

An Area-Optimality Study of Floorplanning *

Jason Cong, Gabriele Nataneli, Michail Romesis, and Joseph R. Shinnerl

UCLA Computer Science Department

{cong,michail,shinnerl}@cs.ucla.edu gabrielenataneli@yahoo.com

ABSTRACT

A novel algorithm for rectangular floorplanning with guaranteed 100% area utilization is used to construct new sets of floorplanning benchmarks. By minimizing the maximum block aspect ratio subject to a zero-dead-space constraint, example zero-dead-space (ZDS) floorplans matching the area profiles of any existing floorplanning benchmark circuits can be constructed. A mathematical analysis shows that the aspect ratios of the ZDS benchmarks' blocks are uniformly bounded within [1, 3] in most cases. Block packings produced by the Parquet, B*-tree, TCG-S, and BloBB packages on these new benchmarks are compared to the optimal-area floorplans produced by the ZDS algorithm.

Categories and Subject Descriptors

B.7.2 [Integrated Circuits]: Design Aids—*placement and routing*; G.4 [Mathematical Software]: Algorithm Design and Analysis; J.6 [Computer-Aided Engineering]: Computer-Aided Design

General Terms

Algorithms, Design

Keywords

Placement, Floorplanning, Block Packing, Aspect Ratios, Benchmarking, Optimality, Area Partitioning

1. INTRODUCTION

How much room for improvement is there in existing physical design algorithms? The answers for floorplanning, partitioning, placement, and routing algorithms will help researchers to determine the phases of the physical design process that need the most attention. Previous studies on the optimality of placement algorithms ([5], [7]) have shown a

*Financial Support from Semiconductor Research Consortium Contract 2003-TJ-1019 is gratefully acknowledged.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

ISPD April 18–21, 2004, Phoenix, Arizona, USA.

Copyright 2004 ACM 1-58113-817-2/04/0004 ...\$5.00.

significant gap (up to 2× in some cases) in the performance of existing algorithms from the optimal solutions. Similar studies for partitioning algorithms [7] showed a much smaller gap. In this paper, a new technique for benchmarking the area optimality of existing floorplanning algorithms is introduced, analyzed, and applied to a few available leading floorplanning packages.

Floorplanning is a critical phase in the physical design of VLSI circuits when a hierarchical design methodology is used. The floorplanning problem is to shape and place N rectangular modules of given areas in the plane. The floorplanning objective is typically to minimize either (a) the area of the smallest rectangle circumscribing the blocks, (b) the total wirelength of the circuit, (c) its timing performance, or, most likely, (d) some combination of these. With these objectives, constraints on the aspect-ratios of the blocks must be imposed in order to ensure physically useful solutions. If the blocks must have a fixed aspect ratio they are called *hard*, otherwise they are called *soft* and are allowed to be shaped.

The importance of good floorplanning algorithms has increased significantly in the last few years, as design hierarchy and IP blocks are used ever more intensively to reduce design complexity. However, their evaluation has been increasingly difficult, for two main reasons.

- (i) The effects of a good floorplanning algorithm are evident in later stages of the physical design process.
- (ii) The available benchmarks are old and small in size.

While the performance of most algorithms is excellent on these small benchmarks, there is a valid concern about the performance of the algorithms on contemporary, more complex designs with many blocks (say, 100 – 1000). Such instances may arise, for example, when certain levels of the logic hierarchy are flattened [16].

Many floorplanning algorithms have been developed in recent years, varying mostly in the representation of the geometric relationships among the modules. They can be divided into two major categories: slicing and non-slicing algorithms. The first slicing algorithms were developed in the 80's ([17], [21]). In the 90's non-slicing algorithms became more popular, especially after the introduction of the Sequence Pair [14] representation. Other non-slicing representations include TCG [11], TCG-S [12], B*-tree [6], CBL [10], O-tree [9], BSG [15], and so on.

