optimal. If we execute a placer P on instance $\mathcal{C}'$, then the difference between the wire length reported by P and the known optimal wire length is exactly equal to the suboptimality gap of P on $\mathcal{C}'$. This method recollects methods used for testing network flow algorithms [11] (Palubetskis algorithm), where flows are added and subtracted along certain paths such that the total flow stays the same. We will quantify the suboptimality of placers on such netlists in the experiment section below. In reality, we are interested in quantifying the suboptimality on instances that are "real" or as close to "real" as possible. Thus, we next examine how to use the ZCNT to calculate lower bounds–instead of exact bounds as the case with the clique–on the suboptimality gaps of netlists that have been synthesized from general netlists.

B. Using ZCNT to Partially Quantify the Suboptimality Gap of Placers on Instances Synthesized From General Instances

The use of ZCNTs for general benchmarking is illustrated in Fig. 3. Given a netlist H, placer P produces a placement π_1 with HPWL $w_1 = L(H_1, \pi_1)$. Given π_1, applying ZCNTs to H_1 produces a new netlist H_2. From the quiescency property, $L(H_2, \pi_1) = L(H_1, \pi_1)$. However, executing P on H_2 produces a new placement π_2 with some wire length $w_2 = L(H_2, \pi_2)$. The main question is whether $w_1 = w_2$. If the placer is optimal, then $w_1 = w_2$. If the placer is suboptimal, then there are three possibilities.

1) $w_1 = w_2$ indicating that the placer is stable and not sensitive to the netlist transformations.
2) $w_2 < w_1$ indicating that the original placement was not optimal for the original netlist H_1 and that the transformations lead the placer to a better suboptimal placement π_2 for H_1 since $L(H_1, \pi_2) \leq w_2 = L(H_2, \pi_2) < w_1 = L(H_2, \pi_1) = L(H_1, \pi_1)$.
3) $w_2 > w_1$ showing that the placer is suboptimal and sensitive to the netlist transformations. Since w_1 acts as an upper bound to the optimal placement of H_2, the amount $w_2 - w_1 = L(H_2, \pi_2) - L(H_2, \pi_1)$ is a lower bound on the suboptimality gap of placer P on the new netlist H_2 which is equal to $w_2 - L(H_2, \pi_2^*) = L(H_2, \pi_2) - L(H_2, \pi_2^*)$.

An important characteristic of ZCNTs is that the optimal placement HPWL of the new netlist H_2 is no less than that of the original netlist H_1 as established in Theorem 1. Thus, executing the placer on the new netlist likely yields the third possibility where $w_2 > w_1$. This will be empirically demonstrated in Section V. The possibility of using ZCNTs to calculate lower bounds on the suboptimality gaps of the synthesized netlists raises a number of interesting questions.

1) How tight is the calculated lower bound? Answering this question entails calculating the exact suboptimality gap for the synthesized netlist which is as hard as calculating the suboptimality gap for the original netlist. Thus, the calculated lower bound can only serve as a certification of placer suboptimality on the synthesized netlist by at least its value.
2) Can the suboptimality gap on a synthesized netlist reveal any information about the suboptimality gap on its original netlist? This is unlikely. However, we can regard the deviation in wire length as a form of placer sensitivity to transformations on the original netlist.

One may wonder about $w_3 = L(H_1, \pi_2)$ produced from using π_2 for the original netlist H_1. This raises the possibility of using netlist transformations to improve the placeability of netlists [14]. Notice that from the hardness property, we already know that $L(H_1, \pi_2) \leq L(H_2, \pi_2)$.

From the taxonomy introduced in the previous section, it is clear that our zero-change methodology fits in the category of instance generation with precalculated wire length. One key difference between our approach and other approaches is the ability to extract partial suboptimality information from synthesized netlists from any given arbitrary benchmark. We now propose a number of ZCNTs to assess the performance of different placers.

IV. ZCNT

In this section, we give a number of netlist transformations, which change the netlist connectivity but not the vertex set and have the key properties of ZCNTs. We propose three basic transformations: hyperedge-cardinality increase, hyperedge decomposition, and edge substitution. We extend the applicability of these transformations in two ways. First, we compose them in a hybrid fashion; and second, we embed them in a flow that preserves basic netlist statistics. Finally, we discuss how to evaluate placer suboptimality using other metrics such as RMST and RSMT. We start by introducing the hyperedge-cardinality-increase transformation.

A. Hyperedge-Cardinality Increase

The purpose of this transformation is to assess the sensitivity of placers to hyperedge-cardinality increase by examining the impact of increasing the cardinality of hyperedges. We only increase the cardinality of hyperedges of degree ≥ 3. Our transformation is simple: Given a netlist H and its placement π_1, the bounding box of each hyperedge (excluding two-pin edges) is calculated, and an additional number of vertices are added to each hyperedge from within its bounding box. The cardinality increase procedure HYPERC is given in Fig. 4. In our experiments, we limit the amount of hyperedge-cardinality increase. In this case, if there are a number of vertices inside the bounding

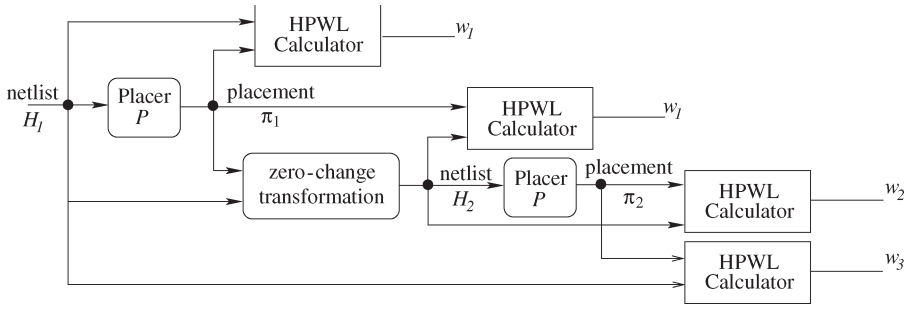

Fig. 3. Conceptual presentation of ZCNTs. The difference $w_2 - w_1$ represents a suboptimality measure of placer P.

Input: A hypergraph $H_1 = (V, E_1)$ and a placement π_1 for H.

Output: A new hypergraph $H_2 = (V, E_1)$.

1. Initialize $E_2 = E_1$.
2. For each hyperedge $e_i \in E_2$ where $|e_i| \geq 3$:
3. Find the set of vertices C_{e_i} enclosed within the bounding box of e_i in placement π_1.
4. If $C_{e_i} \neq \emptyset$ then augment hyperedge e_i as follows: $e_i = e_i \cup S_e$, where $S_{e_i} \subseteq C_{e_i}$.
5. Return hypergraph $H_2 = (V, E_2)$.

Fig. 4. Procedure HYPERC for hyperedge-cardinality increase.

box to choose from, we always prioritize vertices of the least degree to break the ties. Before we prove that HYPERC is a ZCNT, we state the following lemma which is easy to prove.

Lemma 1: Given two sets of nodes S_1 and S_2: If $S_1 \subseteq S_2$, then $l(S_1, \pi_k) \leq l(S_2, \pi_k)$ in any placement π_k. ■

Lemma 1 basically states that HPWL is monotonic [3].

Theorem 2: Procedure HYPERC in Fig. 4 is a ZCNT.

