

# Recursive Bisection Based Mixed Block Placement

Ateen Khatkhate<sup>1</sup> Chen Li<sup>2</sup> Ameya R. Agnihotri<sup>1</sup> Mehmet C. Yildiz<sup>3</sup> Satoshi Ono<sup>1</sup>  
Cheng-Kok Koh<sup>2</sup> Patrick H. Madden<sup>1</sup>  
SUNY Binghamton CSD<sup>1</sup> Purdue University ECE<sup>2</sup> IBM Austin Research Lab<sup>3</sup>

## ABSTRACT

Many current designs contain a large number of standard cells intermixed with larger macro blocks. The range of size in these “mixed block” designs complicates the placement process considerably; traditional methods produce results that are far from satisfactory.

In this paper we extend the traditional recursive bisection standard cell placement tool *Feng Shui* to directly consider mixed block designs. On a set of recent benchmarks, the new version obtains placements with wire lengths substantially lower than other current tools. Compared to *Feng Shui 2.4*, the placements of a *Capo*-based approach have 29% higher wire lengths, while the placements of *mPG* are 26% higher. Run times of our tool are also lower, and the general approach is scalable.

## Categories and Subject Descriptors

J.6 [Computer-Aided Engineering]: CAD

## General Terms

Algorithms

## Keywords

Placement, floorplanning, mixed block design

## 1. INTRODUCTION

There has been explosive growth in the size of integrated circuits; following the exponential curve of Moore’s law, modern designs can have over 50 million transistors. This growth is projected to continue, and circuit designers are having difficulty using this capacity effectively. Hierarchy is commonly used to tame unwieldy designs.

At the highest hierarchical level, floorplanning of a few hundred large blocks can be done effectively. At the lowest level, placement tools can handle hundreds of thousands or millions of standard cells. Between these two extremes is *mixed block placement*,

in which blocks of moderate size are intermixed with many standard cells. Current placement tools have had difficulty in handling this “boulders and dust” problem.

In this paper, we adapt an earlier version of our recursive bisection based standard cell placement tool *Feng Shui* to handle mixed block designs. While many recent academic approaches have sought to address standard cell and macro block placement in separate steps, we consider them simultaneously. We find that this can be done more easily than might be expected, and produces results that are vastly superior to recently published work. As our approach is based on recursive bisection, the tool itself is quite fast and scales well to large designs.

Using benchmarks derived from industry partitioning examples, our new placement tool obtains excellent wire length results. A multi-stage flow using the standard cell placement tool *Capo*, and an integrated approach called *mPG*, obtain results that are respectively 29% and 26% higher *on average*. On some benchmarks, wire lengths obtained by our tool are roughly half of those of *Capo* and *mPG*. The gap in solution quality may be surprising; in most areas of design automation, we might expect only a few percent improvement from a new technique. We have verified our results with publicly available tools, and our placement results and the placement tool itself are available through the web.

We suspect that the magnitude of improvement is largely due to the limited amount of recent published research in the area: most academic groups have focused on either standard cell placement or block placement, with mixed block problems getting comparatively little attention. Inspection of our placements reveals many opportunities for further gains, and we anticipate that there will be additional significant improvement in mixed block placement.

The remainder of this paper is organized as follows. In Section 2, we describe various types of placement problems, and prior work related to mixed block designs. Section 3 describes our approach to mixed block placement. Experimental results are given in Section 4, and we conclude the paper with Section 5.

## 2. PREVIOUS WORK

Circuit placement is a well studied problem, and comes in a variety of forms. Most common are standard cell placement, block placement, and our focus here, mixed block placement.

### 2.1 Standard Cell Placement

In standard cell placement, we may have a great many *cells*, small rectangular blocks that are of uniform height, but possibly varying width. Each cell contains the circuitry for a relatively simple logical function, and the cells are packed into rows much as one might use bricks to form a wall. The desired circuit functionality is obtained by connecting each cell with metal wiring. The arrange-

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’04 April 18–21, 2004, Phoenix, Arizona, USA.