In this paper, a simple slicing algorithm that guarantees a ZDS floorplan while attempting to minimize the maximum block aspect ratio is defined and analyzed. The analysis

shows that the block aspect ratios obtained are uniformly bounded in terms of the area variation of the given blocks. Existing methods typically attempt to maximize area utilization subject to aspect-ratio bounds. By recasting the block aspect ratios computed by the ZDS algorithm as constraints, we interpret the ZDS floorplans it produces as new optimal-area benchmarks for existing algorithms. Thus, new floorplanning benchmarks with known ZDS solutions and realistic, uniform bounds on all blocks' aspect ratios are generated. Experiments then compare the results of a few state-of-the-art floorplanning packages on the new benchmarks. Early results also suggest that further refinements of the ZDS algorithm may render it useful in and of itself as a component in floorplanning and placement algorithms. Previous works on this subject ([23], [19]) analyzed the theoretical upper bounds on the total area achieved by slicing floorplans of soft blocks.

The main limitation of this work to date is that it does not consider wirelength. Wirelength optimality in combination with area optimality will be addressed in future work.

The remainder of this paper is organized as follows. Section 2 presents a ZDS algorithm. Section 3 outlines the proof that, under very mild regularity assumptions, this algorithm creates benchmarks with small, uniform bounds over all blocks' aspect ratios. Section 4 compares experimental results of some leading floorplanning algorithms on our area-optimal benchmarks. Section 5 describes a few possible uses of the ZDS framework beyond benchmarking. The paper is concluded in Section 6.

2. BENCHMARK CONSTRUCTION

The input to the ZDS generator is a set of block areas with no aspect ratio constraints at all; these may be extracted from existing benchmarks. The output of the ZDS generator is a floorplan, i.e., a set of block shapes and locations, with realistic uniform bounds on the aspect ratios of all modules. Compared to the traditional floorplanning formulation, the objective and constraints for the ZDS algorithm are swapped. Rather than minimize unused area subject to block-aspect-ratio constraints, the benchmark generator implicitly seeks to minimize block aspect ratios while maintaining full area utilization at every step. To get a benchmark, we simply interpret the ZDS block-shape output as input constraints for algorithms in the traditional area-optimizing vein. If the benchmark block shapes are held fixed as equality constraints, then the ZDS floorplan is a hard-block packing benchmark. If the benchmark block shapes are allowed to vary within some interval containing the ZDS block shapes, then the ZDS floorplan is a soft-block packing benchmark.

The ZDS algorithm used here is based on recursive top-down area bipartitioning. At each step, the blocks in a region are separated into two groups such that the groups' sizes are as nearly equal as possible. 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 aspect ratios of 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.

Figure 1 shows the pseudocode for this ZDS algorithm:

ALGORITHM 2.1. *Top-Down ZDS Floorplanning.*

input

- (i) Rectangles r_1, \dots, r_n with areas $a_1 \geq \dots \geq a_n$.
- (ii) Rectangular region R of area $A = \sum_1^n a_i$ and dimensions $\ell \times w$, with $\ell/w = \rho(\mathcal{R}) \geq 1$.
- (iii) $\gamma \geq 1$.

end input

if the largest block r_1 satisfies $a_1 \geq \frac{1}{\gamma}A$ **then**

- (i) make the length of one side of r_1 equal to w .
- (ii) place r_1 with this shape against one side of R .

remark In the analysis, let $R(r_1)$ denote this R .

- (iii) **if** $n > 1$ **then**
 - Replace R by the part of R not used by r_1 .
 - Reindex $\{r_2, \dots, r_n\}$ to $\{r_1, \dots, r_{n-1}\}$.
 - if** $n == 2$ **then**
 - place the last block in R and **return**.
 - else** Replace n by $n - 1$.
 - 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 $n > 1$ **then**

1. Select $j \in \{1, \dots, n\}$ such that

$$D_j = \left| \sum_1^j a_i - \sum_{j+1}^n a_i \right| \text{ is minimized.}$$

Let $A_j = \sum_1^j a_i$ and $\bar{A}_j = \sum_{j+1}^n a_i \equiv A - A_j$.
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_1, \dots, r_j to R_j and r_{j+1}, \dots, r_n to \bar{R}_j .
3. Recur on the subregions R_j and \bar{R}_j .

end if

Figure 1: A Floorplanning-Benchmark Generator.

Algorithm 2.1. The notation for Algorithm 2.1 is as follows. Given N rectangles r_1, \dots, r_N with fixed areas $a_1 \geq a_2 \geq \dots \geq a_N$ but variable lengths ℓ_i and widths w_i , we seek to arrange them without overlap in a given rectangle \mathcal{R} of area $A = \sum_1^N a_i$ such that the aspect ratios

$$\rho_i = \rho(r_i) = \max(\ell_i/w_i, w_i/\ell_i)$$

are bounded close to 1.¹ The rectangle \mathcal{R} is the *floorplanning region*. The rectangles r_i are called *blocks*.