Proof: If the netlist produced by procedure HYPERC has the quiescency and hardness properties of Definition 4, then the theorem is proved. We will prove that each of these properties holds.

1) Quiescency: Given a hypergraph H_1 and a placement π_1, applying procedure HYPERC produces a new hypergraph H_2. By construction, adding a number of vertices to a hyperedge from within its bounding box does not change its HPWL value. Therefore, $L(H_2, \pi_1) = L(H_1, \pi_1)$.
2) Hardness: Given some $\pi_k \neq \pi_1$, H_1 would have an HPWL value of $L(H_1, \pi_k)$. Replacing each hyperedge e_i in H_1 with e_i' according to procedure HYPERC yields an HPWL value of $L(H_1, \pi_k) + \sum_i (l(e_i', \pi_k) - l(e_i, \pi_k)) \geq L(H_1, \pi_k)$ since $l(e_i', \pi_k) \geq l(e_i, \pi_k)$ by Lemma 1. Thus, $L(H_2, \pi_k) \geq L(H_1, \pi_k)$. ■

Finally, we note that the inverse or "anti" transformation to HYPERC, where a hyperedge's cardinality is decreased, does not satisfy the zero-change requirements, since it might yield a netlist with lower optimal HPWL than the original netlist.

B. Hyperedge Decomposition

Our second transformation simplifies a hyperedge by decomposing or partitioning it into two intersecting hyperedges,

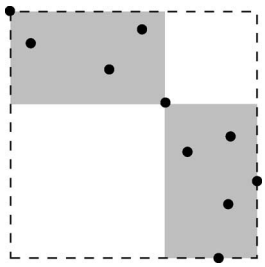

Fig. 5. Example of an optimally decomposable hyperedge.

with each of the new hyperedges having smaller cardinality than the original hyperedge. We define an optimal hyperedge decomposition as follows.

Definition 5: A hyperedge e is optimally decomposable in some placement π_k if it is possible to decompose e into two intersecting hyperedges e_1 and e_2, such that $l(e, \pi_k) = l(e_1, \pi_k) + l(e_2, \pi_k)$ and $e = e_1 \cup e_2$.

Fig. 5 shows a hyperedge, whose bounding box is represented by a dashed line, optimally decomposed into two hyperedges as shown by the dashed rectangles. The following lemma is crucial for our transformation.

Lemma 2: For any two hyperedges e_i and e_j with $|e_i \cap e_j| \neq \emptyset$, $\max(l(e_i, \pi_k), l(e_j, \pi_k)) \leq l(e_i \cup e_j, \pi_k) \leq l(e_i, \pi_k) + l(e_j, \pi_k)$ in any placement π_k.

Proof: The proof is by enumerating all possible cases for bounding boxes of e_i and e_j. Since $|e_i \cap e_j| \neq \emptyset$, there are only three possible configurations for the bounding boxes of e_i and e_j.

1) Contained: In this case, the bounding box of one hyperedge is completely contained within the other bounding box as shown in Fig. 6(a). In this case, $l(e_i \cup e_j, \pi_k) = \max(l(e_i, \pi_k), l(e_j, \pi_k))$.
2) Overlapping: In this case, the two bounding boxes overlap with $l(e_i \cup e_j, \pi_k) < l(e_i, \pi_k) + l(e_j, \pi_k)$ as shown in Fig. 6(b) and (c).
3) Touching: In this case, the two bounding boxes touch each other at a common vertex with $l(e_i \cup e_j, \pi_k) = l(e_i, \pi_k) + l(e_j, \pi_k)$ as shown in Fig. 6(d). ■

From the previous lemma, it is easy to see the following.

Lemma 3: e_1 and e_2 give an optimal decomposition of e in some placement π_k only if $|e_1 \cap e_2| = 1$ and the bounding boxes of e_1 and e_2 touch at their common vertex. ■

Lemma 3 provides us with a simple characterization to optimally decompose any hyperedge. Using this characterization,

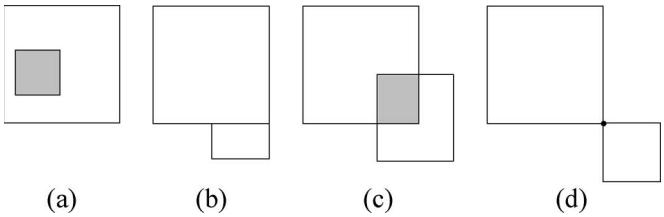

Fig. 6. Enumeration of possible bounding-box configurations.

Input: A hypergraph $H_1 = (V, E_1)$ and its placement permutation π_1.

Output: A hypergraph $H_2 = (V, E_2)$.

1. Iterate until there is no possible decomposition:
2. Set $E_2 = \emptyset$.
3. For each hyperedge e_i in E_1:
4. For each vertex $v_j \in e_i$:
5. If e_i can be partitioned into two sets e_i^1 and e_i^2 such that the bounding boxes of e_i^1 and e_i^2 touch each other at v_j then insert e_i^1 and e_i^2 into E_2 and goto Step 3. else insert e_i into E_2.
6. Set $E_1 = E_2$.
7. Return hypergraph $H_2 = (V, E_2)$

Fig. 7. Procedure HYPERD for hyperedge decomposition.

we devise a procedure for optimal hyperedge decomposition (procedure HYPERD) as given in Fig. 7. The procedure examines every hyperedge with degree larger than two and checks whether it is possible to optimally decompose it at each of its vertices. If there exists more than one possible vertex to optimally decompose at, we break the tie by choosing the vertex that results in the most balanced decomposition or partition. We next prove that the procedure HYPERD is a zero-transformation procedure.

Theorem 3: Procedure HYPERD is a ZCNT.

Proof: If the netlist produced by procedure HYPERD has the quiescency and hardness properties of Definition 4, then the theorem is proved. We will prove that each of these properties holds.

1) Quiescency: Since procedure HYPERD decomposes edges only optimally according to Lemma 3, it is clear that $L(H_1, \pi_1) = L(H_2, \pi_1)$.
2) Hardness: Given some $\pi_k \neq \pi_1$, H_1 would have an HPWL value of $L(H_1, \pi_k)$. Replacing each hyperedge e_i in H_1 with e_i' according to procedure HYPERD yields an HPWL value of $L(H_1, \pi_k) + \sum_i (l(e_i^1, \pi_k) + l(e_i^2, \pi_k) - l(e_i, \pi_k)) \geq L(H_1, \pi_k)$ since $l(e_i^1, \pi_k) + l(e_i^2, \pi_k) \geq l(e_i, \pi_k)$ by Lemma 2. Thus, $L(H_2, \pi_k) \geq L(H_1, \pi_k)$.

Before ending this section, we note that the inverse or "anti" transformation to HYPERD, i.e., merging two touching hyperedges into one bigger hyperedge, does not satisfy the zero-change requirements, since it might yield a netlist with a lower optimal HPWL than the original netlist. ■

Input: A hypergraph $H_1 = (V, E_1)$ and its placement π_1.

Output: A hypergraph $H_2 = (V, E_2)$.

1. Initialize $E_2 = E_1$.
2. Find a two-pin edge $\{u, v\}$ in E_2 and calculate its bounding box in π_1
3. Find a node p inside the bounding box of $\{u, v\}$
4. If such node p exists then
5. Delete $\{u, v\}$ from E_2 and insert two new two-pin edges $\{u, p\}$ and $\{p, v\}$ in E_2.