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

ment of cells is critical to obtaining a high performance circuit. Due to the dominance of interconnect on system delay[9], slight changes to the locations of individual cells can have sweeping impact. Beyond simple performance objectives, a poor placement may be infeasible due to routing issues: there is a finite amount of routing space available, and a placement that requires large amounts of wiring (or wiring concentrated into a small region) may fail to route. Well known methods for standard cell placement include simulated annealing[23, 26], analytic methods[15, 11], and recursive bisection[7, 4].

## 2.2 Block Placement and Floorplanning

Block placement, block packing, and floorplanning[19] problems involve a small number of large rectilinear shapes. There are usually less than a few hundred shapes which are almost always rectangular. Each block might contain large numbers of standard cells, smaller blocks, or a mix of both; the internal details are normally hidden, with the placement tool operating on an abstracted problem.

For blocks, we must arrange them such that there is no overlap; the optimization objective is generally a combination of the minimization of the amount of wasted space, and also a minimization of the total routing wire length. Small perturbations to a placement (for example, switching the orientation of a block, or swapping the locations of a pair of blocks) can introduce overlaps or change wire length significantly; simulated annealing is commonly used to explore different placements, as it is effective in escaping local minima. There are a number of different floorplan and block placement representation methods[18, 17, 1, 20], each having different merits with respect to the computational expense of evaluating a placement.

## 2.3 Mixed Block Design

Between the extremes of standard cell placement and floorplanning is mixed block design. Macro blocks of moderate size are intermixed with large numbers of standard cells. The macro blocks occupy an integral number of cell rows, and complicate the placement process in a number of ways.

If a block is moved, it may overlap a large number of standard cells—these must be moved to new locations if the placement is to be legal. The change in wire length for such a move can make the optimization cost function chaotic. There is also considerable computational expense in simply considering a particular move.

Early researchers [24, 25, 22, 21] used a hierarchical approach, where the standard cells were first partitioned into blocks using either the logical hierarchy or min-cut-based partitioning algorithms. Floorplanning was then performed on the mix of macro blocks and partitioned blocks, with the goal of minimizing wirelength. Finally, the cells in each block were placed separately using detailed placement. While this method reduces problem size to the extent where the floorplanning techniques can be applied, pre-partitioning standard cells to form rectangular blocks may prevent such a hierarchical method from finding an optimal or near-optimal solution.

The *Macro Block Placement* program [25] restricts the partitioned blocks to a rectangular shape. However, using rectilinear blocks are more likely to satisfy today’s high performance circuit needs. The *ARCHITECT* floorplanner [22] overcomes this limitation and permits rectilinear blocks. However, while both these techniques have been demonstrated on small problem sizes, it is not clear how effectively they will handle today’s much larger circuits both in terms of the number of standard cells and the number of macro blocks.

In [24, 21], simulated annealing based techniques were used to

place mixed block designs. In [24], the placer was tested on an industrial circuit from Texas Instruments, Inc. In [21], nine industrial circuits were used for experimentation, but the placement performance metrics were compared to manual layouts. While these methods gave good results for small to medium-size designs, it is unclear how well they scale to modern problems.

Mixed-Mode Placement (*MMP*)[27] uses a quadratic placement algorithm combined with a bottom-up two-level clustering strategy and slicing partitions to remove overlaps. *MMP* was demonstrated on industrial circuits with thousands of standard cells and not more than 10 macro blocks; these designs are also relatively small compared to the designs we consider here.

More recently, a three stage placement-floorplanning-placement flow [2, 3] was presented to place designs with large numbers macro blocks and standard cells. The flow utilizes the *Capo* standard cell placement tool, and the *Parquet* floorplanner. In the first stage, all macro blocks are “shredded” into a number of smaller sub-cells connected by two-pin nets created to ensure that sub-cells are placed close to each other during the initial placement. A global placer is then used to obtain an initial placement. In the second stage, initial locations of macros are produced by averaging the locations of cells created during the shredding process. The standard cells are merged into soft blocks, and a fixed-outline floorplanner generates valid locations for the macro blocks and soft blocks of movable cells. In the final stage, the macro blocks are fixed into place, and cells in the soft blocks go through a detailed placement.

