Fast Floorplanning by Look-Ahead Enabled Recursive Bipartitioning

Jason Cong, *Fellow, IEEE*, Michail Romesis, *Member, IEEE*, and Joseph R. Shinnerl, *Member, IEEE*

***Abstract*—A new paradigm is introduced for floorplanning any combination of fixed-shape and variable-shape blocks under tight fixed-outline area constraints and a wirelength objective. Dramatic improvement over traditional floorplanning methods is achieved by the explicit construction of strictly legal layouts for every partition block at every level of a cutsize-driven top-down hierarchy. By scalably incorporating legalization into the hierarchical flow, *post hoc* legalization is successfully eliminated. For large floorplanning benchmarks, an implementation, called partitioning to optimize module arrangement (PATOMA), generates solutions with half the wirelength of state-of-the-art floorplanners in orders of magnitude less run time. Experiments on standard Gigascale Systems Research Center benchmarks compare PATOMA to the Capo macro placer, the Traffic floorplanner, and to both the default and high-effort modes of the Parquet 4.0 floorplanner. With all blocks hard, PATOMA's average wirelength is comparable to the high-effort mode of Parquet 4.0 floorplanner and Capo, while PATOMA runs significantly faster. With all blocks soft, PATOMA produces wirelength 9% shorter on average than that of Parquet's default mode, and PATOMA runs seven times faster. For a new set of benchmarks with a mix of 500 to 2000 hard and soft blocks, PATOMA produces results with wirelengths roughly half of Parquet's, with a speedup of almost 200×.**

***Index Terms*—Floorplanning, optimization, partitioning, physical design.**

I. Introduction

GIVEN a collection of interconnected variable-dimension "soft" rectangular blocks and fixed-dimension "hard" rectangular blocks, each block with its own prescribed fixed area, the floorplanning problem is to shape the soft blocks and arrange them together with the hard blocks in the plane without overlap. The objective is typically to minimize: 1) the estimated total wirelength of the circuit; 2) its estimated timing performance; 3) the area of the smallest rectangle circumscribing the blocks; or, usually, 4) some combination of these. The most useful algorithms target wirelength minimization under a fixed outline constraint, the aspect ratios of the soft blocks constrained by simple upper and lower bounds.

Manuscript received June 16, 2004; revised December 6, 2004 and May 14, 2005. This work was supported in part by the Semiconductor Research Consortium Contract 2003-TJ-1091, by the National Science Foundation under Grant CCF-0430077, and by Magma Design Automation, Inc., under UC Micro Program 04-019. This paper was recommended by Associate Editor T. Yoshimura.

J. Cong and J. R. Shinnerl are with the Computer Science Department, University of California, Los Angeles, CA 90095 USA (e-mail: cong@cs.ucla.edu; shinnerl@cs.ucla.edu).

M. Romesis is with Magma Design Automation, Eindhoven, 5615 LE, The Netherlands (e-mail: michalis@magma-da.com).

Digital Object Identifier 10.1109/TCAD.2005.859519

Fast floorplanning is critical in the hierarchical physical design of very large-scale integration (VLSI) circuits, for two reasons. First, system designers require a means of rapidly estimating the variation in performance of alternative architectures and logic designs. Second, multilevel [10], [11], [21], [32] and mixed-size [3], [4], [20], [23], [34] placement algorithms typically solve some form of floorplanning problem at the coarsest level of approximation in order to generate an initial coarse placement for subsequent iterative refinement. With the reuse of intellectual property (IP) blocks for multimillion-gate application-specific integrated circuits (ASICs) and system-on-a-chip (SoC) designs, most modern IC designs consist of a very large number of standard cells mixed with many big macros, such as ROMs, RAMs, and IP blocks. When clusters of standard cells are placed simultaneously with macros, the clusters may be treated as soft blocks [3], [4].

Many floorplanning algorithms have been developed in recent years, varying mostly in the representation of geometric relationships among modules. They can be divided into two major categories: slicing and general algorithms. The first slicing algorithms were developed in the 1980s [28], [39]. In the 1990s, general algorithms that do not require slicing cuts became more popular, especially after the introduction of the bounded slicing grids (BSG) [27] and sequence pair [26] representations. Other nonslicing representations include transitive closure graph-based (TCG) [25], B^*-tree [12], corner block list (CBL) [17], O-tree [15], [29], and so on. With the exception of the O-tree-based work [15], [29], simulated annealing (SA) has been used to minimize the area and/or the wirelength under each of these representations. The floorplanner employing the O-tree representation [15], [29] is deterministic, but it does not support a fixed-outline constraint, and its run time is $\mathcal{O}(n^2)$ in the number of blocks.

The precise purpose and the best formulation of floorplanning are subject to some debate. In 2000, Kahng [18] cited five key problems with conventional approaches: 1) packing-driven rather than connectivity-driven algorithms and benchmarks; 2) an unnecessary restriction to rectangular shapes, including "L" or "T" shapes; 3) a lack of attention to scalability; 4) the inability to handle a fixed-outline constraint; 5) the lack of attention to the register-transfer level (RTL)-down methodology context. Recent efforts [2] to address some of these concerns have met with only partial success, highlighting in particular the difficulty of simultaneously considering 3) and 4). In this context, our paper can be viewed as a way of achieving scalability under even the tightest fixed-outline constraint. Surprisingly, our results are obtained without recourse to nonrectangular shapes. Moreover, in contrast to

0278-0070/$20.00 © 2006 IEEE

1) our method shows how scalable area-driven floorplanners can support an extremely robust connectivity driven flow.

Until a few years ago, the inherent slowness of SA was partially hidden by the lack of any need to floorplan more than 100 blocks at a time. Recently, however, growing numbers of IP blocks have increased the sizes of most floorplanning instances, prompting researchers to seek nonstochastic approaches. Ranjan *et al.* [31] propose a two-stage fast floorplanning algorithm. In the first stage, a hierarchy is generated by top-down recursive bipartitioning. Cutline orientations are selected from the bottom up in a way that keeps subregion aspect ratios close to one. In the second stage, low-temperature SA improves the wirelength by reshaping the blocks to produce a more compact layout. Finally, the total wirelength is comparable to or better than that obtained by an SA-based algorithm [39], with speed up of over 1000× in predictor mode (high speed) and 20× in constructor mode (high efsfort). More recently, a fast algorithm called Traffic [33] has been used to generate compact, low-wirelength floorplans without SA. Traffic also uses two stages. In the first stage, the blocks are divided into layers by linear multiway partitioning. In the second stage, every layer is optimized individually; the blocks in each layer are separately arranged into rows and then moved among the rows to balance row widths and reduce wirelength. In the end, pairs of rows are squeezed tightly after being transformed into trapezoids. This final step leads to very compact floorplans, but it also increases the wirelength, because the cells are ordered according to their heights.

The impressive speedups obtained by the last two algorithms raise the question of whether a fast deterministic approach can be used to replace the widely used SA engine with the same or better solution quality on fixed-outline problems. Fast recursive cutsize-driven area bisection by multilevel netlist partitioning is very successfully and widely used in large-scale placement [7], [8], [23], [37]. To date, however, the quality of bisection-based floorplanners has generally fallen short of that of the best SA-based implementations. We believe that this deficiency has been caused largely by a thorny legalization problem that traditionally accompanies the recursive bipartitioning paradigm. Legalization is the process of transforming the end result of recursive bipartitioning, or any so-called global floorplan, which may still contain overlap and/or other constraint violations, to a configuration in which all shape nonoverlap floorplanning boundary constraints are strictly satisfied. As commonly practiced, floorplanning by recursive bipartitioning makes no guarantee that the blocks assigned to a subregion can actually be shaped and arranged there without overlap. In this scenario, defining base cases may be difficult, as many base cases may fail to have legal solutions. Local refinement usually suffices to legalize, but, in general, some form of global legalization strategy is necessary to ensure proper termination. Recently, reported progress in the legalization of mixed-size [23] placements depends on the significant amount of available white space (20% or more). When the area constraints are tight, legalization based on local heuristics alone usually increases the wirelength significantly, if it succeeds at all.

The main idea of the work presented here is simple. The legalizability of a given instance is explicitly determined first, by construction, in fast and scalable run time, before any wirelength optimization is attempted.[1] Once a legal floorplan has been constructed, it is recursively replaced by legal floorplans of decreasing wirelength. While this approach might appear to sacrifice the wirelength for increased robustness on low white space fixed-outline designs, our empirical results show decreased total wirelength compared to that of the leading top-down tools. For example, our implementation beats a leading SA-based engine on the gigascale systems research center benchmarks by 10%–20% in average wirelength and by orders of magnitude in run time. Thus, the contribution of this work is not only a methodology for enhanced robustness for fixed-outline top-down floorplanning, but also a means to scalably attain solutions of superior quality, as measured by the total half-perimeter wirelength.

CPLACE [35] is the first partitioning-based placer to incorporate explicit legalization into every level of the top-down partitioning hierarchy. In CPLACE, this progressive legalization supports accurate modeling of complex constraints such as "irregular images, fixed objects, fixed IOs, large objects, timing-driven placement, and free-space distribution." But in CPLACE, legalization at each level is performed after partitioning, without any formal assurance of its success. In contrast to CPLACE, partitioning to optimize module arrangement (PATOMA) (pronounced PAH-toh-ma from the Greek for "floor") is more narrowly focused on wirelength minimization under a formal assurance of legal termination, which is obtained by applying legalization at each level before the cut-size driven partitioning.