Fig. 8. Procedure EDGESUB for two-pin edge substitution.

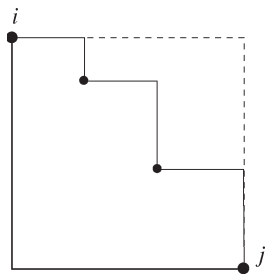

Fig. 9. Substituting an edge between i and j by a path according to procedure EDGESUB does not change the wire length of a given placement.

C. Edge Substitution

Our third transformation deals with edges, i.e., hyperedges of degree two. Our transformation does not change the cardinality of edges but rather increases the total number of two-pin edges. We start with the following fact that characterizes the triangle inequality in Manhattan or rectilinear metric.

Fact 1: If d_{ij} gives the Manhattan distance between two placement sites i and j, then $d_{ij} \leq d_{iq} + d_{qj}$ for any site q, and if q lies within the bounding box defined by sites i and j, then $d_{ij} = d_{iq} + d_{qj}$.

We leverage Fact 1 for netlist transformation as given in procedure EDGESUB of Fig. 8, where we take every edge, calculate its bounding box, and substitute it with two edges by using a third vertex from within its bounding box. Since there can be more than one vertex that is eligible to take the role of the third vertex, we always break ties in favor of the vertex of the least degree. Edge substitution can be carried out more than once, effectively transforming an edge between sites i and j into a path between sites i and j as depicted in Fig. 9.

Theorem 4: Procedure EDGESUB is a ZCNT.

Proof: If the netlist produced by procedure EDGESUB has properties 1) and 2) of Definition 4, then the theorem is proved. We prove that each of these properties holds.

1) Quiescency: Given a hypergraph H_1 and a placement π_1, applying procedure EDGESUB produces a new hypergraph H_2. By construction, substituting an edge by two

TABLE I
IMPACT OF ZCNTS ON TWO BASIC NETLIST STATISTICS

Transformation	Netlist Characteristic	
	Number of Hyperedges	Total pin count (or cardinality)
Hyperedge Cardinality Increase	0	+1
Hyperedge Decomposition	+1	+1
Edge Substitution	+1	+2

edges, or more, using nodes from within its bounding box, does not change the total HPWL.

2) Hardness: Given some $\pi_k \neq \pi_1$, H_1 has HPWL value of $L(H_1, \pi_k)$. Substituting every edge $\{u, \nu\}$ by two edges $\{u, p\}$ and $\{p, \nu\}$ gives a new netlist H_2 such that $L(H_2, \pi_k) = L(H_1, \pi_k) + \sum_{\forall\{u,\nu\}}(d_{\pi_k(u)\pi_k(p)} + d_{\pi_k(p)\pi_k(\nu)} - d_{\pi_k(u)\pi_k(\nu)}) \geq L(H_1, \pi_k)$ by Fact 1.

We note that if procedure HYPERC of Section IV-A (for hyperedge-cardinality increase) is allowed to operate on two-pin edges, then transformation EDGESUB can be considered as a combination of hyperedge-cardinality increase (HYPERC) on two-pin nets, immediately followed by the hyperedge decomposition (HYPERD) of Section IV-B. ■

As a final remark: The inverse or "anti" transformation to EDGESUB, by substituting a path of edges with a single edge, does not satisfy the zero-change requirements.

D. Hybrid Transformations

It is possible to apply the previous three transformations on a given netlist and empirically examine the collective impact of all transformations. As elaborated in Section III, the hybrid or composite application of all ZCNTs is also zero change. For example, we can compose hyperedge-cardinality increase, hyperedge decomposition, and edge substitution in sequence to yield a hybrid ZCNT.

E. Transformations Preserving Netlist Statistics

All of our ZCNTs increase a certain characteristic of the netlist, such as hyperedge cardinality or total number of hyperedges. We tabulate the impact of our transformations on basic netlist statistics in Table I. The table gives the increase in number of hyperedges and total pin count, i.e., total hyperedge cardinality, for each one of our transformations. Since most VLSI instances have a ratio of about one between the number of nets and number of cells, our previous transformations synthesize netlists with a larger ratio. It is certainly desirable that the new hypergraph instance has the same statistics as the original hypergraph. Thus, we investigate in this section how to produce a new hypergraph with reasonable (e.g., preserving realism) statistics.

Many papers have researched what constitutes a realistic netlist [22], [29], [31]. For example, realistic netlists exhibit typical values for: 1) the number of hyperedges in comparison to the nodes; 2) the average node degree; 3) a hyperedge-cardinality power-law distribution [30]; and 4) Rent parameter [22].

To keep a degree of realism in our generated hypergraphs, we use a simple strategy: Before the initial placement, we preprocess the input hypergraph to reduce its hyperedge cardinality and number of hyperedges by exactly the amount that will be increased due to ZCNTs. Using this strategy, a given input hypergraph H_1 is preprocessed to a new hypergraph H_1' which is then used to derive the ZCNT flow of Fig. 3. The final hypergraph H_2 produced from applying ZCNTs to H_1' will have the same amount of hyperedges and total hyperedge cardinality as the original hypergraph H_1.

Certainly, such a framework of hypergraph preprocessing followed by application of ZCNTs does not produce a realistic netlist; nevertheless, it keeps the "basic" netlist statistics intact. The following are the possible preprocessing steps.

1) Removal of random hyperedge which decreases the number of hyperedges in a netlist.
2) Converting multiple hyperedges into one hyperedge. For example, given a number of hyperedges $e_1, e_2, \ldots, e_k$, we can convert them into one bigger hyperedge by deleting all of them and creating a new hyperedge $e_1 \cup e_2 \cup, \ldots, \cup e_k$.
3) Reduction of hyperedge cardinality by removing an arbitrary node from any hyperedge with three or more nodes.

F. Benchmarking Other Metrics: RMST and RSMT

RSMTs or RMST share and differ with HPWL in a number of ways.

1) The bounding box of a net in a given placement is unique; however, there might be more than one minimum RMST or RSMT. Furthermore, since RSMT is an NP-hard problem [33], we can only approximate it using a suboptimal RSMT. There are numerous heuristic RSMT construction algorithms, and one possibility is to use the RMST as a possible RSMT heuristic with a performance ratio of at most $3/2$ [33]. In practice, the average RMST/RSMT ratio for random n points approaches 1.12 [20].
2) HPWL is monotonic. Given any placement π_k: If $S_1 \subseteq S$, then $L(S_1, \pi_k) \leq L(S, \pi_k)$. The monotonic property also holds for SMT since augmenting a set of points by an additional point cannot reduce the length of the SMT connecting these points. On the other hand, such a property does not hold for the minimum spanning tree: Adding an additional point to an existing point set might actually reduce the length of the MST connecting these points. This might happen if the added point is a Steiner point [3], [27].