This flow is similar to the hierarchical design flow as both use floorplanning techniques to generate an overlap-free floorplan followed by standard cell placement. Rather than using pure partitioning algorithms to generate blocks for standard cells, this flow proposes to use an initial placement result to facilitate good soft block generation for standard cells. While this approach scales reasonably well, our experimental results show that it is not competitive in terms of wire length.

A different approach is pursued in [8]. The simulated annealing based multi-level optimization tool, *mPG*, consists of a coarsening phase and a refinement phase. In the coarsening phase, both macro blocks and standard cells are recursively clustered together to build a hierarchy. In the refinement phase, large objects are gradually fixed in place, and any overlaps between them are gradually removed. The locations of smaller objects are determined during further refinement. Considerable effort is needed for legalization and overlap removal; while the results of *mPG* are superior to those of [3], they are also not competitive with the tool we present here.

## 3. NEW PLACEMENT APPROACH

---

**Algorithm 1** Mixed Block Placement approach.

---

*Partition the circuit using “Fractional Cut” based recursive bisection.*

*Remove overlaps using Greedy Legalization technique.*

*Perform branch and bound reordering on standard cells.*

---

Our mixed block placement approach is based on recursive bisection; a high level outline of our approach is given in Algorithm 1. The basic recursive bisection method is well known; a partitioning algorithm splits a circuit netlist into two components, with the elements of each component being assigned to portions of a placement region. The partitioning progresses until each logic element is assigned to its own small portion of the placement region. Placement tools which follow this approach include [6, 10, 7, 4]. In our tool, we use the multi-level partitioner *hMetis*[14].

### 3.1 Fractional Cut Bisection

Traditionally, the placement region is split horizontally and vertically, with all horizontal “cuts” being aligned with cell row boundaries. In [4], the earlier version of our placement tool introduced a *fractional cut* approach; this was used to allow horizontal cut lines that were not aligned with row boundaries. Instead of row-aligned horizontal cuts, the partitioning solution and region areas were determined without regard to cell row boundaries.

After completion of the partitioning process, cells were placed into legal (row aligned and non-overlapping) positions by a dynamic programming based approach.

We make an observation about the fractional cut approach that should give intuition into our method of handling mixed block designs. When bisecting a region, the area of each region must match the area of the logic elements assigned to it, but there is no constraint that the shape of the region be compatible with the logic. For example, it is possible to have a region that is less than one cell row tall. While the logic elements can overlap slightly during bisection, the area constraints ensure that there is enough “space” in the nearby area such that the design can be legalized without a large amount of displacement.

The success of the fractional cut approach motivated our consideration of its use in mixed block design. As with the standard cells, a macro block may require space outside of the region to which it is assigned; we rely on a legalization step to adjust locations slightly to find an acceptable placement.

### 3.2 Mixed Block Enhancement

Our adaptation of the standard fractional cut based bisection process is to simply ignore the fact that some elements of the net list are more than one row tall. Rather than adding software to handle macro blocks, we instead modified our source code to not distinguish between macro blocks and standard cells.

The partitioning process proceeds in the same manner as most bisection based placement tools. The net list is partitioned until each region contains only a single circuit element. The area for each region matches the area of the circuit element that it contains—if the element happens to be a macro block, the area is simply larger than another region that might hold a standard cell.

The output of the bisection process is a set of “desired” locations for each block and cell; as with analytic placement methods, these locations are not legal, and there is some overlap. Fortunately, the amount of overlap is relatively small (due to area constraints and the use of fractional cut lines), allowing legalization to be performed with relative ease.