The most similar work to our look-ahead enabled recursive bipartitioning flow is found in Capo 9.0 [4] for mixed-size placement and floorplanning. Capo 9.0 proceeds top down by cutsize-driven recursive bipartitioning until certain *ad hoc* tests suggest that newly generated subproblems may be difficult to legalize. At that point, standard cells in each subproblem are clustered, and these clusters are treated as soft macros. SA-based fixed-outline floorplanning is then attempted on the hard macros and soft clusters for the given subregion. If it succeeds, the locations of the macros are then fixed, and further refinement proceeds on the declustered soft macros. If it fails, then the subproblem is merged with its sibling, the previous partition of the parent subproblem is discarded, and floorplanning is attempted for the parent subproblem. In principle, this backtracking may continue repeatedly until some ancestor is successfully floorplanned or until failure at the top level occurs. In practice, the *ad hoc* tests used to determine when to commence floorplanning are observed to be good enough that backtracking is only rarely needed.

However, as the tests are set conservatively, their use seems likely to increase the wirelength by fixing the macro positions somewhat earlier than necessary. Moreover, while they typically prevent backtracking, they provide no guarantee of such prevention. Essentially, the *ad hoc* tests amount to a tradeoff

[1] Throughout this paper, legal means "explicitly satisfying all aspect-ratio, nonoverlap, and fixed-outline-boundary constraints." Extension of our methodology to more complex constraints such as routability and timing is left as an open question.

between: 1) robustness and 2) quality and run time. The main differences between our approach and that of Capo 9.0 can be summarized as follows.

1) In our method, neither *ad hoc* tests nor backtracking is ever needed, and thus unnecessary tradeoffs associated with such a flow are avoided.
2) Our look-ahead floorplanners explicitly compute the feasible floorplans to all subproblems to which cutsize-driven partitioning is applied, before it is applied.
3) Capo 9.0 still employs a simulated-annealing engine to do floorplanning, while our method is totally deterministic and is guaranteed scalable in all cases.
4) In our method, a failure to obtain a legal solution is extremely rare. If such a failure occurs, however, it occurs at the initialization of the flow, and therefore in much less run time than would be used by a success on the same circuit.

In its current implementation, our method generates slicing floorplans only. The generality of slicing floorplans augmented by simple compaction has been shown by Lai and Wong [24]. The results presented here confirm that a good slicing algorithm can produce superior results, even without any compaction, particularly on relatively large floorplanning instances. It is possible to extend our method to the construction of general nonslicing floorplans; however, such extensions are not considered here.

The paper is organized as follows. Section II gives an overview of our implementation called PATOMA. Section III describes a zero-dead-space (ZDS) floorplanning algorithm for soft blocks and proves its properties. Section IV presents the adaptation of ZDS to wirelength minimization in PATOMA. Section V describes the row-oriented block packing (ROB) heuristic for floorplanning a combination of hard and soft blocks. Section VI compares PATOMA's performance with that of Parquet [1] and other nonannealing-based floorplanners. The paper is concluded in Section VII.

II. Overview of PATOMA Algorithm

PATOMA attempts to minimize the total wirelength under a fixed-outline area constraint. It couples top-down cutsize-driven recursive bipartitioning with fast area-driven floorplanning on all subproblems. The flow is outlined in Fig. 1.

The initialization stage is the explicit construction of a legal floorplan in extremely fast linear time by simple block-packing heuristics. Because floorplanning is NP-hard, even in the absence of any netlist connectivity, the success of this initialization cannot be guaranteed *a priori*. In practice, however, we have not observed any failure on any circuit, even with as little as 1% white space. With the relative amount of white space held constant, a large set of blocks is empirically observed to be much easier to legally floorplan than a small set of blocks. Therefore, placing the burden of legalization at the top level of hierarchy, with all blocks present, greatly enhances robustness. The success of this initialization stage constitutes a guarantee of the legal termination of the main stage (discussed next).

The main stage is a recursive cutsize-driven bisection in which the satisfiability of all constraints is explicitly enforced at every step. At every level of the cutsize-driven area-bipartitioning hierarchy, each node corresponds to a subset of blocks assigned by terminal propagation to a specific rectangular subregion of the chip. The invariant that the parent of every subproblem has a legal floorplan is maintained as follows. Before each application of cutsize-driven bipartitioning, one of two separate fast area-driven floorplanners, ZDS or ROB, is used to check whether the given subproblem can be legalized. The fast floorplanner determines by a slicing construction whether the blocks assigned to each given subregion can in fact be shaped and laid out within that subregion without overlap. If so, then recursive cutsize-driven area bipartitioning continues in both subregions at the current level. If not, then the cutsize-driven solution at that level is discarded, and a wirelength-reducing symmetry of the previously computed legal "look-ahead" solution to the parent subproblem is used instead. Thus, a legal solution strictly satisfying all nonoverlapping, area, and shape constraints, is explicitly and separately constructed for every subproblem at each level.

The legalized parent of a failed subproblem constitutes an interrupt case because the cutsize-driven partition computed for it cannot be accepted as is. An interrupt case is not actually an end case, however, because its legalized solution is used to generate legalized child subproblems, on which the main recursion, including the cutsize-driven partitioning, continues. Because ZDS and ROB both produce slicing structures, their top-level cuts define floorplanning subproblems with known legal solutions. Cutsize-driven partitioning coupled with subproblem legalization resumes recursively on each of these legalized child subproblems until single-block base cases are reached.

The feedback provided to the recursive bisection by the explicit legalization at each level enables it to proceed significantly longer, deepening the partitioning hierarchy, and thereby improving the wirelength quality.

The area-driven look-ahead floorplanners determine whether a legal solution exists for a given fixed-shape subregion and block subset. These algorithms must be fast and must usually find legal solutions if they exist. The first area-driven floorplanner ZDS is based on a recent study [13] of sufficient conditions for ZDS floorplanning of soft blocks. ZDS is used only when all the blocks in the subregion are soft. Otherwise, a second area-driven floorplanner based on ROB is used. ROB is somewhat similar to Traffic [33]; however, it handles both soft and hard blocks under a fixed-outline constraint. Both ZDS and ROB perform well in reasonable run time. They are reviewed in Sections IV and V below.

PATOMA uses hMetis [22], the well-known cutsize-driven multilevel hypergraph partitioning package. Terminal propagation [14] is used to account for connections between partitions. For a given user-specified area-imbalance tolerance t, hMetis attempts to find a partition $V = V_1 \cup V_2$ of the vertex set V subject to the constraints $0.5 - t \leq a(V_i)/a(V) \leq 0.5 + t, i = 1, 2$, where $a(V_i)$ denotes the sum of the areas of the vertices in V_i, such that the number of hyperedges containing vertices in both V_1 and V_2 is minimized. Different area-imbalance tolerances t were tried for hMetis in PATOMA—0.05, 0.1, 0.16, and 0.20. The value $t = 0.1$ consistently produced the

Algorithm 2.1: PATOMA Floorplanning Algorithm
input: Set of blocks $\mathcal{S} = \{r_1, \ldots, r_m\}$; netlist; aspect ratio constraints for each block, rectangle R of fixed shape.

Each node of the partitioning tree is a set of blocks paired with a subregion.

Initialization Stage.
Generate the root node $(\mathcal{S}, R)$ at level $i = 1$, and a legal floorplan for the root. If this legalization fails, report a failure and quit. Otherwise, legal termination is guaranteed.

Main Stage.
```
while there are still blocks to be placed
    while there are unvisited nodes at level i
        Select unvisited node n = (S_n, R_n) of level i.
        Use terminal propagation to model connections
            between b_i ∈ S_n and b_j ∉ S_n.
        Call hMetis to partition S_n into disjoint subsets S_n1 and
            S_n2, resp. assigned subregions R_n1, R_n2 of R_n.
        done := false.
        repeat
            remark Binary search for cutline position.
            for i = 1, 2
                if (all blocks in S_ni are soft)
                    fit[i] := ZDS(S_ni, R_ni).
                else fit[i] := ROB(S_ni, R_ni).
                end if
            end for
            if (fit[j] and not fit[k], j, k ∈ {1, 2})
                slide the cutline toward R_nj
            else done := true.
            end if
        until (done or cutline search limit reached)
        if (fit[1] and fit[2])
            Create child nodes n_1 and n_2 of n.
            Store the solutions from or ZDS or ROB for possible
            future use.
        else replace the hMetis bipartitioning of (S_n, R_n) with
            a bipartitioning derived from earlier application
            of ZDS or ROB.
        end if
    end while
    i := i + 1.
end while
```
output: A floorplan of $\mathcal{S}$ in R satisfying all area and aspect-ratio constraints.

Fig. 1. PATOMA floorplanning algorithm.

best results in our experiments, yielding total wirelengths up to 3%–5% shorter than the other choices, i.e., neither of the two block subsets produced by hMetis is allowed to hold more than 60% of the total area of all blocks in both subsets.

Using feedback from the look-ahead floorplanners, PATOMA redistributes white space in order to make the result of cutsize-driven partitioning legalizable as often as possible. The exact location of the cutline is initially set in direct proportion to the total areas of the blocks in every partition. If a legal solution is found initially for R_1 but not for its sibling R_2, it may still be possible to find a legal solution for both partitions by moving the white space from R_1 to R_2, i.e., by moving the cutline away from R_2 and toward R_1. Candidate cutline positions can be generated by binary search, as long as each cutline position results in a legal solution in at least one of the partitions.

Various uses of multilevel partitioning and white space redistribution by cutline adjustment are already familiar in min-cut placers Capo [3], [4], [9] and Feng Shui [5], [23], [40], [41]. Feedback to min-cut placement by relocation of ambiguous terminals has been considered by Kahng and Reda [19]. What distinguishes PATOMA's use of feedback from these other forms is both its source—area-driven look-ahead floorplanners, and its result, the guaranteed legality of its final floorplan.