Algorithm 2.1 is parameterized by $\rho(\mathcal{R}) \geq 1$ and $\gamma \geq 1$. By construction, Algorithm 2.1 has the following property.

THEOREM 2.1. For $\rho(\mathcal{R}) \geq 1$ and $\gamma \geq 1$, Algorithm 2.1 generates a slicing floorplan with zero dead space. \square

¹By this definition (which we will use for the remainder of the paper) the aspect ratio of any block is always at least 1.

Although Algorithm 2.1 can accept as input any values $\rho(\mathcal{R}) \geq 1$ and $\gamma \geq 1$, the analysis in Section 3 shows that the generated floorplans display attractive properties for certain choices of $\rho(\mathcal{R})$ and γ . More specifically, if we define

$$\beta = \max_i a_i/a_{i+1}, \quad (1)$$

then appropriate values for γ and $\rho(\mathcal{R})$ are

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

A trace of the algorithm on a simple 5-block example is illustrated in Figure 2.

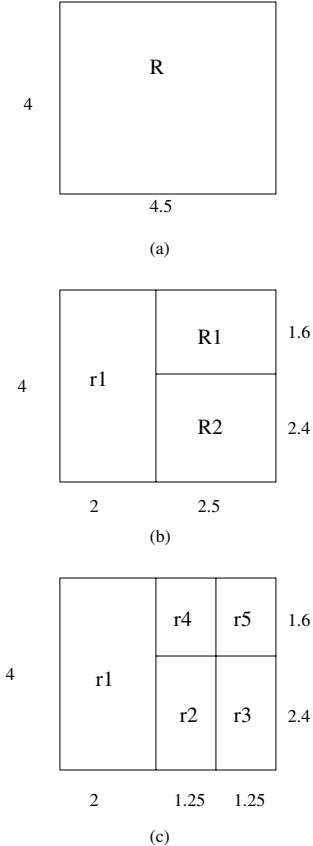


Figure 2: (a) Creation of a ZDS benchmark with 5 blocks, where $a_1 = 8, a_2 = a_3 = 3, a_4 = a_5 = 2$. In this case, $\gamma = 2.67$. Region R has area 18 and is initialized to an aspect ratio of 1.125. (b) Since $a_1 \geq 18/2.67$, block r_1 is placed. After partitioning of the remaining blocks, region 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. The maximum aspect ratio is $2 \leq \gamma + 1$.

3. ANALYSIS

The utility of Algorithm 2.1 rests on the fact that for nearly all realistic circuits, all the block aspect ratios it computes are guaranteed to lie within a single small interval of the form $[1, \gamma + 1]$, when γ is defined as in (2). Hence, if the blocks are arranged in nonincreasing sorted order by area,

the aspect ratios are bounded by one plus the maximum ratio of consecutive block areas, when this latter ratio exceeds 2. Otherwise, the aspect ratios are bounded above by 3. These facts are established here, under Assumptions 3.1 below.

ASSUMPTIONS 3.1. *Block-locking threshold; floorplanning-region aspect ratio bound.*

(a) For β as defined in (1), $\gamma \geq \max\{2, \beta\}$.

(b) The aspect ratio of the floorplanning region satisfies $\rho(\mathcal{R}) \leq \gamma + 1$.

Assumption 3.1(a) can be rephrased as follows: the threshold fraction of subregion area that a block must occupy in order to be shaped and locked in place is not set above $1/2$. Although these assumptions are stronger than necessary to achieve zero dead space and acceptably bounded block aspect ratios, they are not very restrictive on the sets of block areas that may be considered. Further discussion of the assumptions appears at the end of this section.

3.1 Derivation of the Aspect-Ratio Bound

Throughout this section, we consider the properties of Algorithm 2.1 under Assumptions 3.1. Due to page limits, only a sketch of the mathematical derivation of the aspect-ratio bound can be given here. Detailed proofs are available online in a technical report [20].

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*

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

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

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.