TABLE II
EXACT SUBOPTIMALITY QUANTIFICATION FOR SPECIAL INSTANCES. A "–" INDICATES PLACER FAILED WHILE PLACING THE INSTANCE. WE REPORT THE ACTUAL HPWL AND THE PERCENTAGE DEVIATION FROM THE OPTIMAL PLACEMENT BETWEEN PARENTHESES

bench	Statistics after ZCNT			Optimal	Placer			
	nodes	nets	pins	HPWL	Capo	FengShui	Dragon	mPL
C100	100	6435	15480	33000	33301 (0.91%)	-	33502 (1.52%)	33086 (0.26%)
C400	400	103740	207480	1064000	1085590 (1.97%)	-	-	1071430 (0.65%)
C900	900	525915	1051830	8091000	8367070 (3.33%)	-	-	-
C1600	1600	1662960	3325920	34112000	35019800 (2.64%)	-	-	-

In this paper, we use the RMST as a heuristic to construct the RSMT as suggested by Property 1. Thus, we focus our discussions on the RMST. Given a set of nodes S representing a hyperedge and a placement π_1, our transformations for RMST/RSMT suboptimality evaluation are as follows.

1) Hyperedge-cardinality increase: According to our previous discussion, adding a node from within the bounding box of a net S might either increase or decrease the RMST length of S. Thus, we modify the hyperedge cardinality in a simple way to handle this. After increasing the cardinality, we recalculate the RMST value of all nets and use the total RMST as the precalculated RMST value of the netlist. This value is used to benchmark the RMST value produced when the placer is executed on the new netlist.[4]
2) Hyperedge decomposition: If hyperedge S is optimally decomposed into two hyperedges S_1 and S_2, then the RMST of S_1 plus the RMST of S_2 is greater than or equal the RMST of S. This can be proven by contradiction: If the RMST of S_1 plus RMST of S_2 is less than the RMST of S, then we can "concatenate" the RMSTs of S_1 and S_2 to obtain a new RMST for S that is of less value than the original minimal RMST of S.
3) Edge substitution: In this case, the RMST is equivalent to HPWL since this transformation only applies to two-pin edges and produces two-pin edges where both RMST and HPWL have the same value.

Our discussion of various netlist transformations is now complete. We empirically explore the impact of ZCNTs in the next section.

V. EXPERIMENTAL RESULTS

In this section, we empirically evaluate the suboptimality of the existing placers with respect to our proposed transformations. This evaluation is carried out using the IBM benchmarks (version 1)[5] and four academic placers. The components of the IBM (version 1) benchmarks are comprised of standard cells with varying widths, where the pins of all components are by default placed at the center of their respective cells. We use the following placers in our empirical evaluation.

1) Capo [4] (version 9.0 with feedback [19]): Capo uses a min-cut engine in a top–down bisection framework to deliver fast results.
2) FengShui [35] (version 2.6): FengShui uses a min-cut engine based on iterative deletion in a top–down framework.
3) Dragon [34] (version 3.01): Dragon uses a min-cut engine in a top–down quadrisection framework. Dragon also uses simulated annealing to improve its results.
4) mPL (version 4.0) [10]): mPL is an analytical placer that uses a nonlinear programming formulation in a multilevel optimization framework.

Since all pins are placed at the center of their respective cells in all circuits of the IBM benchmarks, we measure the HPWL center-to-center, and any additional pins necessitated by our transformations are also placed at centers of the cells. Before carrying out our experiments, we estimate the noise [1], [18] of different placers by reporting the average difference in HPWL for two different executions of a single placer on the same netlist, but with different ordering of nets [17]. Our results show that FengShui has a noise margin of around 0.92%, mPL has a noise margin of around 0.89%, Capo has a noise margin of around 2.9%, and Dragon has a noise margin of around 3.37%.

A. Using ZCNT to Quantify the Exact Suboptimality Gap of Placers on Special Instances

In this special case, we first construct a clique and fix any placement as its optimal reference placement (we use square layouts). This is valid since all placements give the same optimal wire length for a given clique. Using the reference placement, we transform the clique using ZCNTs to a nontrivial instance, by increasing the total hyperedge cardinality by 30%, and the total number of two-pin edges by 30% using the edge

[4]Since the RMST length might change after the transformation, the transformation is no longer zero change. In the "lucky" case where the final RMST value is higher than the precalculated, the lower bound on suboptimality, as measured by the difference between the new and precalculated values, is likely to be of less value than those corresponding to ZCNT.

[5]Other benchmarks like the PEKO instances [6] are not suitable, since all of their nets are local in the optimal placement. This leaves no room to exploit our transformations. For example, there is no possibility to execute two-pin edge substitution.

substitution. We then execute the placers on the new netlist and report the difference between the observed wire length and the known optimal wire length. We give the results in Table II. The suboptimality gap is small and around 1%–3%. Certainly, such instances do not resemble typical VLSI instances, but they nevertheless demonstrate how ZCNTs can be used to exactly quantify the entire suboptimality gap.

B. Using ZCNT to Partially Quantify the Suboptimality Gap of Placers on Instances Synthesized From General Instances

In this section, we use the ZCNT to partially quantify the placers, i.e., by calculating lower bounds on the suboptimality gap of the placers on instances synthesized from general instances. Our experimental execution flow is based on the outline of Fig. 3. We note that before applying our transformations, we sort all nets by their HPWL in the given placement in a decreasing order. In all experiments, we report the percentage deviations $(w_2 - w_1)/w_1$, and in only one experiment, HYPEREDGE DECOMPOSITION, we report $(w_3 - w_1)/w_1$. The first amount, $(w_2 - w_1)/w_1$, is a lower bound on deviation from the optimal HPWL of the new netlists. The second amount, $(w_3 - w_1)/w_1$, is reported to see if the transformations can lead to an improvement in the placeability of the original netlists. We have encountered a limited amount of unsuccessful placement executions. The results of these runs are reported in the tables by a dash (–).

1) Hyperedge-Cardinality Increase: In a first series of experiments, we empirically determine the performance of placers with respect to the zero-change hyperedge-cardinality-increase transformation as given by procedure HYPERC. Table III gives the results of HYPERC with a total cardinality, i.e., total pin count, increase of 20% for each benchmark. We report the deviation of each placer HPWL with respect to its own placement. From the results, it is clear that none of the four placers managed to maintain their original HPWL. All placers exhibit a substantial unnecessary increase in HPWL.

To further study the impact of zero-change hyperedge-cardinality increase, we focus on the IBM01 benchmark and measure the HPWL in response to a total cardinality increase from 0% to 25% in increments of 5%. We plot the results in Fig. 10. Notice that in contrast to Table III, we directly report the HPWL values and not the percentage deviations. From the figure, current placers exhibit a suboptimal behavior, and there is a general trend of increasing the HPWL in response to hyperedge cardinality. We also notice that the relative performance ranking of various placers differs depending on the amount of hyperedge-cardinality increase. For example, FengShui's performance is deteriorating more gracefully than Dragon.

2) Hyperedge Decomposition: In a second series of experiments, we empirically determine the performance of placers to zero-change hyperedge decomposition as given by the HYPERD procedure. All hyperedges are decomposed until no further decomposition is possible. The empirical results are given in Table IV, where the percentage changes $(w_2 - w_1)/w_1$ and $(w_3 - w_1)/w_1$ are reported. It is clear from the empirical results that w_3 is always less than w_2 in agreement with the hardness property. We also notice that placers are sometimes able to exploit such transformation to slightly improve their results.