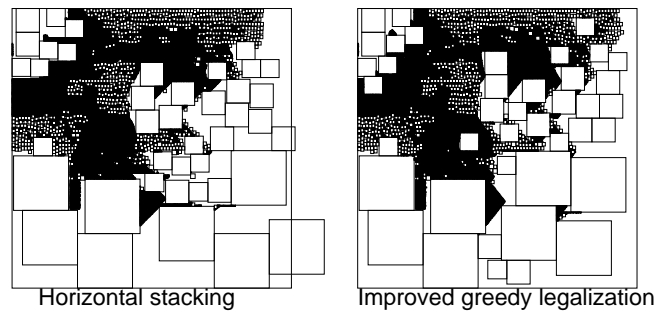
### 3.3 Placement Legalization

Our *Feng Shui 2.0* placement tool[4] used a dynamic programming based legalization method. The legalization process operated on a row-by-row basis, selecting cells to assign to a row. As macro blocks span multiple rows, this method could not be used directly.

#### 3.3.1 Initial Attempt

Our first attempt at a legalization method used a recursive greedy algorithm. We first attempted to find good locations for the macro blocks in the core region, fixing them in place, and then placing the standard cells in the remaining available space.

A block was considered to be legal if it was inside the core region and did not overlap with any of the previously placed blocks. Blocks were processed one at a time; if the block position was acceptable, the location was finalized. If the block position was not acceptable, a recursive search procedure ensued to find a nearby location where the block could be fixed into place.



**Figure 1: IBM10 placements; the simple greedy legalization method can result in a horizontal stacking of macroblocks that exceeds the core area. By reducing the penalty for movement in the  $y$  dimension, the stacking problem can be eliminated.**

After fixing the block locations, the standard cell rows were “fractured” to obtain space in which the standard cells could be placed. A modified version of the dynamic programming method presented in [4] was used to assign standard cell locations.

For designs with relatively few blocks, or blocks that were uniformly distributed in the placement region, our initial approach worked well. However, for the designs with many macro blocks, the large numbers of overlaps caused this approach to fail.

#### 3.3.2 Greedy Legalizer

While our initial attempts at placement legalization used a variety of complex methods, we have obtained the best results using a technique that is remarkably simple. The method we describe here is based on an algorithm presented in a technical report by Li[16]; we later learned that the algorithm was comparable to an earlier method patented by Hill[13]. The method by Hill can handle only objects with uniform height; we have improved on this method to allow legalization of designs with both standard cells and macro blocks.

For standard cell design, the method by Hill uses a simple greedy approach. All cells are sorted by their  $x$  coordinate; each cell is then packed one at a time into the row which minimizes total displacement for that cell. To avoid cell overlaps, the “right edge” for each row is updated, and the packing is done such that the cell being inserted does not cross the right edge. The patent describes packing from the left, right, top, or bottom, for objects that are either of uniform height or uniform width.

Our extension of this method removes the need for uniform height or width. As with the patented method, all circuit elements are sorted by their desired  $x$  coordinate, and assignment is performed in a greedy manner. Macro blocks are considered simultaneously with standard cells; the “right edge” checking is enhanced to consider multiple rows when packing a macro block. This method is outlined in Algorithm 2.

The macro blocks are treated very much like standard cells; they must be placed at the end of a growing row, and not overlap with any placed cell. The introduction of multi-row objects can result in “white space” within the placement region. Assuming that there are a number of nearby standard cells, the “liquidity” of the placement allows the cells to flow into the gaps, resulting in a tight packing while considering both blocks and cells simultaneously.

#### 3.3.3 Horizontal Stacking

For some designs, the greedy legalization method initially failed to place all blocks within the core area. This occurred most fre-

**Algorithm 2** Greedy legalization; circuit elements are processed one at a time, with each being assigned to the row that gives a minimum displacement.

*Sort all cells/macros by their “left edge” locations.*

```

for each row  $r$  do
   $right\_edge[r] = lx[r]$ ;
end for
for each cell/macro  $c$  do
   $llx[c] = legal\_x[lx[c]]$ ;
  for each row  $r$  do
    if macro crosses upper/lower boundary then
      continue;
    end if
    if  $llx[c] < right\_edge[r]$  then
       $dx = Diff(lx[c], right\_edge[r])$ ;
    else
       $dx = Diff(lx[c], llx[c])$ ;
    end if
     $dy = Diff(ly[c], ly[r])$ ;
     $cost = COST(dx, dy)$ ;
    Store  $best\_cost, best\_llx, best\_row$ .
  end for
   $llx[c] = best\_llx$ ;
   $row[c] = best\_row$ ;
end for

```

quently on the benchmark IBM10; the problem is shown in Figure 1. Our fractional cut representation created regions that did not match the shape of the actual blocks, resulting in them “stacking” during the greedy legalization step.