III. ZDS Algorithm

The ZDS algorithm generates a ZDS floorplan for a set of soft blocks inside a fixed-dimension region. The aspect ratios of the blocks of the final floorplan are uniformly bounded by the properties of the blocks. We will show that these bounds are realistic and satisfy the constraints for most existing benchmarks. Previous work on this subject either analyzed the theoretical upper bounds on the total area achieved by slicing floorplans of soft blocks [30], [42], or generate ZDS floorplans with no consideration of aspect-ratio constraints [36], [38]. The section also proves the properties of the algorithm.

The ZDS algorithm used here is based on a recursive top-down area bipartitioning. At each step, the blocks in a region are separated into two groups such that the groups' total areas are as nearly equal as possible. That is, given a group of blocks $a_1, \ldots, a_n$, index j is found such that

Algorithm 3.1: Top-Down Fixed-Outline ZDS Floorplanning.

input

(i) Rectangles $r_i, \ldots, r_n$ with respective areas $a_i \geq \ldots \geq a_n$.
(ii) Rectangular region R of area $A = \sum_{k=i}^{n} a_k$ and dimensions $\ell \times w$, with $\ell / w = \rho(\mathcal{R}) \geq 1$.
(iii) $\gamma \geq 1$.

end input

if the largest block r_i satisfies $a_i \geq \frac{1}{\gamma} A$ **then**

(i) make the length of one side of r_i equal to w.

(ii) place r_i with this shape against a side of R of length w.

remark In the analysis, let $R(r_i)$ denote this R.

(iii) **if** $n > i$ **then**

Replace R by the part of R not used by r_i.

Replace i by $i+1$.
if $n == i$ **then** place the last block in R and **return**. **end if**

else return

end if

remark If R contains more than 2 blocks, we place at most one block before repartitioning.

end if

if $n > i$ **then**

1. Select $j \in \{i, \ldots, n-1\}$ such that

$$D_j = |\sum_{k=i}^{j} a_k - \sum_{k=j+1}^{n} a_k| \quad \text{is minimized.}$$

Let $A_j = \sum_i^j a_k$ and $\bar{A}_j = \sum_{j+1}^n a_k \equiv A - A_j$.

2. Cut R parallel to its shorter side into two rectangular subregions R_j and $\bar{R}_j$ of areas A_j and $\bar{A}_j$ respectively. Assign $r_i, \ldots, r_j$ to R_j and $r_{j+1}, \ldots, r_n$ to $\bar{R}_j$.
3. Recur on the subregions R_j and $\bar{R}_j$.

end if

Fig. 2. Top-Down fixed-outline ZDS floorplanning algorithm.

$D_j = |\sum_1^j a_i - \sum_{j+1}^n a_i|$ is minimized. The region is then cut parallel to its shorter side such that each group fits exactly into one of the regions. Cutting parallel to the shorter side keeps the aspect ratios of the subregions bounded in terms of the area variation among the blocks. Blocks are placed once they fill a sufficient fraction of their subregions; this fraction is expressed as the reciprocal of the parameter γ introduced below.

Fig. 2 shows the pseudocode for the ZDS algorithm. The notation is as follows. Given N rectangles $r_1, \ldots, r_N$ with fixed areas $a_1 \geq a_2 \geq \cdots \geq a_N$ but variable lengths ℓ_i and widths w_i, we seek to arrange them without overlap in a given rectangle $\mathcal{R}$ of area $\mathcal{A} = \sum_1^N a_i$, such that the aspect ratios

$$\rho_i = \rho(r_i) = \max\left(\frac{\ell_i}{w_i}, \quad \frac{w_i}{\ell_i}\right)$$

are bounded close to 1.[2] The rectangle $\mathcal{R}$ is the floorplanning region. The rectangles r_i are the blocks.

[2]By this definition of ρ_i, which we use for the remainder of the paper, the aspect ratio of every block is always at least one.

Algorithm 3.1 is parameterized by $\rho(\mathcal{R}) \geq 1$ and $\gamma \geq 1$. By construction, Algorithm 3.1 has the following property.

Theorem 3.1: For $\rho(\mathcal{R}) \geq 1$ and $\gamma \geq 1$, Algorithm 3.1 generates a slicing floorplan with ZDS. ■

A trace of the algorithm on a simple five-block example is illustrated in Fig. 3.

Although Algorithm 3.1 can accept any values $\rho(\mathcal{R}) \geq 1$ and $\gamma \geq 1$ as input, the analysis in the next section shows that the generated floorplans display attractive properties for certain choices of $\rho(\mathcal{R})$ and γ. Specifically, if we define

$$\beta = \max_i \frac{a_i}{a_{i+1}} \tag{1}$$

and let

$$\gamma = \max\{2, \ \beta\} \quad \text{and} \quad \rho(\mathcal{R}) \in [1, \ \gamma + 1] \tag{2}$$

then Algorithm 3.1 produces a ZDS floorplan with all blocks' aspect ratios in $[1, \ \gamma + 1]$.

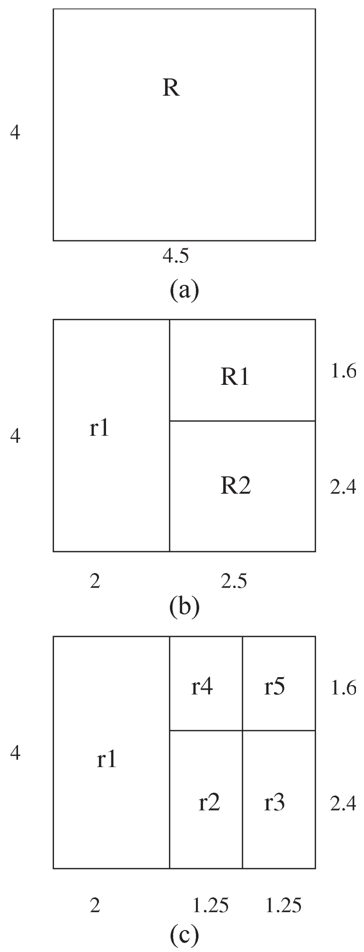

Fig. 3. (a) Creation of ZDS benchmark with five blocks, where $a_1 = 8$, $a_2 = a_3 = 3$, $a_4 = a_5 = 2$. In this case, $\gamma = 2.67$. Region R has area 18 and is initialized to an aspect ratio of 1.125. (b) Since $a_1 \geq 18/2.67$, block r_1 is placed. After partitioning of remaining blocks, region R_1 is assigned to blocks r_2 and r_3, and region R_2 to blocks r_4 and r_5. (c) Final placement of all blocks. Maximum aspect ratio is $2 \leq \gamma + 1$.

A. Analysis

The utility of Algorithm 3.1 rests in the fact that the maximum aspect ratio of any block in the floorplan it generates is guaranteed to lie within a single small interval of the form $[1,\ \gamma + 1]$, when γ is defined as in (2). In other words, the maximum aspect ratio of any block will not be large, as long as the areas of the blocks decay relatively gradually from largest to smallest.

These facts are established here, under Assumptions 3.1 below.

Assumptions 3.1: Block-locking threshold and floorplanning-region aspect-ratio bound:

1) for β as defined in (1), $\gamma \geq \max\{2,\ \beta\}$;
2) the aspect ratio of the floorplanning region satisfies $\rho(\mathcal{R}) \leq \gamma + 1$.

In Assumption 3.1, 1) can be rephrased as follows: the threshold fraction of the subregion area that a block must occupy in order to be shaped and locked in place is not set above the minimum of 1/2 and $1/\beta$. Although these assumptions are stronger than necessary to achieve ZDS and acceptably bounded block aspect ratios, they are not very restrictive on the sets of block areas that may be considered. In fact, for realistic circuits, the algorithm can be extended to handle most abrupt jumps in block area as well as regions $\mathcal{R}$ with $\rho(\mathcal{R}) > \gamma + 1$ (see Section III-C). However, such extensions are not used in PATOMA.

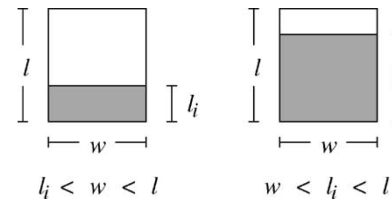

Fig. 4. Aspect ratio of block (shaded) compared to aspect ratio of its enclosing subregion.

B. Derivation of Aspect-Ratio Bound

Throughout this section, we consider the properties of Algorithm 3.1 under Assumptions 3.1. The first lemma shows that, in order to bound the aspect ratios of the blocks, it suffices to bound the aspect ratios of the regions in which they are placed.

Lemma 3.1: The aspect ratio ρ_i of any placed block r_i satisfies

$$\rho_i \leq \max\{\gamma,\ \rho\left(R(r_i)\right)\}$$

where $\rho(R(r_i))$ denotes the aspect ratio of the smallest subregion in which r_i is placed.

Proof: Suppose that subregion R has an aspect ratio $\rho = \rho(R)$, if R contains just one block, then that block r_i will also have $\rho(r_i) = \rho$. Hence, R contains more than one block. By Algorithm 3.1, the blocks $\{r_i, \ldots, r_p\}$ in R form a contiguous subsequence of the original set of blocks $\{r_1, \ldots, r_N\}$ and, therefore, satisfy the area decay bounds $a_k \geq a_{k+1} \geq a_k/\gamma$. Moreover, the block r_i placed in R will have one of its side lengths w equal to the shorter side length of R, as shown in Fig. 4. Let ℓ_i denote the length of the other side of r_i, and let ℓ denote the length of the longer side of R.