We also carry out a more detailed study on the IBM01 benchmark as given in Fig. 11. In the plot, the x-axis gives the amount of zero-change hyperedge decomposition in increments of 5%, and the y-axis gives the HPWL produced from the various placers. In general, the response of placers

TABLE III
DEVIATIONS IN HPWL IN RESPONSE TO 20% ZERO-CHANGE TOTAL HYPEREDGE-CARDINALITY INCREASE. DEVIATIONS ARE CALCULATED WITH RESPECT TO EACH PLACER'S ORIGINAL PLACEMENT RUN

bench	Placer			
	Capo	FengShui	Dragon	mPL
ibm01	9.72%	5.55%	20.43%	7.28%
ibm02	10.55%	6.79%	-	10.31%
ibm03	13.17%	3.63%	6.67%	5.38%
ibm04	12.53%	3.86%	16.26%	8.72%
ibm05	3.75%	1.50%	2.84%	4.01%
ibm06	4.48%	3.66%	15.71%	13.74%
ibm07	8.30%	3.61%	16.85%	5.50%
ibm08	5.48%	9.34%	7.97%	9.48%
ibm09	10.52%	8.93%	24.51%	7.54%
ibm10	11.02%	3.43%	12.90%	11.22%
ibm11	22.49%	5.40%	17.98%	24.42%
ibm12	4.14%	3.71%	11.14%	22.49%
ibm13	6.02%	4.33%	7.43%	6.10%
ibm14	16.73%	3.18%	15.80%	12.34%
ibm15	9.97%	3.03%	9.04%	12.71%
ibm16	5.30%	7.89%	29.11%	18.38%
ibm17	11.69%	4.87%	17.09%	14.18%
ibm18	7.22%	3.54%	20.29%	16.10%
Average	9.62%	4.79%	14.82%	11.66%

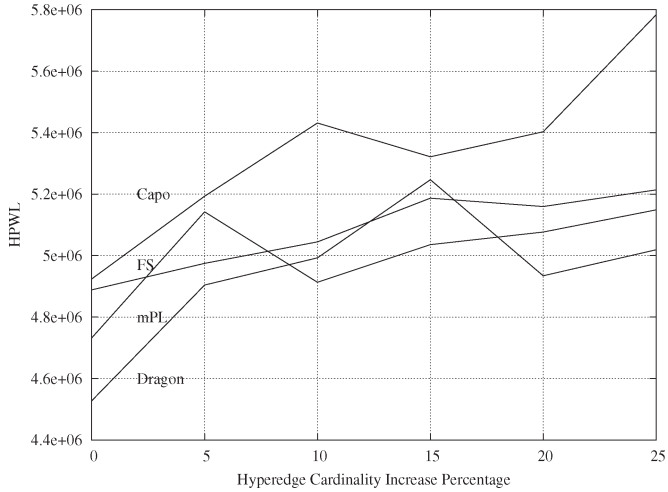

Fig. 10. Effect of zero-change total hyperedge cardinality, total pin count, increase on the performance of various placers on the IBM01 benchmark. Total hyperedge-cardinality increase is varied from 0% to 25% in increments of 5%.

TABLE IV
HPWL DEVIATIONS IN RESPONSE TO ZERO-CHANGE HYPEREDGE DECOMPOSITION. DEVIATIONS ARE CALCULATED WITH RESPECT TO EACH PLACER'S ORIGINAL PLACEMENT RUN

bench	Placer							
	$\frac{w_2-w_1}{w_1}$ results				$\frac{w_3-w_1}{w_1}$ results			
	Capo	FengShui	Dragon	mPL	Capo	FengShui	Dragon	mPL
ibm01	4.69%	-	3.52%	0.17%	1.99%	-	2.04%	-1.33%
ibm02	2.08%	6.75%	18.83%	3.30%	0.32%	4.62%	12.68%	2.31%
ibm03	2.19%	1.83%	2.12%	0.30%	0.98%	0.98%	1.05%	-0.62%
ibm04	2.17%	3.54%	-0.20%	-3.15%	0.84%	2.37%	-0.97%	-4.18%
ibm05	7.19%	-0.28%	0.72%	-0.74%	5.61%	-0.82%	0.14%	-1.21%
ibm06	2.61%	1.62%	4.06%	-0.61%	0.88%	0.70%	2.88%	-2.05%
ibm07	2.93%	2.77%	0.06%	-0.73%	1.48%	1.84%	-1.08%	-1.77%
ibm08	2.64%	-0.77%	3.25%	3.75%	1.42%	-1.55%	2.26%	2.71%
ibm09	2.78%	2.22%	2.97%	0.84%	0.83%	0.96%	1.79%	-0.31%
ibm10	8.64%	3.37%	6.90%	9.45%	6.63%	2.02%	5.48%	7.95%
ibm11	2.61%	-0.78%	2.95%	0.37%	1.05%	-1.69%	1.98%	-0.82%
ibm12	3.49%	2.90%	0.73%	0.84%	1.87%	1.64%	-0.43%	-0.36%
ibm13	3.64%	2.15%	3.09%	1.08%	2.10%	1.02%	2.12%	0.05%
ibm14	2.94%	0.94%	2.25%	-1.23%	1.60%	0.02%	1.11%	-1.23%
ibm15	3.97%	1.96%	0.34%	0.97%	2.48%	0.85%	-0.74%	-0.07%
ibm16	2.64%	2.63%	-0.09%	11.62%	1.00%	1.38%	-1.34%	10.13%
ibm17	4.03%	1.77%	3.47%	2.74%	2.65%	0.81%	2.32%	1.59%
ibm18	1.70%	2.53%	7.67%	1.13%	0.23%	1.59%	6.62%	0.09%
Average	3.50%	2.27%	3.48%	1.67%	1.89%	0.98%	2.11%	0.60%

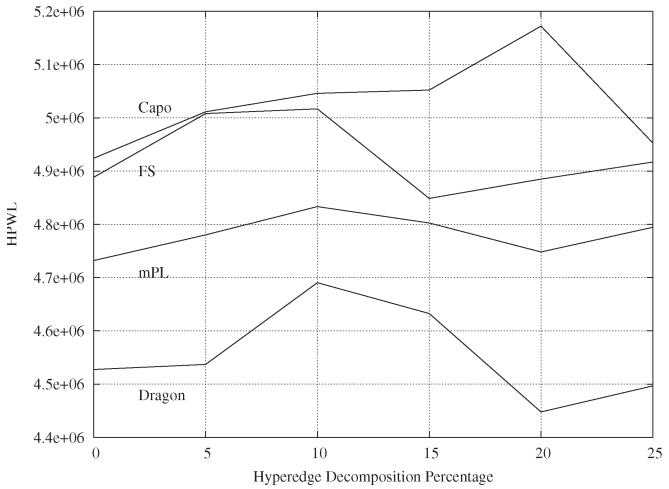

Fig. 11. Effect of zero-change hyperedge decomposition on the performance of various placers on the IBM01 benchmark. The amount of hyperedge decomposition is varied from 0% to 25% in increments of 5%.

to hyperedge decomposition is relatively "milder" than hyperedge-cardinality increase in Fig. 10. The magnitude of changes in HPWL is relatively small, and the performance is overall stable.

From the results, it is not significantly clear that the hyperedge decomposition can improve the performance of existing placers. Zero-change hyperedge decomposition simplifies a netlist by taking large hyperedges and optimally decomposing them into two or more smaller hyperedges. This simultaneously decreases the cardinality of hyperedges and increases the number of hyperedges. Our results show no improvement in performance due to the hyperedge decomposition. We can envision that perhaps by selective hyperedge decomposition and careful tuning, this transformation can be used within a placement run to simplify netlists and lead to better placements.