LEMMA 3.2. *Suppose subregion R is partitioned into subregions R_1 and R_2 with areas A_1 and A_2 . Let*

$$y = \max\{A_1/A_2, A_2/A_1\}.$$

Then

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

THEOREM 3.1. *In Algorithm 2.1,*

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

From Lemma 3.2 and Theorem 3.1, we immediately obtain the following bound.

COROLLARY 3.1. *Suppose subregion R is partitioned into subregions R_1 and R_2 . Then*

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

THEOREM 3.2. *Under Assumptions 3.1, the result of Algorithm 2.1 is a slicing floorplan with zero dead space and every block's aspect ratio bounded above by $\gamma + 1$.*

PROOF. Follows directly from Corollary 3.1 and Assumptions 3.1, by induction. \square

3.2 Remarks

The assumption $\gamma \geq 2$ 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 2 (e.g., $a_i/a_{i+1} \leq 1.5$) are not guaranteed to reduce the maximum aspect ratio below what can be attained with $a_i/a_{i+1} \leq 2$.

Similarly, a large value of γ does not necessarily indicate any large aspect ratios in the final floorplan, as Figure 5 illustrates. In the figure, one large block occupies one sub-region, 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.

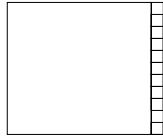


Figure 3: Block aspect ratios may all remain small, even when some subregion has a large aspect ratio.

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 $\max\{2, \max_i a_i/a_{i+1}\}$. However, a few simple preprocessing steps can usually be used to reduce this value significantly. The basic idea is simply to aggregate smaller, similarly sized blocks together until the aggregates are more comparable to larger blocks or sets of blocks. Suppose $a_m/a_{m+1} = \max_i a_i/a_{i+1}$. Define $R_m, \bar{R}_m, A_m, \bar{A}_m$ as in Section 2 above, but with $j \equiv m$ instead of j being chosen to minimize D_j . If $A_m/\bar{A}_m < a_m/a_{m+1}$, then since a_{m+1} and a_m are placed in separate subregions, Theorem 3.1 ensures that γ is reduced to

$$\max\{A_m/\bar{A}_m, \max_{i \neq m} a_i/a_{i+1}\}.$$

If $\max_{i \neq m} a_i/a_{i+1} \gg A_m/\bar{A}_m$, a few recursive iterations on R_m and \bar{R}_m can be used to reduce the bound further. Although it is trivial to construct examples where this preprocessing will be useless (e.g., when $N = 2$, or when $m = N - 1$), on practical examples with large N , the reduction in γ will likely be considerable, when m is sufficiently less than N .

4. AREA-OPTIMALITY STUDY

Algorithm 2.1 was implemented in C++/STL and applied to the MCNC benchmarks [13] and the largest of the GSRC [8] benchmarks without regard to connectivity and with aspect ratios unconstrained. The results were then interpreted as benchmarks that match the block-area characteristics of real circuits but have known optimal-area solutions. The new benchmarks differ from the original benchmarks in that they specify either (i) the exact aspect ratio of each block or (ii) a range of aspect ratios for each block that includes the ratio for the block computed by Algorithm 2.1. For most circuits, these aspect-ratio values and ranges are only slightly larger (or even smaller) than those in the original

benchmarks. For a scalability analysis we also used as prototype a version of the largest benchmarks (**ami49** and **n300**) with 10 times the number of blocks of the original circuit (**ami49-10x** and **n300-10x** respectively). We grouped the new benchmarks in a suite named FEKO-A (Floorplanning Examples with Known Optimal Area). The benchmarks and their optimal solutions can be found online [22].

Table 1 shows the characteristics of the circuits of the FEKO-A suite. The name of the original circuit, the number of blocks, the value of γ and the maximum aspect ratio of a block are displayed. In all the experiments, γ is set to the value from (2); hence, we interpret γ as both a circuit property and a ZDS-algorithm parameter. As expected, the aspect ratio of the blocks are bounded above by $\gamma+1$. With the exception of the **apte** circuit, the values of γ guarantee a relatively low maximum aspect ratio.