First, suppose $\ell_i < w$, because the algorithm requires the area a_i of r_i be at least $1/\gamma$ times the area of R; the other side ℓ_i of r_i is at least $1/\gamma$ times the length of the longer side ℓ of R. Hence

$$\frac{w}{\ell_i} \leq \frac{w}{\frac{\ell}{\gamma}} = \frac{\gamma}{\rho} \leq \gamma$$

since $\rho \geq 1$. Second, suppose $\ell_i \geq w$, because the blocks r_k in R satisfy $a_k \geq a_{k+1} \geq a_k/\gamma$, the subregion of R containing these other blocks must occupy an area at least $1/(\gamma+1)$ times the area of R, and therefore $\ell_i \leq (\gamma/(\gamma+1))\ell$. Hence

$$\frac{\ell_i}{w} \leq \frac{\gamma}{(\gamma+1)}\frac{\ell}{w} = \frac{\gamma}{\gamma+1}\rho < \rho. \tag{3}$$

■

The next lemma bounds the aspect ratio of sibling subregions in terms of their area ratio and the aspect ratio of their common parent subregion.

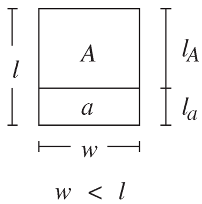

Fig. 5. Aspect ratios of two sibling subregions compared to aspect ratio of parent subregion.

Lemma 3.2: Suppose the subregion R is partitioned into subregions R_1 and R_2 with areas A_1 and A_2, let

$$y = \max\left\{\frac{A_1}{A_2},\ \frac{A_2}{A_1}\right\}.$$

Then

$$\max\{\rho(R_1),\ \rho(R_2)\} = \max\left\{\frac{y+1}{\rho(R)},\quad \frac{y}{y+1}\rho(R)\right\}.$$

Proof: Following the notation in Fig. 5, let $A \equiv A_1$, $\rho_A = \rho(R_1)$, $a \equiv A_2$, $\rho_a = \rho(R_2)$, and assume without loss of generality that $A > a$, so that $y = A/a$. The longer side of R has length ℓ, and the shorter side has length w. Now

$$A + a = (y+1)a = \ell w$$

and therefore

$$\ell_a = \frac{a}{w} = \frac{\ell}{(y+1)} \quad \text{and} \quad \ell_A = y\ell_a = \frac{y}{y+1}\ell.$$

If $\rho_A \geq \rho_a$, then $\rho_A = \ell_A/w$ (otherwise, $\rho_A = w/\ell_A < w/\ell_a = \rho_a$); hence, $\rho_A = (y/(y+1))\rho(R)$. Similarly, if $\rho_a \geq \rho_A$, then $\rho_a = w/\ell_a$, and therefore $\rho_a = (y+1)/\rho(R)$. ■

Lemma 3.3: With the notation in Algorithm 3.1

$$D_j \leq \begin{cases} a_j, & \text{if } A_j > \bar{A}_j \\ a_{j+1}, & \text{if } A_j \leq \bar{A}_j \end{cases}.$$

Proof: Suppose $A_j > \bar{A}_j$; in this case, $D_j = A_j - \bar{A}_j$. First, observe that $D_j < 2a_j$; otherwise, $D_{j-1} < D_j$ — contradicting the minimality of D_j. Next, suppose that $D_j > a_j$, i.e.,

$$2a_j > \sum_1^j a_i - \sum_{j+1}^N a_i > a_j.$$

But, then subtracting $2a_j$ gives

$$0 > \sum_1^{j-1} a_i - \sum_j^N a_i > -a_j$$

and hence

$$0 < \sum_j^N a_i - \sum_1^{j-1} a_i < a_j$$

which is another contradiction to the minimality of D_j. The case $A_j \leq \bar{A}_j$ is similar. ■

Theorem 3.2: In Algorithm 3.1, under Assumptions 3.1

$$\frac{1}{\gamma} \leq \frac{A_j}{\bar{A}_j} \leq \gamma.$$

Proof: As before, let $A = A_j + \bar{A}_j$. We first express the conclusion of the theorem in terms of D_j. The conclusion can be rewritten as

$$\frac{1}{\gamma+1}A \leq A_j \leq \frac{\gamma}{\gamma+1}A. \tag{4}$$

But, the conclusion can also be written as $1/\gamma \leq (\bar{A}_j/A_j) \leq \gamma$ or

$$\frac{1}{\gamma+1}A \leq \bar{A}_j \leq \frac{\gamma}{\gamma+1}A. \tag{5}$$

From (4) and (5), the conclusion can be equivalently expressed as

$$D_j \leq \frac{\gamma-1}{\gamma+1}A.$$

Now, for all $\gamma \geq 2$, it suffices to show that $D_j \leq A/3$ or

$$\frac{1}{3}A \leq A_j \leq \frac{2}{3}A.$$

Now, if $j \geq 3$, then Lemma 3.3 ensures that $D_j \leq a_3$, and because $a_1 \geq a_2 \geq a_3 \geq \cdots \geq a_n > 0$, it follows that $a_3 \leq A/3$. The same result also holds if $j = 2$ and $A_j \leq \bar{A}_j$. Therefore, we need only consider the following two cases.

Case 1: $j = 2$, and $a_1 + a_2 > \sum_3^N a_i$.

By Lemma 3.3, $a_1 + a_2 - \sum_3^N a_i \leq a_2$; hence

$$a_1 \leq \sum_3^N a_i$$

and since $a_1 \geq a_2$, we have $a_2 \leq \sum_3^N a_i$ as well. Thus, in this case

$$\frac{1}{\gamma} \leq 1 \leq \frac{A_j}{\bar{A}_j} \equiv \frac{a_1 + a_2}{\sum_3^N a_i} \leq 2 \leq \gamma.$$

Case 2: $j = 1$.

By Assumptions 3.1

$$\frac{A_j}{\bar{A}_j} \equiv \frac{a_1}{\sum_2^N a_i} \leq \frac{a_1}{\frac{a_1}{\gamma} + \sum_3^N a_i} < \gamma.$$

Hence, it suffices to show that $A_j/\bar{A}_j > 1/\gamma$ in this case. If $A_1 \geq \bar{A}_1$, then we are done. Hence, we assume that $a_1 < \sum_2^N a_i$. Lemma 3.3 therefore gives $D_1 \equiv \sum_2^N a_i - a_1 \leq a_2$; i.e.,

$$a_1 \geq \sum_3^N a_i. \tag{6}$$

Now if $a_2 > \sum_3^N a_i$, then

$$\frac{a_1}{\sum_2^N a_i} = \frac{a_1}{a_2 + \sum_3^N a_i} \geq \frac{a_1}{a_2 + a_2} \geq \frac{a_1}{a_1 + a_1} = \frac{1}{2} \geq \frac{1}{\gamma}.$$

Otherwise, if $a_2 \le \sum_3^N a_i$, then (6) implies that

$$\frac{a_1}{\sum_2^N a_i} \ge \frac{\sum_3^N a_i}{a_2 + \sum_3^N a_i} = \frac{1}{\left(\frac{a_2}{\sum_3^N a_i}\right) + 1} \ge \frac{1}{2} \ge \frac{1}{\gamma}.$$

■

From Lemma 3.2 and Theorem 3.2, we immediately obtain the following bound.

Corollary 3.1: Suppose subregion R is partitioned into subregions R_1 and R_2, then

$$\max\{\rho(R_1),\ \rho(R_2)\} \le \max\left\{\frac{\gamma+1}{\rho(R)},\ \frac{\gamma}{\gamma+1}\rho(R)\right\}.$$

Theorem 3.3: Under Assumptions 3.1, the result of Algorithm 3.1 is a slicing floorplan with ZDS and every block's aspect ratio bounded above by $\gamma + 1$.

Proof: Follows directly from Corollary 3.1 and Assumptions 3.1, by induction. ■

C. Remarks and Extensions

The implementation of ZDS in PATOMA follows the above description and has been found to be very effective and robust in practice. For all the Gigascale Systems Research Center (GSRC) benchmarks, the value of γ is always two. It is interesting that for the soft block version of these examples, the aspect-ratio constraint for all the blocks is in the range [0.3, 3], which means that ZDS can find a ZDS solution for all of them. This shows that the conditions required by ZDS are realistic and the algorithm itself can be used for the manipulation of soft blocks in a floorplanning algorithm.

Nevertheless, a number of obvious improvements can be made, and the possibilities are intriguing. Some of these are sketched in this section.

1) Weakening Subregion Aspect-Ratio Constraint: It is easy to show that 2) of Assumption 3.1 can be replaced by the weakest possible restriction, namely that $\rho(\mathcal{R})$ is small enough that the largest block r_1 can be shaped and placed in $\mathcal{R}$ without violating the aspect-ratio constraint of r_1, when all other blocks are ignored.[3] Assuming that the maximum aspect ratio allowed for any block is $\gamma + 1$, this requirement can be written $w(\mathcal{R}) \ge \sqrt{a_1/(\gamma+1)}$, where $w(\mathcal{R})$ denotes the length of the shorter side of $\mathcal{R}$, and a_1 is the area of r_1. This result follows by a simple inductive argument, a sketch of which follows. Throughout this sketch, we maintain 1) of Assumption 3.1; namely, that $\gamma = \max\{2,\ \max a_i/a_{i+1}\}$. The weakening of 1) of Assumption 3.1 is discussed subsequently.

Suppose $\rho(\mathcal{R}) \gg 1$, but $w(\mathcal{R}) \ge \sqrt{a_1/(\gamma+1)}$. The ZDS algorithm will repeatedly cut $\mathcal{R}$, by lines parallel its shorter side, into subregions of ever smaller aspect ratios (i.e., ever closer to one). Once cuts are chosen in the direction orthogonal to $\mathcal{R}$'s shorter side, it must be the case that 2) of Assumption 3.1 holds for the subregions these orthogonal cuts separate and, therefore, all subsequent subregions they contain. If, on the other hand, a block is shaped and placed before any cut is made in the direction orthogonal to $\mathcal{R}$'s shorter side, it is placed flush against that shorter side of $\mathcal{R}$, and since the length $w(\mathcal{R})$ of that side satisfies $w(\mathcal{R}) \ge \sqrt{a_1/(\gamma+1)}$, the aspect-ratio constraint of the block will be satisfied.