3) Edge Substitution: In a third series of experiments, we determine the performance of existing placers with respect to zero-change edge substitutions using procedure EDGESUB. We present our results in Table V, where we keep on substituting edges until the amount of nets increase by 10%. From the results, placers exhibit a systematic unnecessary increase in HPWL. To further study the impact of procedure EDGESUB, we apply it to the IBM01 benchmark in varying amounts, increasing the total number of nets from 0% to 25% in increments of 5%. The HPWL results are plotted in Fig. 12. From the plot, we notice that placers exhibit a consistent increase in HPWL; the larger the amount of substitution, the greater the amount of HPWL. Overall, we conclude that zero-change edge substitutions lead placers to display a suboptimal behavior.

4) Hybrid Transformations: In a fourth series of experiments, we test the performance of placers with respect to hybrid

TABLE V
HPWL DEVIATIONS IN RESPONSE TO ZERO-CHANGE EDGE SUBSTITUTION. NUMBER OF NETS INCREASES BY 10% FOR ALL BENCHMARKS. DEVIATIONS ARE CALCULATED WITH RESPECT TO EACH PLACER'S ORIGINAL PLACEMENT RUN

bench	Placer			
	Capo	FengShui	Dragon	mPL
ibm01	5.24%	6.09%	11.85%	5.76%
ibm02	5.99%	8.35%	-	8.36%
ibm03	8.66%	3.44%	4.67%	1.64%
ibm04	6.47%	4.83%	3.72%	10.61%
ibm05	2.38%	1.13%	1.99%	1.60%
ibm06	4.59%	2.62%	6.65%	4.90%
ibm07	5.50%	4.14%	7.68%	3.46%
ibm08	5.57%	2.13%	7.18%	4.54%
ibm09	9.48%	7.05%	12.64%	5.12%
ibm10	7.78%	3.80%	9.53%	3.90%
ibm11	10.34%	5.93%	16.57%	16.20%
ibm12	5.38%	4.00%	8.62%	9.16%
ibm13	9.86%	5.81%	7.91%	14.50%
ibm14	14.27%	12.03%	13.54%	12.94%
ibm15	3.49%	2.83%	8.97%	11.27%
ibm16	8.50%	5.27%	9.08%	9.54%
ibm17	6.63%	5.24%	11.02%	5.94%
ibm18	9.64%	3.40%	10.74%	6.60%
Average	7.21%	4.89%	8.96%	7.56%

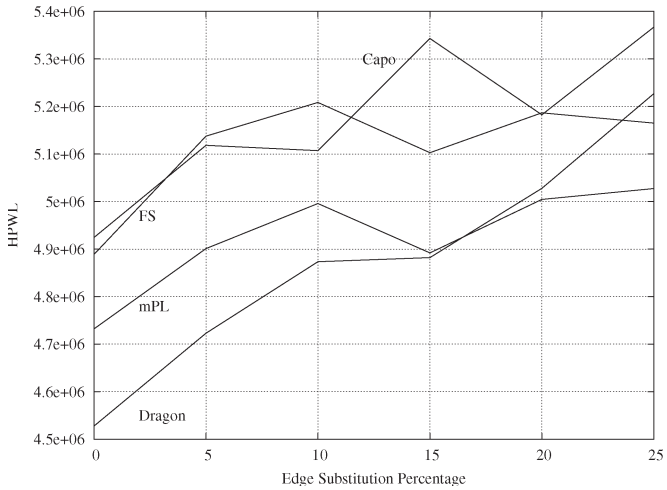

Fig. 12. Effect of zero-change edge substitution on the performance of various placers on the IBM01 benchmark. The amount of hyperedge substitution is varied from 0% to 25% in increments of 5%.

transformations. We give our results in Table VI. Given the initial placements of the various placers on different benchmarks, we apply: 1) hyperedge cardinality increase leading to a total cardinality increase of 5%; 2) edge substitution increasing the total number of nets by 5%; and 3) hyperedge decomposition

TABLE VI
HPWL DEVIATIONS IN RESPONSE TO HYBRID ZCNTS. DEVIATIONS ARE CALCULATED WITH RESPECT TO EACH PLACER'S ORIGINAL PLACEMENT RUN

bench	Placer			
	Capo	FengShui	Dragon	mPL
ibm01	8.52%	2.31%	9.05%	4.70%
ibm02	6.39%	-	-	8.12%
ibm03	11.99%	2.22%	3.26%	2.06%
ibm04	8.52%	1.59%	3.76%	11.31%
ibm05	1.67%	1.10%	1.08%	-0.31%
ibm06	5.43%	16.78%	5.02%	9.31%
ibm07	4.83%	2.83%	3.54%	2.81%
ibm08	13.78%	3.13%	9.93%	12.16%
ibm09	7.17%	7.24%	14.28%	3.56%
ibm10	10.95%	4.11%	8.92%	1.94%
ibm11	9.96%	2.04%	10.35%	11.10%
ibm12	3.55%	3.26%	10.62%	7.74%
ibm13	3.96%	6.60%	10.76%	10.02%
ibm14	11.19%	6.64%	16.35%	-
ibm15	8.25%	2.17%	7.76%	12.41%
ibm16	5.48%	3.72%	9.85%	7.27%
ibm17	9.70%	1.52%	7.77%	6.35%
ibm18	7.05%	2.17%	9.19%	12.37%
Average	7.07%	4.08%	8.32%	7.23%

further increasing the total number of nets by an additional 5%. The results show that placers again exhibit a suboptimal behavior with respect to hybrid ZCNTs.

5) Transformations Preserving Netlist Statistics: In this experiment, we apply the techniques of Section IV-E to partially evaluate the placer suboptimality while keeping a degree of realism into the transformed netlists. We start by preprocessing all netlists to reduce the amount of hyperedges and total cardinality by k and $2k$ amounts, respectively, where $k \geq 1$. This can be achieved by applying the following k times.

1) Join any two hyperedges that share a common node into one larger hyperedge. This reduces both the number of hyperedges and the total hyperedge cardinality (or total pin count) by one.
2) Reduce the hyperedge cardinality by removing a vertex from any arbitrary hyperedge with a degree greater than or equal to three.

The preprocessed netlists are placed, and zero-change edge substitution are then applied k times to yield a transformed netlist with the same amount of total nets and total cardinality as the original netlist. The new netlist thus has the same basic netlist statistics as the original, but the netlists differ structurally. HPWL suboptimality deviations of the transformed netlists are reported in Table VII, where k is set to be 10% of the number of nets of each netlist. The results demonstrate that placers exhibit consistent unnecessary suboptimality deviations

TABLE VII
HPWL DEVIATIONS IN RESPONSE TO NETLIST-STATISTICS PRESERVING ZCNTS. TRANSFORMED NETLISTS HAVE THE SAME NUMBER OF TOTAL NETS AND TOTAL CARDINALITY AS THE ORIGINAL NETLISTS

bench	Placer			
	Capo	FengShui	Dragon	mPL
ibm01	6.74%	3.98%	10.82%	0.00%
ibm02	4.90%	18.96%	0.00%	0.30%
ibm03	6.29%	5.29%	8.25%	11.17%
ibm04	5.08%	6.40%	4.00%	-
ibm05	0.91%	0.69%	0.64%	1.50%
ibm06	2.26%	4.63%	5.95%	4.20%
ibm07	4.05%	5.79%	9.13%	2.84%
ibm08	3.06%	6.52%	8.77%	6.74%
ibm09	4.95%	5.23%	7.04%	14.92%
ibm10	5.08%	4.90%	9.04%	6.82%
ibm11	6.45%	4.59%	7.97%	3.89%
ibm12	5.78%	4.69%	5.53%	6.39%
ibm13	4.64%	3.56%	9.79%	5.76%
ibm14	9.48%	11.02%	14.97%	5.98%
ibm15	7.33%	2.41%	9.98%	13.34%
ibm16	8.10%	8.46%	10.57%	14.57%
ibm17	5.09%	7.12%	7.17%	12.87%
ibm18	5.31%	3.39%	12.24%	11.94%
Average	5.31%	5.98%	7.88%	7.24%

