Fast Floorplanning by Look-Ahead Enabled Recursive Bipartitioning

Jason Cong, Michail Romesis, and Joseph R. Shinnerl
UCLA Computer Science Department
{cong,michail,shinnerl}@cs.ucla.edu *

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 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 PATOMA, generates solutions with half the wirelength of state-of-the-art floorplanners in orders of magnitude less run time.

I. INTRODUCTION

Fast floorplanning is critical in the hierarchical physical design of 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, multiscale and mixed-size 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 IP blocks for multi-million-gate ASICs and 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.

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 non-slicing algorithms. The first slicing algorithms were developed in the 1980's (e.g. [13], [16]). In the 1990's, non-slicing algorithms became more popular, especially after the introduction of the BSG [12] and Sequence Pair [11] representations. Other non-slicing representations include TCG [10], B*-tree [3], CBL [8], O-tree [6], and so on. Simulated annealing (SA) has been used to minimize area and/or wirelength under each of these representations.

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 non-stochastic approaches. Ranjan et al. [14] proposed 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 wirelength by reshaping blocks to produce a more compact layout. Final, total wirelength was comparable to or better than that obtained by an SA-based algorithm [16], with speed-up of over 1000× in predictor mode (high-speed) and 20× in constructor mode (high-effort). More recently, a fast algorithm called Traffic [15] has been used to generate high-quality floorplans without simulated annealing. Traffic also uses two stages. In the first stage, the blocks are divided into layers by linear multi-way 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 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. 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. The work presented here is the first, as far as we know, to define a floorplanning flow driven by recursive cutsize-driven bisection in which the satisfiability of all constraints is explicitly enforced at every step, so that the need for post-hoc legalization is completely removed. *Legal*, a.k.a. *feasible* solutions, strictly satisfying all non-overlapping, area, and shape constraints, are explicitly constructed for every subproblem at each intermediate level. Their feedback to the recursive bisection enables it to proceed significantly longer, deepening the partitioning hierarchy and thereby improving wirelength quality. Our implementation of this flow beats a leading SA-based engine on the GSRC benchmarks by 10–20% in average wirelength and by orders of magnitude in run time.

The paper is organized as follows. Section II gives an

*Financial supports from Semiconductor Research Consortium Contracts 2001-TJ-910 and 2003-TJ-1019 and National Science Foundation Grant CCR-0096383 are gratefully acknowledged.

overview of our implementation, called PATOMA.[1] Section III describes a zero-dead-space floorplanning algorithm (ZDS) and its adaptation to wirelength minimization in PATOMA. Section IV describes the ROB (Row-Oriented Block Packing) heuristic for floorplanning a combination of hard and soft blocks. Section V compares PATOMA's performance with that of Parquet-2 [1]. The paper is concluded in Section VI.

II. Overview of The PATOMA Algorithm

PATOMA attempts to minimize 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 Figure 1. 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. Before each application of cutsize-driven bipartitioning, however, one of two separate fast, area-driven floorplanners 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.[2] 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 then resumes recursively on these subproblems, until single-block base cases are reached.

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 [4] of sufficient conditions for zero-dead-space 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 row-oriented block packing (ROB) is used. ROB is somewhat similar to Traffic [15]; however, it handles both soft and hard blocks under a fixed-outline constraint. Both algorithms perform well in reasonable run time. They are reviewed in Sections III and IV below.

PATOMA uses the well-known multilevel partitioning package hMetis [9]. Neither of the two block subsets produced is allowed to hold more than 60% of the total area of all blocks in both subsets. This choice of area balance produced the best results in our experiments. Terminal propagation is used to account for connections between partitions.

[1] An acronym for "Partitioning To Optimize Module Arrangement." Pronounced *PAH-toh-ma*, from the Greek for "floor."

[2] Failure of ZDS (Section III) or ROB (Section IV) to find a legal *initial* solution, prior to recursive bipartitioning, is highly unlikely. We have not observed any such failure on any circuit.