This problem motivated the slight enhancement shown in Algorithm 3. If the legalization results in circuit elements being placed outside of the core region, we gradually reduce the penalty for displacing a cell or macro block in the vertical direction. During legalization, rather than shifting blocks horizontally (creating a “stack”), the reduced vertical displacement penalty results in blocks and cells moving up or down to find positions in rows that are closer to the left side of the placement region. While this generally increases the total wire length, it allows all benchmarks to be legalized within the allowed core area.

### 3.4 Detailed Placement

Following legalization, we apply a window-based branch-and-bound detailed placement step. Macro blocks are not moved during this step—we consider only a small number of standard cells at a time, and enumerate all orderings to find an order which minimizes wire length.

By default, we consider groups of six cells at a time, with the window spanning one or more rows at a time. The detailed placement engine in *Feng Shui 2.4* can be used to post-process the placement results of other tools, and a number of research groups are currently using our tool in this manner.

### 3.5 Summary

To summarize our approach—we use a number of fairly simple techniques to obtain good results. The basic placement framework is traditional recursive bisection, with a fractional cut representation. We essentially ignore the difference between standard cells and macro blocks during bisection, and consider only the total area. Following bisection, we apply a very simple greedy legalization technique. Detailed placement is performed with traditional window-based branch-and-bound.

**Algorithm 3** Improved greedy legalization. For some circuits, macro block overlaps resulted in a horizontal arrangement that exceeded the core width. By reducing  $ycost$ , the penalty of shifting blocks vertically, we obtain placements that fit within the core area.

```

 $done = 0$ ;
Initialize “ $ycost$ ”.
while  $!done$  do
   $done = 1$ ;
  for each cell/macro do
     $greedy\_legalize()$ ;
    if cell crosses right boundary then
      Decrease  $ycost$ .
       $done = 0$ ;
      break;
    else
      Store best solution.
    end if
  end for
end while

```

## 4. EXPERIMENTAL RESULTS

To validate our approach, we performed placements on the mixed block benchmarks available on the GSRC bookshelf site[12]. The benchmarks are derived from partitioning examples released by IBM[5]. The partitioning benchmarks were in fact derived from mixed block circuit designs—information related to block shapes, net signal directions, etc., was removed. The partitioning benchmarks have been used to create standard cell placement benchmarks, and have also been used to generate synthetic benchmarks.

The mixed block designs are relatively large, and contain many macro blocks and standard cells (except for IBM05, which has no macro blocks). All macro blocks are assumed to be hard blocks with fixed aspect ratios. The circuit characteristics are listed in Table 1; we list the number of nets, cells, macro blocks, and pads, as well as the total area of all cells and all macro blocks. Each design contains roughly 20% “white space”—the core area for placement is larger than the total area of cells and macro blocks. We assume a fixed die placement paradigm, with our legalizer ensuring that all circuit elements are placed within the core region.

Bench Mark	# Nets	# Cells	# Macro	# Pads	%Cell Area	%Macro Area
ibm01	14111	12260	246	246	37.23	42.76
ibm02	19584	19071	271	259	24.69	55.31
ibm03	27401	22563	290	283	30.04	49.96
ibm04	31970	26925	295	287	38.03	41.98
ibm05	28446	28146	0	1201	80.01	0.00
ibm06	34826	32154	178	166	34.60	45.41
ibm07	48117	45348	291	287	44.07	35.93
ibm08	50513	50722	301	286	38.79	41.20
ibm09	60902	52857	253	285	40.18	39.82
ibm10	75196	67899	786	744	20.34	59.66
ibm11	81454	69779	373	406	42.36	37.63
ibm12	77240	69788	651	637	28.35	51.65
ibm13	99666	83285	424	490	43.82	36.18
ibm14	152772	146474	614	517	60.36	19.64
ibm15	186608	160794	393	383	53.26	26.74
ibm16	190048	182522	458	504	42.11	37.89
ibm17	189581	183992	760	743	62.80	17.20
ibm18	201920	210056	285	272	71.31	8.69