[3]If the aspect-ratio bounds on the different blocks are different, we must check that the restriction on $\rho(\mathcal{R})$ is strong enough that any block will fit in $\mathcal{R}$ when all other blocks are ignored.

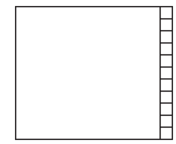

Fig. 6. Block aspect ratios may all remain small, even when some subregion has a large aspect ratio.

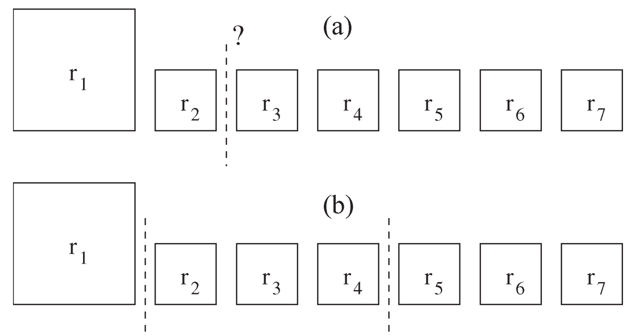

Fig. 7. Given blocks $r_1, \ldots, r_7$ with $a_1 = 4a_2$ and $a_2 = a_3 = \cdots = a_7$, (a) default area-balancing cut shown in will typically result in large aspect ratio for r_2, which is unwisely paired with r_1. (b) Better set of cuts. First cut is made between r_1 and r_2, because a_1/a_2 is both maximal and larger than two. Next cut, between r_4 and r_5, is made in order to balance cluster sizes.

2) ZDS Floorplanning of Mixed-Size Blocks: The assumption that $\gamma \ge 2$ presents no practical restriction on the sets of blocks that may be considered. It just means that the upper bound on block aspect ratios guaranteed by the analysis here for the given algorithm is at least 3. That is, consecutive-pairwise area bounds tighter than two (e.g., $a_i/a_{i+1} \le 1.5$) are not guaranteed to reduce the maximum aspect ratio below what can be attained with $a_i/a_{i+1} \le 2$.

Similarly, a large value of γ does not necessarily indicate any large aspect ratios in the final floorplan, as Fig. 6 illustrates. In the figure, one large block occupies one subregion, and several small blocks occupy another subregion. Although the area ratio of the subregions may be arbitrarily large, the presence of sufficiently many small blocks used to fill the small subregion prevents any single block's aspect ratio from becoming large.

For some designs, the presence of a few very large or very small blocks may result in a large value of γ, if γ is defined simply as $\max\{2, \max_i a_i/a_{i+1}\}$. In such cases, a simple partitioning step can be used to reduce this value significantly (as illustrated in Fig. 7).

The idea is to partition the sorted list of blocks into clusters such that the areas of any two consecutive blocks in the same cluster are comparable, i.e., within a factor of two. Areas of clusters, computed simply as sums of contained blocks' areas, must also be somewhat comparable, but, as explained below, the limit on cluster area ratios is higher than that for the areas of blocks within any cluster. Each cluster is simply a contiguous subsequence of the list of blocks ordered by the nonincreasing area. The partition is therefore simply a set $\mathcal{P} \subset \{1, \ldots, n\}$ of "cluster cuts" in the ordered list of blocks; the precise choice of $\mathcal{P}$ is discussed briefly. Treating the clusters in this partition as individual blocks, we find a ZDS floorplan for them. The aspect ratios of the clusters in this floorplan will be governed by the maximum ratio of successive cluster areas, when these are ordered nonincreasing. A separate fixed-outline ZDS floorplan can then be separately computed for each individual cluster; the largest aspect ratio ρ_i of any block in any cluster $C \subset \{r_1, \ldots, r_n\}$ is governed entirely by the blocks in that cluster

$$\max_{r_i \in C} \rho_i \leq 1 + \max \left\{ 2,\ \max_{r_i \in C} \frac{a_i}{a_{i+1}} \right\}.$$

The choice of $\mathcal{P}$ is motivated by the desire to reduce the upper bound $\gamma + 1$ on the maximum aspect ratio of any block as much as possible. By the result of Section III-C1, the only restriction on the aspect ratios of the clusters is that the largest blocks they contain can be shaped and placed within them. As long as this condition holds, then the conclusion of Theorem 3.3 holds with $\beta \equiv \max_{i \notin \mathcal{P}} a_i/a_{i+1}$ and $\gamma = \max\{\beta,\ 2\}$. Therefore, the priority in selecting cluster cuts should be placed on separating blocks of disparate areas rather than on balancing cluster areas, i.e., $\mathcal{P}$ can contain as many i's as possible such that $a_i/a_{i+1} > 2$, subject only to the constraint that the largest block in each cluster can be legally shaped and placed within that cluster's assigned subregion of the ZDS cluster floorplan.

This restriction on cluster shapes can be enforced by imposing a low-enough upper bound on the ratios of successive cluster areas, when these are sorted in nonincreasing order. However, many clusters will likely be able to take aspect ratios far above the bound without adversely affecting any blocks' aspect ratios; therefore, it is not generally necessary to enforce the bound strictly. In general, it is enough that clusters assigned to the same ZDS subregion satisfy area-ratio bounds specific to their own set of aspect-ratio constraints. That is, clusters can also be partitioned, and the restrictions on their area ratios are considerably looser than those on the blocks.

For generality and efficiency, partitioning can proceed adaptively and recursively. For example, an initial partition $\mathcal{P}_0 \subset \{1, \ldots, n\}$ may be kept small by limiting the cuts to only the most abrupt area changes. After a ZDS floorplan for the clusters defined by $\mathcal{P}_0$ has been computed, some clusters, e.g., C, may still contain two consecutive blocks r_i and r_{i+1} for which a_i/a_{i+1} is large. If so, then the blocks within C can be partitioned, and the ZDS floorplan of the resulting clusters in C can be computed. Partitioning of the blocks within these clusters and ZDS floorplanning of the result can continue recursively, as necessary, to reduce the largest ratio a_i/a_{i+1} of the consecutive blocks in the same cluster, until all clusters either have no such large ratios or consist of three blocks or fewer. This approach amounts to multilevel ZDS floorplanning.

Although it is trivial to construct examples where this multilevel ZDS approach will be useless (e.g., when $N = 2$), on practical examples with large N, the reduction in γ will likely be considerable.

3) Nonslicing Extensions: For circuits with one block much smaller in area than all the others, however, the top-down slicing ZDS algorithm presented here obviously cannot produce a good aspect ratio for the smallest block, irrespective of whether the blocks are partitioned into clusters as described above. In these cases, nonslicing ZDS extensions must be considered, e.g., a wheel with the smallest block at its center, as shown in Fig. 8. For the blocks $r_1, \ldots, r_5$ of fixed areas $a_1 \geq \cdots \geq a_4 \gg a_5$ arranged as shown in a subregion of fixed shape $w \times l$, it can be shown that the horizontal lengths l_1 and l_3 of blocks r_1 and r_3 must satisfy the following 2×2 nonlinear system of equations:

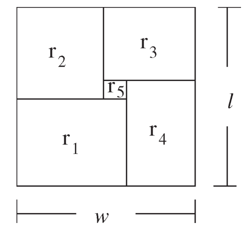

Fig. 8. When one block is much smaller in area than all others, only nonslicing ZDS floorplan can possibly satisfy useful bounds on aspect ratios of all blocks.

$$\begin{aligned} \frac{a_3}{l_3} + \frac{a_4}{w - l_1} &= l \\ \frac{a_1}{l_1} + \frac{a_3}{l_3} + \frac{a_5}{l_1 + l_3 - w} &= l. \end{aligned} \tag{7}$$

Two other analogous systems can be written for the two other arrangements (symmetries) of the blocks in the wheel (these are obtained by interchanging the relative positions of blocks r_2 and r_3 or of blocks r_3 and r_4). The existence of such a nonslicing floorplan in which all subregions' or blocks' aspect-ratio constraints are satisfied can be determined by computing and examining all solutions of (7) and, if necessary, its two analogs. Further details of such extensions are left as open questions.

IV. Wirelength-Aware ZDS Floorplanning

As described in the previous section, the ZDS algorithm ignores the wirelength. In this section, the extensions of ZDS for PATOMA including the wirelength consideration are presented.

PATOMA extends the original ZDS algorithm in two ways. First, available dead space is used to increase the frequency with which ZDS satisfies all aspect-ratio constraints. Let $\rho_{\max}$ denote the maximum aspect ratio allowed for any block. When $\gamma + 1 \leq \rho_{\max}$, success of ZDS is guaranteed, because the aspect ratios of the subregions for which ZDS is called are also in the range $[1/\rho_{\max}, \rho_{\max}]$, by the partitioning and cutline decisions made at the higher levels of the hierarchy. When $\gamma + 1 > \rho_{\max}$, the effective value of γ can be reduced by padding some of the blocks by dead space. If the reduction in γ is not enough to guarantee success, the ZDS algorithm is applied anyway, because its conditions for the creation of a legal solution are sufficient but not necessary. Second, in the original ZDS algorithm, the side of a subregion in which a block or block subset is placed is left unspecified. In PATOMA, when ZDS must be used instead of cutsize-driven bipartitioning

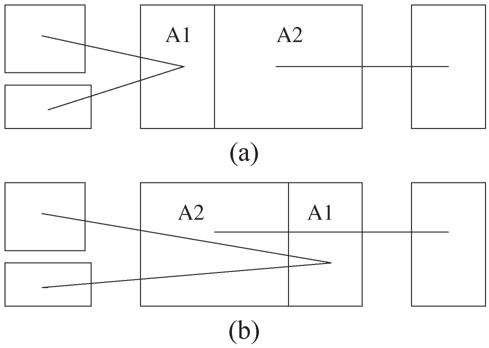

Fig. 9. Example of wirelength consideration during enhanced ZDS algorithm. Two alternatives for relative location of blocks in regions A_1 and A_2. First alternative is selected because of its better wirelength.

to guarantee legalizability of the resulting subproblems, each block subset is placed in the subregion side that reduces the total lengths of connections between blocks in the subset and other blocks. An example is shown in Fig. 9.