Algorithm II.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. Generate the root node $(\mathcal{S}, R)$ at level $i = 1$, and a legal floorplan for the root.

while *there are still blocks to be placed*
  while *there are unvisited nodes at level i*
    Select unvisited node $n = (\mathcal{S}_n, R_n)$ of level i.
    Use terminal propagation to model connections between $b_i \in \mathcal{S}_n$ and $b_j \notin \mathcal{S}_n$.
    Call hMetis to partition $\mathcal{S}_n$ into disjoint subsets $\mathcal{S}_{n1}$ and $\mathcal{S}_{n2}$, 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 $\mathcal{S}_{ni}$ are soft)*
          fit$[i]$ *:= ZDS*$(\mathcal{S}_{ni}, R_{ni})$.
        else *fit*$[i]$ *:= ROB*$(\mathcal{S}_{ni}, R_{ni})$.
        end if
      end for
      if *(fit*$[j]$ **and not** *fit*$[k]$, $j, k \in \{1, 2\}$*)*
        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 $(\mathcal{S}_n, R_n)$ 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. The PATOMA floorplanning algorithm.

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 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.

III. Wirelength-Aware ZDS Floorplanning

Zero-dead-space (ZDS) floorplanning is used in PATOMA only when all blocks are soft. The ZDS algorithm ignores wirelength. Under conditions reviewed below, its result is a ZDS floorplan with the aspect ratios of all blocks bounded between $1/3$ and 3. Both the original ZDS algorithm [4] and PATOMA's extensions to it are reviewed here.

Let the blocks be sorted by nonincreasing areas, $a_1 \geq \ldots \geq a_N$, and let β be the maximum ratio of the areas of any two consecutive blocks; $\beta = \max_i\{a_i/a_{i+1}\}$. Let $\gamma = \max\{2, \beta\}$. An analysis shows that, if all block aspect ratios ρ_i are allowed to range freely in $[1/(\gamma+1),\ \gamma+1]$, then a zero-dead-space floorplan for this set of blocks can be found for any given region with area equal to the sum of the areas of the blocks and any fixed aspect ratio in $[1/(\gamma+1),\ \gamma+1]$.

The ZDS algorithm proceeds as follows. At each step, the blocks are sorted according to their area, and the largest block is examined. If it fills up at least $1/\gamma$ of the area of its enclosing subregion, it is shaped and placed flush against one side of that subregion. A cut is made for the remaining unplaced sorted blocks such that the resulting subsets' total areas are as nearly equal as possible. The subregion is then cut parallel to its shorter side so that the areas of the resulting subregions equal those of the two partitioned block sets. Cutting parallel to the shorter side keeps aspect ratios of subregions bounded in terms of the area variation among the blocks.

The ZDS algorithm is very fast, both asymptotically ($\mathcal{O}(n \log n)$) and in practice (it floorplans 300 blocks in a few seconds). All the GSRC soft-block-packing benchmarks can be solved optimally by this algorithm; i.e., all blocks can be shaped and placed with zero dead space and with all blocks' aspect ratio constraints $1/3 \leq \rho_i \leq 3$ satisfied. Thus, its required conditions are not very restrictive.

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 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.

IV. Row-based Floorplanning

The ROB (Row-Oriented Block Packing) heuristic is used by PATOMA for floorplanning a combination of fixed- and variable-dimension blocks. It is similar to Traffic [15] in that it organizes the blocks by rows according to their dimensions; however, it satisfies a a fixed-outline constraint and handles both hard and soft blocks. Assume 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 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. During the first stage, blocks are considered one by one and either added to 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 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, 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, some rows are eliminated by redistributing blocks one by one. The rows are scanned in a decreasing height order. Blocks from 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.