**Table 1: Statistics for the 18 IBM Benchmarks. In each design, there is roughly 20% white space available.**

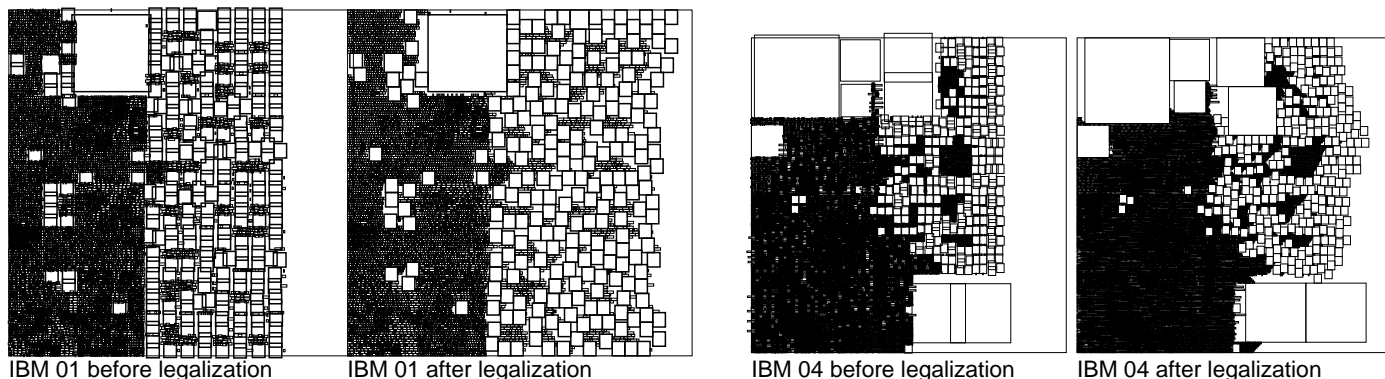


Figure 2: IBM 01 and 04, before and after legalization.

For comparison, we use recently published results on these benchmarks. In [2], a “shredding” approach was used with the *Capo* standard cell placement tool; macro blocks were handled using a multiple stage approach. This method subsequently improved in [3]. The multi-level approach of *mPG* was described in [8], and is also used for comparison.

Table 2 shows half perimeter wire length (HPWL) results for the the *Capo*-based and *mPG* placement tools on the IBM mixed block benchmarks. We also show the results of our own tool, and give the percentage difference between our tool and *mPG* and the best result from any *Capo*-based approach. The gap in half perimeter wire length may be quite unexpected; we have verified our results using public tools, and have also exchanged placement results with our colleagues to check for errors.

Run times are given for each tool, but are not directly comparable. *Capo I* results are from runs on a 1 GHz Pentium 3 based workstation, *Capo II* and *III* results are with a 2 GHz Pentium 4 workstation, *mPG* results are from a 750MHz Sun Blade 1000 workstation, and our results were obtained on a 2.5GHz Pentium 4 workstation. With equalized hardware resources, we expect that our placement method would still obtain the lowest run times. None of the tools have exceptionally high run times, and all should scale reasonably well.

In Figure 2, we show placements for IBM 01 and 04, before and after legalization. We refrain from showing larger benchmarks, as this increases the size of the electronic version of this paper. With the electronic version, it should be possible to zoom in to see a detailed view of the placements.

## 5. SUMMARY AND FUTURE WORK

In this paper, we have shown that a traditional recursive bisection placement approach can be adapted to handle mixed block designs. Key elements of our tool are the use of a *fractional cut* representation, and a remarkably simple legalization method. While it is certainly possible to have designs that cannot be legalized by our method, we find that in the currently available benchmark circuits, this is not a problem.