Circuit	Original	num of blocks	γ	max aspect ratio
FEKOA1	apte	9	24.3	10.01
FEKOA2	hp	11	2.80	2.01
FEKOA3	xerox	10	2.00	1.95
FEKOA4	ami33	33	2.00	2.84
FEKOA5	ami49	49	2.81	2.46
FEKOA6	n300	300	2.00	2.97
FEKOA7	ami49-10x	490	2.81	2.98
FEKOA8	n300-10x	3000	2.00	2.99

Table 1: Characteristics of the FEKO-A examples

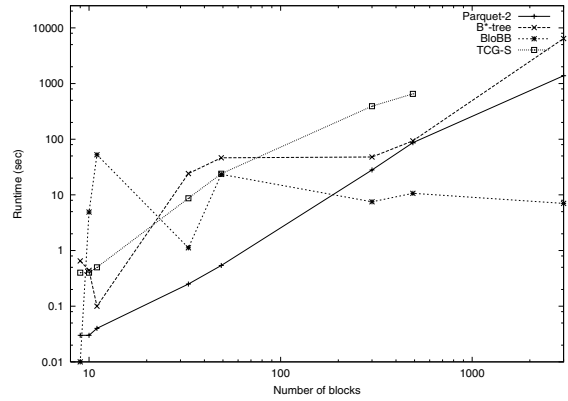


Figure 4: Runtime of one run of the floorplanning algorithms for different circuit sizes in log-log scale.

Note that on the **xerox** example, all the blocks have aspect ratio less than 2. This means that the optimal block placement of the benchmarks is a valid solution of the soft version problem for the original floorplanning benchmark, where the blocks can be reshaped to any aspect ratio up to 2. The same is true for the **n300** example, where the aspect ratio bound is 3 (as for all the GSRC benchmarks with soft blocks²). Even for the **hp** example, where the maximum aspect ratio of the blocks is 2.01, a simple postprocessing procedure of stretching the floorplan in one of the coordinate directions can reduce the maximum aspect ratio of the

²The value of γ for all the GSRC benchmarks is 2, so the Algorithm presented in this paper can solve them optimally.

blocks to 1.67. Thus, we have created optimal-area solutions for all these floorplanning benchmarks, which as far as we know have never before been reported in the literature.

We used the FEKO-A circuits to evaluate the performance of four state-of-the-art floorplanners whose code is available on the Internet.

- Parquet-2. Parquet [1] is a floorplanning package that uses the Sequence Pair [14] representation. The public release supports block shaping.³ Parquet-2 can be downloaded online [18].
- B*-tree. This package is based on the B*-tree geometric representation [6]. It can be downloaded online [3]. The public release does not support block shaping.
- TCG-S. TCG-S [12] is an extension of the well-known TCG [11] algorithm. The package can be downloaded online [3] and it does not support block shaping.
- BloBB (Block-packing with Branch-and-Bound) [4]. BloBB is a floorplanner for minimizing area only that solves small instances optimally (up to 12 blocks) and is using a hierarchical approach for larger instances. BloBB can be downloaded online [2] and it does not support block shaping.

We tested these four packages on the FEKO-A suite with all block aspect ratios held fixed. Every program was run 10 times and the best results were collected. All the packages were run in area minimization mode, since most of them support wirelength minimization as well, except for BloBB. Parquet-2, B*-tree and BloBB were run in a Linux Red-Hat 8.0 environment on a 2.2 GHz Pentium 4 processor (2 GB memory), while TCG-S was run on a SunBlade 1000 machine on Solaris 2.8 with a 750 MHz processor (2 GB memory). Table 2 shows the experimental results for each of the circuits in terms of dead space and runtime. The runtime statistics are also shown graphically in Figure 6. Since each benchmark has a known optimal solution with zero dead space, the dead space (measured as a percentage of the total area of a floorplan) generated by an algorithm shows the area gap between the algorithm’s result and the optimal solution.

Circuit	Parquet-2		B*-tree		TCG-S		BloBB	
	Dead space (%)	Run-time (sec)	Dead space (%)	Run-time (sec)	Dead space (%)	Run-time (sec) ⁴	Dead space (%)	Run-time (sec)
FEKO A1	14.36	0.03	4.76	0.65	5.40	0.4	0	0.01
FEKO A2	9.69	0.04	6.74	0.10	8.66	0.5	0	52.48
FEKO A3	11.31	0.03	3.95	0.44	3.61	0.4	0	4.91
FEKO A4	6.26	0.25	2.32	24.16	4.62	8.7	4.58	1.12
FEKO A5	5.55	0.54	2.05	46.29	4.80	24.0	6.56	23.34
FEKO A6	9.11	28.04	8.99	47.85	10.98	392.3	8.56	7.51
FEKO A7	8.70	86.65	11.99	93.59	15.30	651.3	8.81	10.67
FEKO A8	21.17	1383	27.35	6433	NA	NA	12.06	7.03

Table 2: Experimental results for the FEKO-A examples. The dead space is the best of 10 runs, the runtime is for one run.