V. Experiments and Results

We compare PATOMA to to Parquet-2 [1], a state-of-the-art SA-based floorplanner using the Sequence Pair geometric representation, Traffic [15] and FFPC, the fast floorplanner of Ranjan et al. [14]. For a fair comparison, all experiments were performed on the same machine, a 2.4GHz Pentium IV running RedHat Linux 8.0. Due to the page limit, result tables are omitted; they can be found in a technical report [5]. We compared on four sets of benchmarks. For all the experiments, the floorplanners are trying to minimize the wirelength in a fixed outline. The first set of benchmarks includes the 4 largest GSRC ciruits (size 200 - 300 blocks), where all the blocks are soft. For this set we compare only to Parquet-2, 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-2 in two modes. The first mode is the default and is very fast, due to a shorter simulated-annealing schedule that hurts the wirelength quality. The second mode is a high-effort mode, where we impose a time limit of one hour to allow SA to attain a better solution. In the all-soft-block examples, PATOMA uses only the ZDS algorithm and not ROB to enforce the legalizability of all floorplanning subproblems. All blocks are allowed to be reshaped with any aspect ratios in $[1/3, 3]$. The default mode of Parquet-2 produces results that are 19% higher in wirelength than PATOMA, while its runtime is $37\times$ slower. The high-effort mode of Parquet-2 is 11% worse in wirelength and $824\times$ slower than PATOMA.

The second set of experiments includes the same GSRC benchmarks, but with all blocks of given, fixed dimensions. In these examples, PATOMA uses only ROB and not ZDS to enforce the legalizabilty of floorplanning subproblems, because all blocks are hard. On these benchmarks, PATOMA produces results of 10% lower wirelength than the default mode of Parquet-2, with a speedup of $33\times$, and of 5% lower wirelength than the high-effort mode of Parquet-2, with an average speedup of $523\times$.

The third set of experiments includes the same GSRC circuits all blocks hard, but without pads. PATOMA was compared with Traffic and FFPC for these benchmarks, since these floorplanners do not use pads or shape soft blocks. FFPC's wirelength is 3% longer than PATOMA's, on average, while its run time is $6\times$ longer. With Traffic's run-time limit set to PATOMA's run time, Traffic's average total wirelength is 60% longer than PATOMA's.

In the fourth set of experiments, we generated large-scale floorplanning benchmarks from the IBM/ISPD98 suite [2] 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 [9]. 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 2,000 blocks. We called this suite of benchmarks the HB-suite (hybrid blocks). These benchmarks are available online [7]. For these examples, Parquet-2's wirelength is on average 104% higher than PATOMA's, while it is $209\times$ slower.

VI. Conclusions

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 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.

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] C.J. Alpert. The ISPD98 Circuit Benchmark Suite. In *Proc. Int'l Symp. on Phys. Design*, pages 80–85, 1998.

[3] 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.

[4] J. Cong, G. Nataneli, M. Romesis, and J. Shinnerl. An Area-Optimality Study of Floorplanning. In *Proc. Int'l Symposium on Physical Design*, pages 78–83, 2004.

[5] J. Cong, M. Romesis, and J. Shinnerl. Fast fboorplanning by look-ahead enabled recursive bipartitioning. Technical Report TR040043, Computer Science Dept., University of California, Los Angeles, 2004.

[6] P. Guo, C. Cheng, and T. Yoshimura. An O-tree Representation of Non-slicing Floorplan and its Applications. In *Proc. Design Automation Conf.*, pages 328–334, 1999.

[7] http://cadlab.cs.ucla.edu/cpmo/HBsuite.html/.

[8] 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 Conf.*, pages 268–273, 1999.

[9] G. Karypis, R. Aggarwal, V. Kumar, and S. Shekhar. Multilevel hypergraph partitioning: Application in vlsi domain. In *Proc. 34th ACM/IEEE Design Automation Conference*, pages 526–529, 1997.

[10] J. Lin and Y. Chang. TCG: A Transitive Closure Graph-Based Representation for Non-Slicing Floorplans. In *Proc. Design Automation Conf.*, pages 764–769, 2001.

[11] 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.

[12] 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.

[13] R. Otten. Automatic Floorplan Design. In *Proc. Design Automation Conf.*, pages 261–267, 1982.

[14] A. Ranjan, K. Bazargan, S. Ogrenci, and M. Sarrafzadeh. Fast Floorplanning for Effective Prediction and Construction. In *IEEE Trans. on VLSI Sys.*, pages 341 – 351, 2001.

[15] P. Sassone and S.K. Lim. A Novel Geometric Algorithm For Fast Wire-Optimized Floorplanning. In *Proc. International Conference on Computer-Aided Design*, 2003.

[16] D.F. Wong and C.L. Liu. A New Algorithm for Floorplan Design. In *Proc. Design Automation Conference*, pages 101 – 107, 1986.