V. Row-Based Floorplanning

The ROB heuristic is used by PATOMA for floorplanning a combination of fixed and variable-dimension blocks. It is similar to Traffic [33] in that it organizes the blocks by rows according to their dimensions; however, it satisfies a fixed-outline constraint and handles both hard and soft blocks. Assume that given a set of blocks to be placed in a region with fixed height H and fixed width W, if $H > W$, the blocks will be organized in rows; otherwise, in columns. By organizing the blocks in rows along the shorter subregion dimension, there is room to pack more rows and, therefore, a wider variety of block heights can be efficiently supported. For the rest of this section, we assume, for simplicity, that the blocks are packed in rows.

ROB ignores connectivity. It consists of two stages. In the first stage, the blocks are grouped into rows according to their dimensions. In the second stage, emptier rows are merged with fuller rows until all rows fit inside the given fixed-shape region. The outline of the ROB algorithm is shown in Fig. 10. During the first stage, the blocks are considered one by one and either added to the existing rows or used to create new ones. Hard blocks are considered first. For every block, if one of its dimensions matches the height of an existing row and its addition to that row does not create overflow, it is placed there. Otherwise, a new row is generated with height equal to the smaller dimension of the block. Soft blocks are considered next. As they can be reshaped, they are more likely to match the height of an existing row. When a block can fit in multiple rows, the shortest one is preferred. If no such row can be found, a new one is generated with a height equal to the smallest possible dimension of the block.

At the end of the first stage, a set of rows has been generated. Each row width is less than the fixed width W of the region,[4] but it is possible that the sum of the row heights is larger than the fixed height H of the region. In the second stage,

Algorithm 5.1: ROB Floorplanning Algorithm

```
input: Set of blocks a_1, ..., a_n; netlist; block aspect
ratio constraints; region R with height H, width W; H ≥ W.
output: true if and only if a legal solution is found.
Order the blocks first by flexibility (hard blocks first)
and second by area (larger area first).
for every block B
    From the current list of partially filled rows, form a list
    of those into which B can fit, called B's row candidates.
    if (B's list of candidate rows is empty) then
        Create a new row with height equal to the smallest
        valid dimension of B. Add B to the new row.
    else
        Add B to the shortest in height of its candidate rows.
    end if
end for
Curr_height := sum of row heights
if Curr_height ≤ H return true
Order the rows according to height (taller first)
for every row r in the height-ordered list of rows
    Let S denote the list of all rows shorter in height than r
    Order S by width (emptier rows first)
    for every row r' in S
        for every block a in r'
            Move a to r if no overflow in r. The orientation or
            shape of a is selected so that the height of a is as
            close to the height of r as possible without exceeding it.
            Update Curr_height.
            if Curr_height ≤ H return true
        end for
    end for
end for
return false
```

Fig. 10. ROB floorplanning algorithm.

some rows are eliminated by redistributing the blocks one by one. The rows are scanned in a decreasing height order. The blocks from the rows shorter than the currently selected one are added to the selected row where possible. Priority is given to rows of smallest width. When a block is moved to another row, it is allowed to be rotated or reshaped for the purpose of matching the height of its new row as closely as possible without exceeding it. The procedure is repeated until either all the rows have been scanned or enough rows have been eliminated such that the sum of the heights of the remaining rows is less than H. In the first case, the algorithm ends without finding a legal solution, while in the second it reports a success.

When legalizability of a cutsize-driven partition of a given subproblem cannot be ensured, ROB's solution to that subproblem is employed instead, by interpreting it as a partition. Since the solution of ROB is organized in rows (columns), it is guaranteed to have at least one slicing horizontal or vertical cut that can be used as the cutline for a bipartitioning of the blocks. The bipartitionings generated by these cuts are compared with their symmetric ones for wirelength, and the best bipartitioning is selected to replace the infeasible hMetis solution.

In ROB's current prototype implementation within PATOMA, its worst case order is $\mathcal{O}(nr)$, where n is the number of blocks and r is the number of rows. However, a more careful implementation can reduce this asymptotic to $\mathcal{O}(nk)$, for some fixed constant k limiting the length of the list of candidate rows to which a block can be added.

[4]Unless there is a block with both its dimensions larger than W. In that case, a legal placement under the area constraints obviously cannot be found.

TABLE I
COMPARISON OF PATOMA WITH PARQUET 4.0 ON GSRC BENCHMARKS WITH ALL BLOCKS SOFT

Circuit	PATOMA		Parquet 4.0 (default)		Parquet 4.0 (high-effort)	
	WL	Runtime (sec)	WL Ratio	Runtime Ratio	WL Ratio	Runtime Ratio
n300	566242	4	1.36	40	0.99	219
n200	505736	3	1.15	19	1.07	305
n200b	509403	3	1.17	14	1.01	310
n200c	490249	3	1.14	15	1.10	288
n100	283452	2	1.12	5	1.03	482
n100b	312666	2	1.03	5	1.03	389
n100c	280278	2	1.11	5	1.11	319
n50	180115	1	1.05	2	0.97	667
n50b	217241	1	1.04	2	0.97	703
n50c	200759	1	1.06	2	1.05	599
n30	156921	1	1.06	1	0.97	511
n30b	163635	1	1.04	1	0.97	760
n30c	192709	1	1.00	1	0.97	647
n10	52258	1	0.97	1	0.97	781
n10b	62178	1	1.02	1	0.94	807
n10c	51958	1	1.03	1	0.95	503
Averages			1.09	7	1.01	518

VI. EXPERIMENTS AND RESULTS

We compare PATOMA to: 1) Parquet 4.0 [1], a state-of-the-art SA-based floorplanner using the sequence pair geometric representation; 2) Traffic [33]; 3) Hierarchical plus simulated annealing (HierPlusSA) the fast floorplanner of Ranjan *et al.* [31]; and 4) Capo 9.3 [4], which can perform placement of standard cells and macros. For a fair comparison, all experiments were performed on the same machine, a 2.4-GHz Pentium IV running RedHat Linux 8.0. We compared on four sets of benchmarks. For all the experiments, the floorplanners are trying to minimize the total half-perimeter wirelength under a fixed-outline boundary constraint. For the benchmarks with soft macros, we compare only to Parquet 4.0, because in addition to the high-quality floorplans it produces, it is, as far as we know, the only freely available package online that can consider both fixed-outline constraints and soft blocks. We run Parquet 4.0 in two modes. The first mode is the default and is very fast, due to a shorter simulated-annealing schedule that degrades the wirelength quality. The second mode is a high-effort mode, where we impose a time limit of 1 h to allow SA to attain a better solution.

A. *Soft Blocks Only*

The first set of benchmarks includes the GSRC circuits (size 10–300 blocks), where all the blocks are soft. Pad locations fix the amount of the given white space at approximately 55%–75%. In order to reduce the wirelength, however, PATOMA restricts its floorplan to an inner core region with 30% white space for all-hard-block examples and 0% white space for all-soft-block examples. In the all-soft-block examples, PATOMA uses only the ZDS algorithm and not the ROB to enforce the legalizability of all floorplanning subproblems. All blocks are allowed to be reshaped with any aspect ratios in [1/3, 3]. Interrupt cases, i.e., subproblems with nonlegalizable child subproblems, are observed to be small, containing fewer than 2 blocks on average and at most 27 on any instance. This result confirms that ZDS failures are quite rare and occur only for relatively small sets of blocks. The recursive cutsize-driven flow thus proceeds on average almost all the way to individual blocks. The results are shown in Table I. The default mode of Parquet 4.0 produces results that are 9% higher (21% higher for the largest four benchmarks) in wirelength than PATOMA, while its runtime is 7× slower (22× slower for the largest benchmarks). The high-effort mode of Parquet 4.0 is 1% worse in wirelength (4% worse for the largest benchmarks) and 518× slower (281× slower for the largest benchmarks) than PATOMA.

B. *Hard Blocks Only*

The second set of experiments includes the same GSRC benchmarks, but with all blocks of given fixed dimensions as specified in the benchmarks. In these examples, PATOMA uses only ROB and not ZDS to enforce the legalizability of floorplanning subproblems, because all blocks are hard. Table II shows the results of this set of experiments. It includes the results of Capo 9.3, which, although a placer, can perform hard block floorplanning. Capo 9.3 is based on Parquet to perform SA for floorplanning subproblems, but its top-down methodology allows it to perform much better than plain Parquet. On these benchmarks, PATOMA produces results of 2% lower wirelength than the default mode of Parquet 4.0 (11% lower for the four largest benchmarks only), with a speedup of 15× (48× for the largest benchmarks), and of 3% higher wirelength than the high-effort mode of Parquet 4.0, with an average speedup of 570×. However, when only the largest benchmarks are considered, PATOMA is better than the high-effort version of Parquet by 5%, while running 407× faster. PATOMA also produces better results than Capo 9.3 by 1% with an average speedup of 3× (5× for the largest benchmarks).

The third set of experiments includes the four largest GSRC circuits (200–300 blocks), all blocks hard but without pads. PATOMA was compared with Traffic and FFPC on these benchmarks, as these floorplanners do not use pads or shape soft blocks.[5] Table III lists the results of these experiments.