From the table and the figure, we draw the following conclusions:

³In these experiments, for a fair comparison with the other programs, we fix the aspect ratios of all the blocks.

⁴The experiments with TCG-S were run on a 750 MHz processor, unlike the rest of the experiments.

- The BloBB package can find as expected the optimal solution for the three smallest benchmarks. For the other circuits, it finds non-optimal solutions of very good quality (12.06% from the optimal solution in the worst case).
- The performance of the other algorithms on the benchmarks with size up to 500 blocks is very good, 12% from the optimal in the worst case in the case of B*-tree, 15.3% in TCG-S, and 14.4% from the optimal in Parquet-2.
- For the largest benchmark (3000 blocks), Parquet’s floorplan has dead space equal to 21.17%, while the dead space of B*-tree was 27.35%. TCG-S did not complete one run after 24 hours.
- BloBB shows very good scalability in terms of run time. Actually its runtime for large circuits is sometimes better compared to the smaller circuits, since in the former case it is run in a non-optimal hierarchical mode, while in the latter case it finds the optimal solution.

5. OTHER APPLICATIONS

While the primary motivation for Algorithm 2.1 was to create benchmarks with known optimal-area solutions and reasonable aspect ratios, it seems likely that extensions of the ZDS idea can have more uses in physical design.

- As shown in Section 4, Algorithm 2.1 can pack optimally 2 out of the 5 MCNC benchmarks and all the GSRC benchmarks (soft-block version) within the given aspect ratio bounds. This suggests that a top-down ZDS algorithm can be used as a stand-alone soft-block packing algorithm for area minimization.
- Algorithm 2.1 does not consider wirelength, but there is some similarity between the recursive area-bipartitioning step of Algorithm 2.1 and the recursive process in partitioning-based placement algorithms. A multilevel floorplanner that can combine or alternate between these two steps has the potential of focusing on both objectives (area and wirelength), if the blocks’ aspect ratios can be relaxed up to three (or even less than three with a penalty of some dead space). This approach can be applied, for example, at the coarser levels of a multilevel placement algorithm, where cells are grouped into clusters with no predetermined shape.
- A ZDS algorithm can also be used to determine the shapes of blocks in a hierarchical design, where close interaction with designers at the highest levels of design can help achieve very compact block packings.

6. CONCLUSION

A novel algorithm was used to create floorplanning benchmarks with known optimal-area solutions. The construction of the benchmarks is such that the aspect ratios of the modules are uniformly bounded in terms of the area variation of the given blocks. This fact, in combination with the capability of the program to create benchmarks with any block-area distribution given by the user, makes the benchmarks realistic and suitable for evaluation of floorplanning

algorithms. Experiments were conducted with four state-of-the-art floorplanning packages run in block-packing mode with no reshaping. For circuits with fewer than 500 blocks, the algorithms perform very well, but for larger benchmarks they may leave dead space of up to 27%. These examples can potentially help algorithm designers to fine tune their programs and understand some of the limitations of existing floorplanning and block-packing algorithms.

Algorithm 2.1 can be viewed as one solution to an alternative formulation of floorplanning in which the maximum aspect ratio is minimized subject to a strict zero-dead-space constraint. From this view, the performance of Algorithm 2.1 can likely be improved considerably in at least two ways: (i) using one level of aggregation to reduce maximum area variations of sibling partitions, as discussed in Section 3; (ii) using direct search on the set of available aspect ratios of the entire floorplanning region in order to reduce the maximum aspect ratio of any block in the final floorplan. Additionally, we expect that more sophisticated postprocessing analysis of each block's sequence of ancestor subregions may be useful in converting a large aspect ratio of a single isolated block to a set of small perturbations to each of a larger set of blocks.