TABLE VIII
RMST DEVIATIONS IN RESPONSE TO HYPEREDGE-CARDINALITY TRANSFORMATIONS. DEVIATIONS ARE CALCULATED WITH RESPECT TO PRECALCULATED RMST VALUES FROM THE PLACER'S ORIGINAL PLACEMENT RUN

bench	Placer			
	Capo	FengShui	Dragon	mPL
ibm01	12.20%	4.07%	15.87%	4.62%
ibm02	9.01%	2.77%	43.46%	6.42%
ibm03	15.00%	2.74%	7.75%	9.14%
ibm04	9.97%	2.36%	10.09%	5.79%
ibm05	2.82%	1.56%	2.18%	3.38%
ibm06	4.86%	2.11%	12.03%	5.83%
ibm07	5.53%	3.37%	13.39%	4.70%
ibm08	7.48%	4.79%	13.98%	4.52%
ibm09	8.76%	13.17%	7.34%	5.41%
ibm10	7.96%	3.15%	13.22%	13.93%
ibm11	7.41%	3.05%	15.54%	9.32%
ibm12	15.88%	2.47%	7.46%	8.06%
ibm13	7.14%	5.24%	16.60%	10.08%
ibm15	10.51%	2.21%	15.86%	18.96%
ibm16	11.48%	6.11%	17.59%	21.86%
ibm17	12.08%	13.35%	7.71%	15.14%
ibm18	6.87%	3.29%	17.91%	16.64%
Average	8.61%	4.21%	13.22%	9.10%

as previous experiments. This also indicates that HPWL deviations are not an artifact of increasing netlist statistics—as measured by the ratio of the number of nets to number of cells, or total pin cardinality—beyond typical values,[6] but rather inherent in the suboptimal performance of the placers.

6) Suboptimality Evaluation of RMST and RSMT Metrics: We also evaluate the performance of placers with respect to the RMST which serves as a heuristic construction to RSMT. Our experimental testbed is set up as elaborated in Section IV-F. The hyperedge-cardinality-increase procedure is applied to increase the total cardinality by 20%, and then the RMST value is calculated with respect to the original given placement. Our results are reported in Table VIII. From the results, it is clear that the placers are also suboptimal when it comes to the RMST metric, in addition to the HPWL as shown earlier.

VI. CONCLUSION AND FUTURE WORK

In this paper, we have introduced a new concept, ZCNTs, to: 1) calculate the exact suboptimality of the existing placers on artificial constructed instances and 2) calculate the partial suboptimality of placers on synthesized netlists from arbitrary nelists. While one may envision many transformations that do not change the HPWL of a given netlist, our transformations share an important property: The optimal HPWL of new netlist is not less than the original HPWL optimal value, and consequently, the placement of the new benchmarks is not "easier" than the original benchmarks. By applying our transformations and reexecuting a placer, we can interpret any deviation in HPWL results as a lower bound on the deviation from the optimal HPWL value. Our set of netlist transformations can be summarized as follows.

1) Zero-change HYPEREDGE-CARDINALITY INCREASE increases, if possible, the cardinality of hyperedges with degree ≥ 3 while leaving two-pin edges intact.
2) Zero-change HYPEREDGE DECOMPOSITION simplifies large hyperedges (when possible) of degree ≥ 3 by decomposing a larger hyperedge into two or more smaller hyperedges.
3) Zero-change EDGE SUBSTITUTION increases the number of two-pin edges if possible.

Our empirical results show that even when testing few netlist variants, we can easily find consistent deviations in placement in HPWL. From our empirical results, we make the following general remarks.

[6]The reader should notice that the synthesized netlist has larger values of statistics than the netlist used to derive it, yet its statistics are identical, i.e., typical, with respect to the original unprocessed netlist.

Remark 1: Increasing the cardinality of hyperedges leads to consistent suboptimal behavior from the placers.

Remark 2: Empirical results indicate that placers exhibit large amount of sensitivity with respect to the edge substitution.

Remark 3: The hyperedge-decomposition transformation simultaneously increases the number of hyperedges while reducing their cardinality. Thus, it has the potential to reduce the HPWL (from Remark 1), and to increase the HPWL (from Remark 2). Our experimental results show negligible change in wire length; thus, we can surmise that possible improvement in the performance of the placer due to the hyperedge-cardinality reduction is counteracted by the increase in number of hyperedges.

Remark 4: The suboptimal behavior of placers in response to zero-change transformations is not an artifact of the increase in netlist statistics of ZCNTs. Embedding ZCNTs in a flow that preserves netlist statistics clearly shows that placers exhibit the same suboptimal behavior even though basic netlist statistics are kept intact.

Remark 5: Suboptimality trends that are demonstrated for HPWL are also demonstrated for RMST.

Experimental researchers in physical design would no doubt agree that there is a tendency to tune algorithms and codes to specific benchmarks [2]. A good placer should be good not just for a single real instance but also for "similar" instances. Using our transformations allows the creation of a range of instances around any given arbitrary benchmark. Thus, for the first time, the field is afforded a means of creating "similar"' instances in a systematic way, such that the consistency of placement quality can be immediately evaluated.

Our future work will: 1) examine in further detail the source of suboptimality in current placers and how our transformations can inspire improvements in current placers; 2) investigate the use of simplifying netlist transformations such as hyperedge decomposition to produce better placements; and 3) study the possibility of calculating lower bounds on the HPWL.

REFERENCES

[1] S. Adya, I. Markov, and P. Villarrubia, "On whitespace and stability in mixed-size placement," in *Proc. IEEE Int. Conf. Comput. Aided Des.*, 2003, pp. 311–318.

[2] 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," in *Proc. ACM/IEEE Int. Symp. Phys. Des.*, 2003, pp. 95–103.

[3] J. Beardwood, J. Halton, and J. Hammersley, "The shortest path through many points," in *Proc. Cambridge Philos. Soc.*, 1959, vol. 55, pp. 299–327.

[4] A. E. Caldwell, A. B. Kahng, and I. L. Markov, "Can recursive bisection alone produce routable placements?" in *Proc. ACM/IEEE Des. Autom. Conf.*, 2000, pp 477–482.

[5] C. Chang, J. Cong, M. Romesis, and M. Xie, "Optimality and scalability study of existing placement algorithms," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 23, no. 4, pp. 537–549, Apr. 2004.

[6] C. Chang, J. Cong, and M. Xie, "Optimality and scalability study of existing placement algorithms," in *Proc. IEEE Asia and South Pacific Des. Autom. Conf.*, 2003, pp. 621–627.