[5]FFPC does support *semisoft* blocks, whose shapes must be selected from a fixed finite set. Currently, PATOMA supports only soft blocks, not semisoft. It is readily adapted to directly handle semisoft blocks as well.

TABLE II
COMPARISON OF PATOMA WITH CAPO 9.3 AND PARQUET 4.0 ON GSRC BENCHMARKS WITH ALL BLOCKS HARD

Circuit	PATOMA		Capo 9.3		Parquet 4.0 (default)		Parquet 4.0 (high effort)	
	WL	Runtime (sec)	WL Ratio	Runtime Ratio	WL Ratio	Runtime Ratio	WL Ratio	Runtime Ratio
n300	599821	4	1.00	5	1.14	75	1.10	329
n200	530325	3	1.01	5	1.11	38	1.03	341
n200b	531403	3	1.01	4	1.11	38	1.02	519
n200c	508983	3	1.01	5	1.08	42	1.04	438
n100	300477	2	1.02	4	1.06	10	0.96	501
n100b	319748	2	1.01	4	1.05	10	0.96	503
n100c	298201	2	0.98	3	1.04	10	0.98	402
n50	190610	1	1.01	2	1.00	4	0.94	730
n50b	229479	1	1.03	2	0.95	4	0.94	516
n50c	209762	1	1.00	4	1.01	4	0.95	827
n30	159724	1	1.05	2	1.02	1	0.98	538
n30b	169037	1	1.01	1	0.97	1	1.02	440
n30c	198793	1	1.00	1	0.98	1	0.94	853
n10	54477	1	1.02	1	0.92	1	0.91	572
n10b	63366	1	1.04	1	0.99	1	0.93	1117
n10c	54596	1	1.01	1	0.95	1	0.82	48
Averages			1.01	3	1.02	15	0.97	570

TABLE III
COMPARISON OF PATOMA WITH TRAFFIC [33] AND FFPC WITH ALL BLOCKS HARD. PADS ARE IGNORED

Circuit	PATOMA		FFPC		Traffic	
	WL	Run-time (sec)	WL Ratio	Run-time Ratio	WL Ratio	Run-time Ratio
n300	351463	4	1.03	6	1.71	1
n200	234085	3	1.09	9	1.52	1
n200b	193008	3	1.02	4	1.65	1
n200c	226147	3	0.96	5	1.53	1
Averages			1.03	6	1.60	1

TABLE IV
COMPARISON OF PATOMA WITH PARQUET 4.0 ON FLOORPLANNING BENCHMARKS DERIVED FROM IBM CIRCUITS. PARQUET 4.0 FAILED TO FIND LEGAL SOLUTION FOR CIRCUIT HB12

Circuit	# soft blocks	# hard blocks	PATOMA		Parquet 4.0	
			WL	Run-time (sec)	WL Ratio	Run-time Ratio
HB01	665	246	3.10E+06	12	2.32	132
HB02	1200	271	6.42E+06	35	2.11	136
HB03	999	290	9.80E+06	26	1.68	168
HB04	1289	295	1.10E+07	33	2.03	192
HB05	564	0	1.46E+07	20	1.55	75
HB06	571	178	8.83E+06	25	1.89	80
HB07	829	291	1.70E+07	41	2.08	132
HB08	968	301	1.88E+07	47	2.06	169
HB09	860	253	1.87E+07	34	1.94	158
HB10	809	786	5.39E+07	46	1.46	366
HB11	1124	373	2.89E+07	53	2.38	202
HB12	582	651	5.86E+07	42	-	-
HB13	530	424	3.69E+07	41	1.99	187
HB14	1021	614	6.60E+07	90	2.52	250
HB15	1019	393	9.04E+07	102	2.17	210
HB16	633	458	1.03E+08	84	1.88	208
HB17	682	760	1.46E+08	107	1.91	380
HB18	658	285	7.32E+07	71	2.33	179
Averages					2.02	190

FFPC's wirelength is 3% longer than PATOMA's, on average, while its run time is 6× longer. With Traffic's run-time limit set to PATOMA's run time, Traffic's average total wirelength is 60% longer than PATOMA's. Extending Traffic's run-time limit beyond PATOMA's was attempted but was not observed to improve its wirelength quality; the results of these experiments were not tabulated.

C. Hard and Soft Blocks Together

In the fourth set of experiments, we generated large-scale floorplanning benchmarks from the IBM/ISPD98 suite [6] that include both hard and soft blocks on a fixed die with 20% whitespace. The soft blocks are clusters of standard cells generated by the First Choice clustering heuristic [22]. The hard blocks are the same macros as in the original benchmarks. The allowed range of aspect ratios for the soft blocks was set at [1/3, 3]. The sizes of the benchmarks range from 500 to 2000 blocks. We called this suite of benchmarks the HB-suite (hybrid blocks). These benchmarks are available online [16]. The characteristics of the benchmarks are shown in Table IV. The same table lists the results of both PATOMA and Parquet 4.0 (default mode) on them. For these examples, Parquet's wirelength is on average 102% higher than PATOMA's, while it is 190× slower. We conclude that PATOMA has a big advantage over other floorplanning methodologies for large benchmarks and benchmarks that include soft blocks. But even for smaller benchmarks with hard blocks only, PATOMA is competitive with more traditional SA-based floorplanners.

Fig. 11 shows a floorplan for the benchmark extracted from ibm01. All the benchmarks have an available deadspace of 20%. This is a reasonable value for modern fixed-size designs. The results show that the recursive-bisection flow of the PATOMA algorithm is very effective and efficient compared to other floorplanning algorithms.

VII. CONCLUSION

A new paradigm has been presented for floorplanning a combination of fixed-and variable-dimension blocks under a wirelength objective and a fixed-outline constraint. By constructively ensuring satisfiability of all constraints at each level by fast area-driven heuristics, recursive cutsize-driven

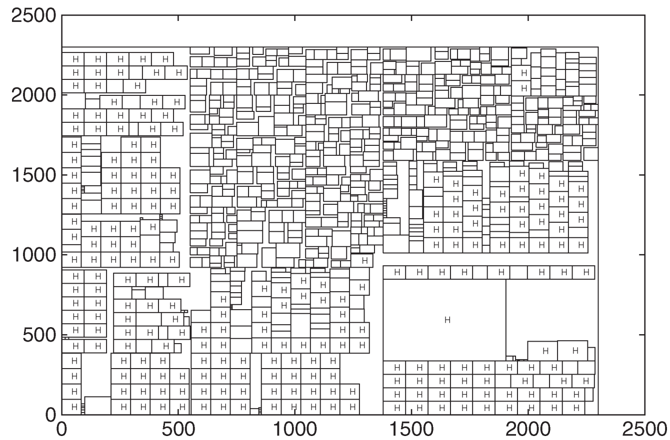

Fig. 11. Sample output of PATOMA algorithm for benchmark generated from ibm01. Hard blocks are shown with "H" mark.

bipartitioning is allowed to proceed longer, and *post hoc* legalization is eliminated. The resulting flow is scalable and produces superior wirelengths in orders of magnitude less run time than a leading SA-based tool. In the current implementation, feedback to the bipartitioning is used only to adjust cutline positions, and only until satisfiability is ensured. More elaborate feedback can be expected to improve results significantly on benchmarks with more than 500 blocks.

REFERENCES

[1] S. Adya and I. Markov, "Fixed-outline floorplanning through better local search," in *Proc. Int. Conf. Computer Design*, Austin, TX, 2001, pp. 328–334.

[2] ——, "Fixed-outline floorplanning: Enabling hierarchical design," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 11, no. 6, pp. 1120–1135, Dec. 2003.

[3] S. N. Adya and I. L. Markov, "Consistent placement of macro-blocks using floorplanning and standard-cell placement," in *Proc. Int. Symp. Physical Design*, San Diego, CA, Apr. 2002, pp. 12–17.

[4] S. N. Adya, S. Chaturvedi, J. A. Roy, D. A. Papa, and I. L. Markov, "Unification of partitioning, placement and floorplanning," in *Proc. Int. Conf. Computer-Aided Design*, San Jose, CA, 2004, pp. 12–17.

[5] A. R. Agnihotri, M. C. Yildiz, A. Khatkhate, A. Mathur, S. Ono, and P. H. Madden, "Fractional cut: Improved recursive bisection placement," in *Proc. Int. Conf. Computer-Aided Design*, San Jose, CA, 2003, pp. 307–310.

[6] C. J. Alpert, "The ISPD98 circuit benchmark suite," in *Proc. Int. Symp. Physical Design*, Monterey, CA, 1998, pp. 80–85.

[7] M. A. Breuer, "Min-cut placement," *J. Des. Autom. Fault-Toler. Comput.*, vol. 1, no. 4, pp. 343–362, Oct. 1977.

[8] A. E. Caldwell, A. B. Kahng, and I. L. Markov, "Improved algorithms for hypergraph partitioning," in *Proc. Asia South Pacific Design Automation Conf.*, Yokohama, Japan, 2000, pp. 661–666.

[9] ——, "Can recursive bisection produce routable placements?" in *Proc. 37th IEEE/ACM Design Automation Conf.*, Los Angeles, CA, 2000, pp. 477–482.

[10] T. F. Chan, J. Cong, T. Kong, and J. R. Shinnerl, "Multilevel circuit placement," in *Multilevel Optimization in VLSICAD*, J. Cong and J. R. Shinnerl, Eds. Boston, MA: Kluwer, 2003.

[11] T. F. Chan, J. Cong, and K. Sze, "Multilevel generalized force-directed method for circuit placement," in *Proc. Int. Symp. Physical Design*, San Francisco, CA, 2005, pp. 185–192.