The FEKO-A benchmarks were created for area optimization only with no netlist information. A trivial extension for the generation of benchmarks with optimal wirelength can include the creation of zero-wirelength nets with degree up to 4 and pins placed at the intersections of block boundaries. However, this approach has limitations on matching the profile of real circuits (for example, the absence of high-degree nets etc.). Future work will explore the generation of realistic examples with known optimal or near-optimal solutions in terms of both area and wirelength.

7. REFERENCES

- [1] S. Adya and I. Markov. Fixed-outline floorplanning through better local search. In *Proc. International Conference on Computer Design*, pages 328–334, 2001.
- [2] <http://vlsicad.eecs.umich.edu/bk/blobb>.
- [3] <http://cc.ee.ntu.edu.tw/~ywchang/research.html>.
- [4] H. Chan and I. Markov. Symmetries in rectangular block-packing. In *Proc. of the International Workshop on Symmetry in Constraint Satisfaction Problems*, Sep. 2003.
- [5] C.C. Chang, J. Cong, and M. Xie. Optimality and scalability study of existing placement algorithms. In *Asia South Pacific Design Automation Conference*, pages 325–330, Kitakyushu, Japan, Jan 2003.
- [6] Y.C. Chang, Y.W. Chang, G. Wu, and S. Wu. B*-trees: A new representation for non-slicing floorplans. In *Proc. Design Automation Conference*, pages 458–463, 2000.
- [7] J. Cong, M. Romesis, and M. Xie. Optimality, stability and scalability study of partitioning and placement algorithms. In *Proc. of the International Symposium on Physical Design*, pages 89–94, 2003.
- [8] <http://www.cse.ucsc.edu/research/surf/GSRC/GSRCbench.html>.
- [9] P. Guo, C. Cheng, and T. Yoshimura. An O-tree representation of non-slicing floorplan and its applications. In *Proc. Design Automation Conference*, pages 328–334, 1999.
- [10] X. Hong, S. Dong, G. Huang, Y. Ma, Y. Cai, C. Cheng, and J. Gu. A non-slicing floorplanning algorithm using corner block list topological representation. In *Proc. Design Automation Conference*, pages 268–273, 1999.
- [11] J. Lin and Y. Chang. TCG: A transitive closure graph-based representation for non-slicing floorplans. In *Proc. Design Automation Conference*, pages 764–769, 2001.
- [12] J. Lin and Y. Chang. TCG-S: Orthogonal coupling of P*-admissible representations for general floorplans. In *Proceedings of the Design Automation Conference*, pages 842–847, 2002.
- [13] <http://www.cse.ucsc.edu/research/surf/GSRC/MCNCbench.html>.
- [14] H. Murata, K. Fujiyoshi, S. Nakatake, and Y. Kajitani. Rectangle-packing-based module placement. In *Proc. International Conference on Computer-Aided Design*, pages 472–479, 1995.
- [15] S. Nakatake, K. Fujiyoshi, H. Mirata, and Y. Kajitani. Module packing based on the BSG-structure and IC layout applications. In *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, volume 17, pages 519–530, 1998.
- [16] T. Okamoto and T. Yoshimura. A new approach to VLSI floorplanning based on quadratic programming and rectangle packing. In *Proc. of the 10th Workshop on Synthesis and System Integration of Mixed Technologies*, 2001.
- [17] R. Otten. Automatic floorplan design. In *Proc. Design Automation Conference*, pages 261–267, 1982.
- [18] <http://vlsicad.eecs.umich.edu/bk/parquet/>.
- [19] H. Peixoto, M. Jacome, A. Royo, and J. Lopez. A tight upper bound for slicing floorplans. In *Proc. of IEEE VLSI 2000*, Jan. 2000.
- [20] J. Shinnerl. A theorem on partitioning a sorted list of numbers with an application to VLSI floorplanning. Report 040006, Computer Science Dept., University of California, Los Angeles, <http://www.cs.ucla.edu/~shinnerl>, Feb. 2004.
- [21] D.F. Wong and C.L. Liu. A new algorithm for floorplan design. In *Proc. Design Automation Conference*, pages 101–107, 1986.
- [22] <http://cadlab.cs.ucla.edu/~pubbench/floor>.
- [23] F. Young and D. Wong. How good are slicing floorplans? In *Proc. of the International Symposium on Physical Design*, pages 144–149, 1997.