[7] C.-C. Chang, J. Cong, D. Pan, and X. Yuan, "Multilevel global placement with congestion control," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 22, no. 4, pp. 395–409, Apr. 2003.

[8] C. E. Cheng, "RISA: Accurate and efficient placement routability modeling," in *Proc. IEEE Int. Conf. Comput. Aided Des.*, 1994, pp. 690–695.

[9] J. Cong, M. Romesis, and M. Xie, "Optimality and scalability study of partitioning and placement algorithms," in *Proc. ACM/IEEE Int. Symp. Phys. Des.*, 2003, pp. 88–94.

[10] J. Cong, J. R. Shinnerl, M. Xie, T. Kong, and X. Yuan, "Large-scale circuit placement," *ACM Trans. Des. Automat. Electron. Syst.*, vol. 10, no. 2, pp. 389–430, 2005.

[11] D. Cyganski, R. Vaz, and V. Virball, "Quadratic assignment problems generated with the palubetskis algorithm are degenerate," *IEEE Trans. Circuits Syst. I, Fundam. Theory Appl.*, vol. 41, no. 7, pp. 481–484, Jul. 1994.

[12] W. E. Donath, "Statistical properties of the placement of a graph," *SIAM J. Appl. Math*, vol. 16, no. 2, pp. 439–457, Mar. 1968.

[13] H. Eisenmann and F. M. Johannes, "Generic global placement and floorplanning," in *Proc. ACM/IEEE Des. Autom. Conf.*, 1998, pp. 269–274.

[14] W. Gosti, A. Narayan, R. K. Brayton, and A. L. Sangiovanni-Vincentelli, "Wireplanning in logic synthesis," in *Proc. IEEE Int. Conf. Comput. Aided Des.*, 1998, pp. 26–33.

[15] L. W. Hagen, D. J. H. Huang, and A. B. Kahng, "Quantified suboptimality of VLSI layout heuristics," in *Proc. ACM/IEEE Des. Autom. Conf.*, 1995, pp. 216–221.

[16] M. Hanan and J. M. Kurtzberg, "Placement techniques," in *In Design Automation of Digital Systems*, M. A. Breuer, Ed. Englewood Cliffs, NJ: Prentice-Hall, 1972, pp. 213–282.

[17] A. B. Kahng and S. Mantik, "On mistmatches between incremental optimizers and instance perturbations in physical design tools," in *Proc. ICCAD*, 2000, pp. 17–22.

[18] ——, "Measurement of inherent noise in EDA tools," in *Proc. Int. Symp. Quality Electron. Des.*, 2002, pp. 206–211.

[19] A. B. Kahng and S. Reda, "Placement feedback: A concept and method for better min-cut placement," in *Proc. ACM/IEEE Des. Autom. Conf.*, 2004, pp. 357–362.

[20] A. B. Kahng and G. Robins, "A new class of iterative steiner tree heuristics with good performance," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 11, no. 7, pp. 893–902, Jul. 1992.

[21] J. M. Kleinhans, G. Sigl, F. M. Johannes, and K. J. Antreich, "GORDIAN: VLSI placement by quadratic programming and slicing optimization," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 10, no. 3, pp. 356–365, Mar. 1991.

[22] B. Landman and R. Russo, "On a pin versus block relationship for partitions of logical graphs," *IEEE Trans. Comput.*, vol. C-20, no. 12, pp. 793–813, 1971.

[23] Q. Liu and M. Marek-Sadowska, "A study of netlist structure and placement efficiency," in *Proc. ISPD*, 2004, pp. 198–203.

[24] S. Ono and P. Madden, "On structure and suboptimality in placement," in *Proc. IEEE Asia and South Pacific Des. Autom. Conf.*, 2005, pp. 331–336.

[25] M. Queyranne, "Performance ratio of polynomial heuristics for triangle inequality quadratic assignment problem," *Oper. Res. Lett.*, vol. 4, no. 5, pp. 231–234, Feb. 1986.

[26] S. Sahni and T. Gonzalez, "P-complete approximation problems," *J. ACM*, vol. 23, no. 3, pp. 555–565, Jul. 1976.

[27] J. Steele, "Subadditive Euclidean functionals and non-linear growth in geometric probability," *Ann. Probab.*, vol. 9, no. 3, pp. 365–376, 1981.

[28] L. Steinberg, "The backboard wiring problem: A placement algorithm," *SIAM Rev.*, vol. 3, no. 1, pp. 37–50, 1961.

[29] D. Stroobandt, J. Depreitere, and J. V. Compenhout, "Generating new benchmark designs using a multi-terminal net model," *Integr. VLSI J.*, vol. 27, no. 2, pp. 113–129, Jul. 1999.

[30] D. Stroobandt and F. J. Kurdahi, "In the characterization of multi-point nets in electronic designs," in *Proc. IEEE Great Lakes Symp. VLSI*, 1998, pp. 344–350.

[31] D. Stroobandt, P. Verplaetse, and J. V. Campenhout, "Generating synthetic benchmark circuits for evaluating CAD tools," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 19, no. 9, pp. 1011–1022, Sep. 2000.

[32] W.-J. Sun and C. Sechen, "Efficient and effective placement for very large circuits," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 14, no. 5, pp. 349–359, 1995.

[33] V. V. Vazirani, *Approximation Algorithms*, 1st ed. New York: Springer-Verlag, 2001.

[34] M. Wang, X. Yang, and M. Sarrafzadeh, "DRAGON2000: Standard-cell placement tool for large industry circuits," in *Proc. IEEE Int. Conf. Comput. Aided Des.*, 2001, pp. 260–263.

[35] M. Yildiz and P. Madden, "Global objectives for standard-cell placement," in *Proc. IEEE Great Lakes Symp. VLSI*, 2001, pp. 68–72.

Andrew B. Kahng (A'89–M'03) received the A.B. degree in applied mathematics from Harvard College, Cambridge, MA, and the M.S. and Ph.D. degrees in computer science from the University of California at San Diego, La Jolla.

He is a Professor in the Departments of Computer Science and Engineering and Electrical and Computer Engineering, University of California at San Diego. From 1989 to 2000, he was a member of the Computer Science Faculty, University of California, Los Angeles. Since 1997, he has defined the physical design roadmap for the International Technology Roadmap for Semiconductors (ITRS) and, from 2000 to 2003, chaired the U.S. and international working groups for Design Technology for the ITRS. He has been active in the MARCO Gigascale Silicon Research Center since its inception. He was also the founding General Chair of the ACM/IEEE International Symposium on Physical Design and Co-Founded the ACM Workshop on System-Level Interconnect Planning. He has published more than 200 papers in the VLSI CAD literature. His research is mainly in physical design and performance analysis of VLSI, as well as the VLSI design–manufacturing interface. Other research interests include combinatorial and graph algorithms, as well as large-scale heuristic global optimization.

Dr. Kahng was the recipient of three Best Paper Awards and an NSF Young Investigator Award.

Sherief Reda received the B.Sc. and M.Sc. degrees in electrical and computer engineering from Ain Shams University, Cairo, Egypt, in 1998 and 2000, respectively. He is currently working toward the Ph.D. degree at University of California at San Diego, La Jolla, CA.

He has over 30 refereed publications in the areas of physical design, VLSI test and diagnosis, combinatorial optimization, and CAD for new technologies.

Mr. Reda received a Best Paper Award at DATE 2002 and the First Place Award at the ISPD 2005 Placement contest.