[12] Y. C. Chang, Y. W. Chang, G. Wu, and S. Wu, "B*-trees: A new representation for non-slicing floorplans," in *Proc. Design Automation Conf.*, Los Angeles, CA, 2000, pp. 458–463.

[13] J. Cong, G. Nataneli, M. Romesis, and J. Shinnerl, "An area-optimality study of floorplanning," in *Proc. Int. Symp. Physical Design*, Phoenix, AZ, 2004, pp. 78–83.

[14] A. E. Dunlop and B. W. Kernighan, "A procedure for placement of standard-cell VLSI circuits," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. CAD-4, no. 1, pp. 92–98, Jan. 1985.

[15] P. Guo, C. Cheng, and T. Yoshimura, "An O-tree representation of non-slicing floorplan and its applications," in *Proc. Design Automation Conf.*, New Orleans, LA, 1999, pp. 268–273.

[16] Hybrid-Block Floorplanning Benchmarks [Online]. Available: http://cadlab.cs.ucla.edu/cpmo/HBsuite.html/

[17] X. Hong, S. Dong, G. Huang, Y. Cai, C.-K. Cheng, and J. Gu, "Corner block list representation and its application to floorplan optimization," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 51, no. 5, pp. 228–233, May 2004.

[18] A. B. Kahng, "Classical floorplanning harmful?" in *Proc. Int. Symp. Physical Design*, San Diego, CA, Apr. 2000, pp. 207–213

[19] A. B. Kahng and S. Reda, "Placement feedback: A concept and method for better min-cut placements," in *Proc. Design Automation Conf.*, San Diego, CA, Jun. 2004, pp. 357–362.

[20] A. B. Kahng and Q. Wang, "An analytic placer for mixed-size placement and timing-driven placement," in *Proc. Int. Conf. Computer-Aided Design*, San Jose, CA, Nov. 2004, pp. 565–572.

[21] ——, "Implementation and extensibility of an analytic placer," in *Proc. Int. Symp. Physical Design*, Phoenix, AZ, 2004, pp. 18–25.

[22] G. Karypis, R. Aggarwal, V. Kumar, and S. Shekhar, "Multilevel hypergraph partitioning: Application in VLSI domain," in *Proc. 34th ACM/IEEE Design Automation Conf.*, Anaheim, CA, 1997, pp. 526–529.

[23] A. Khatkhate, C. Li, A. R. Agnihotri, S. Ono, M. C. Yildiz, C.-K. Koh, and P. H. Madden, "Recursive bisection based mixed block placement," in *Proc. Int. Symp. Physical Design*, Phoenix, AZ, 2004, pp. 84–89.

[24] M. Lai and D. F. Wong, "Slicing tree is a complete floorplan representation," in *Proc. Design, Automation, Test Eur.*, Munich, Germany, 2001, pp. 228–232.

[25] J. Lin and Y. Chang, "TCG: A transitive closure graph-based representation for non-slicing floorplans," in *Proc. Design Automation Conf.*, Las Vegas, NV, 2001, pp. 764–769.

[26] H. Murata, K. Fujiyoshi, S. Nakatake, and Y. Kajitani, "Rectangle-packing-based module placement," in *Proc. Int. Conf. Computer-Aided Design*, San Jose, CA, 1995, pp. 472–479.

[27] S. Nakatake, K. Fujiyoshi, H. Mirata, and Y. Kajitani, "Module packing based on the BSG-structure and IC layout applications," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 17, no. 6, pp. 519–530, Jun. 1998.

[28] R. Otten, "Automatic floorplan design," in *Proc. Design Automation Conf.*, Las Vegas, NV, 1982, pp. 261–267.

[29] Y. Pang, C.-K. Cheng, and T. Yoshimura, "An enhanced perturbing algorithm for floorplan design using the o-tree representation," in *Proc. ISPD*, San Diego, CA, 2000, pp. 168–173.

[30] H. Peixoto, M. Jacome, A. Royo, and J. Lopez, "A tight upper bound for slicing floorplans," in *Proc. IEEE VLSI*, Calcutta, India, Jan. 2000, pp. 280–285.

[31] A. Ranjan, K. Bazargan, S. Ogrenci, and M. Sarrafzadeh, "Fast floorplanning for effective prediction and construction," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 9, no. 2, pp. 341–351, Apr. 2001.

[32] M. Sarrafzadeh, M. Wang, and X. Yang, *Modern Placement Techniques*. Boston, MA: Kluwer, 2002.

[33] P. Sassone and S. K. Lim, "A novel geometric algorithm for fast wire-optimized floorplanning," in *Proc. Int. Conf. Computer-Aided Design*, San Jose, CA, 2003, pp. 74–80.

[34] C. Sechen, "Chip planning, placement, and global routing of macro/custom cell integrated circuits using simulated annealing," in *Proc. Design Automation Conf.*, Anaheim, CA, 1988, pp. 73–80.

[35] P. Villarrubia, G. Nusbaum, R. Masleid, and E. T. Patel, "IBM RISC chip design methodology," in *Proc. ICCD*, Cambridge, MA, Oct. 1989, pp. 143–147.

[36] K. Wang and W.-K. Chen, "A class of zero wasted area floorplan for VLSI design," in *Proc. ISCAS*, Chicago, IL, 1993, pp. 1762–1765.

[37] M. Wang, X. Yang, and M. Sarrafzadeh, "Dragon2000: Standard-cell placement tool for large circuits," in *Proc. IEEE/ACM Int. Conf. Computer-Aided Design*, San Jose, CA, Apr. 2000, pp. 260–263.

[38] S. Wimer, I. Koren, and I. Cederbaum, "Floorplans, planar graphs, and layouts," *IEEE Trans. Circuits Syst.*, vol. 35, no. 3, pp. 267–278, Mar. 1988.

[39] D. F. Wong and C. L. Liu, "A new algorithm for floorplan design," in *Proc. Design Automation Conf.*, Las Vegas, NV, 1986, pp. 101–107.

[40] M. C. Yildiz and P. H. Madden, "Global objectives for standard cell placement," in *Proc. 11th Great-Lakes Symp. VLSI*, West Lafayette, IN, 2001, pp. 68–72.

[41] M. C. Yildiz and P. H. Madden, "Improved cut sequences for partitioning-based placement," in *Proc. Design Automation Conf.*, Las Vegas, NV, 2001, pp. 776–779.

[42] F. Y. Young and D. F. Wong, "How good are slicing floorplans?" in *Proc. Int. Symp. Physical Design*, Napa Valley, CA, 1997, pp. 144–149.

Jason Cong (S'88–M'90–SM'96–F'00) received the B.S. degree in computer science from Peking University, China, in 1985, and the M.S. and Ph.D. degrees in computer science from the University of Illinois, Urbana–Champaign, in 1987 and 1990, respectively.

Currently, he is a Professor and Codirector of the very large scale integration (VLSI) CAD Laboratory in the Computer Science Department, University of California, Los Angeles. His research interests include synthesis and layout of VLSI circuits, highly scalable VLSI design algorithms and tools, design and synthesis of programmable circuits and systems, and system-on-a-chip (SoC) designs. He has published over 220 research papers and led over 30 research projects.

Dr. Cong served on technical program committees and executive committees of many professional conferences, such as the Digital/Analog Converter (DAC) International Conference on Computer Aided Design (ICCAD) and the International Symposium on Circuits and Systems (ISCAS), including serving as the General Chair of the 1993 ACM/SIGDA Physical Design Workshop, the Program Chair and General Chair of the 1997 and 1998 International Symposiums on field programmable gate arrays (FPGAs), respectively, the Program Cochair of the 1999 International Symposium on Low-Power Electronics and Designs, and the Program Cochair and General Cochair of the Asia and South Pacific Design Automation Conference (ASPDAC)'2003 and 2005. He served as an Associate Editor for IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION SYSTEMS, from 1999 to 2002, and has been an Associate Editor of *ACM Transaction on Design Automation of Electronic Systems* since 1995. He served on the ACM SIGDA Advisory Board from 1993 to 1999. He served on the Board of Governors of the IEEE Circuits and Systems Society from 2001 to 2004. He received the Best Graduate Award from the Peking University, in 1985, and the Ross J. Martin Award for Excellence in Research from the University of Illinois, in 1989. He received the NSF Young Investigator Award, in 1993, the Northrop Outstanding Junior Faculty Research Award from the University of California Los Angeles (UCLA), in 1993, and the ACM/SIGDA Meritorious Service Award, in 1998. He has received three best paper awards, including the 1995 IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN Best Paper Award, the 2005 International Symposium on Physical Design Best Paper Award, and the 2005 *ACM Transaction on Design Automation of Electronic Systems* Best Paper Award. He also received the Semiconductor Research Corporation (SRC) Inventor Recognition Award and the SRC Technical Excellence Award, both in 2000.

Michail Romesis (S'01–M'04) received the B.S. degree in electrical and computer engineering from the National Technical University of Athens, Athen Greece, in 1999, and the M.S. and Ph.D. degrees in computer science from the University of California, Los Angeles, in 2001 and 2005, respectively.

He is currently working at Magma Design Automation, Eindhoven, The Netherlands. His research interests include very large-scale integration computer-aided design algorithms for placement and floorplanning.

Dr. Romesis received the Dimitris Chorafas Foundation Award, in 2003.

Joseph R. Shinnerl (M'00) is a Lecturer and Assistant Researcher in the Department of Computer Science, University of California, Los Angeles (UCLA). His interests include multiscale optimization and interior-point methods for linear and nonlinear programming. Since 1998, he has worked with Tony Chan the UCLA Mathematics Department, and Jason Cong the UCLA Computer Science Department on multiscale algorithms for large-scale circuit placement, including the mPL placement package. He is Coeditor of the book "*Multiscale Optimization in VLSICAD*."