It should be obvious that mixed block placement is far from a mature research area. The improvement we have obtained is not incremental, and we anticipate that there is still a great deal of optimization potential. We have focused solely on half perimeter wire length in this paper; still to be addressed are issues such as timing, power consumption and thermal issues, and the host of other problems that plague modern integrated circuit design.

Routability of placements is a significant concern. Our legalization method does not perform the deliberate insertion of “white

space,” so our designs may be somewhat more dense than those of *Capo* or *mPG*. In our research on standard cell placement, however, we find that a lack of white space does not necessarily imply routing failures—one of our current efforts is on improving routability without the need for excess white space insertion.

As part of our current work, we are attempting to improve the quality of the legalizer—we anticipate that another 5 to 10% reduction in wire length can be obtained easily. Our current mixed block placement tool cannot handle fixed macro blocks or obstacles within the placement area, and we are working on methods to handle this as well.

### Acknowledgements:

This work was supported by the SRC under project 947.1, an IBM Faculty Partnership Award, and an equipment grant from Intel. We would like to thank the EDA research staff at the IBM T. J. Watson labs (particularly David Kung), and also Bill Halpin from Intel, for helpful discussions.

## 6. REFERENCES

- [1] S. N. Adya and I. L. Markov. Fixed-outline floorplanning through better local search. In *Proc. IEEE Int. Conf. on Computer Design*, pages 328–334, 2001.
- [2] S. N. Adya and I. L. Markov. Consistent placement of macroblock using floorplanning and standard-cell placement. In *Proc. Int. Symp. on Physical Design*, pages 12–17, 2002.
- [3] S. N. Adya, I. L. Markov, and P. G. Villarrubia. On whitespace in mixed-size placement and physical synthesis. In *Proc. Int. Conf. on Computer Aided Design*, pages 311–318, 2003.
- [4] A. Agnihotri, M. C. YILDIZ, A. Khatkhate, A. Mathur, S. Ono, and P. H. Madden. Fractional cut: Improved recursive bisection placement. In *Proc. Int. Conf. on Computer Aided Design*, pages 307–310, 2003.
- [5] C. J. Alpert. The ispd98 circuit benchmark suite. In *Proc. Int. Symp. on Physical Design*, pages 80–85, 1998.
- [6] M. A. Breuer. A class of min-cut placement algorithms. In *Proc. Design Automation Conf*, pages 284–290, 1977.
- [7] Andrew E. Caldwell, Andrew B. Kahng, and Igor L. Markov. Can recursive bisection alone produce routable placements? In *Proc. Design Automation Conf*, pages 477–482, 2000.
- [8] C. C. Chang, J. Cong, , and X. Yuan. Multi-level placement for large-scale mixed-size ic designs. In *Proc. Asia South Pacific Design Automation Conf.*, pages 325–330, 2003.
- [9] Jason Cong, Lei He, Cheng-Kok Koh, and Patrick H. Madden. Performance optimization of VLSI interconnect

Bench mark	Capo I[2]		Capo II[3]		Capo III[3]		mPG[8]		Feng Shui 2.4				
	HPWL	CPU (min)	HPWL	CPU (min)	HPWL	CPU (min)	HPWL	CPU (min)	HPWL	%Better (Capo)	%Better (MPG)	CPU (min)	Legal. (sec)
ibm01	3.96	18	3.36	13	3.05	20	3.01	18	2.41	26.56	24.90	3	1
ibm02	8.37	31	8.23	240	6.83	11	7.42	32	5.34	27.90	38.95	5	< 1
ibm03	12.16	42	11.53	22	10.38	59	11.20	32	7.51	38.22	49.13	6	2
ibm04	13.48	47	11.93	25	10.11	15	10.50	42	7.96	27.01	31.91	7	< 1
ibm05	11.51	8	11.20	5	11.10	5	10.90	36	10.10	9.90	7.92	8	1
ibm06	10.25	56	9.63	19	9.94	18	9.21	45	6.82	41.20	35.04	10	2
ibm07	15.75	58	15.80	39	15.25	25	13.70	68	11.71	30.23	16.99	13	1
ibm08	21.18	94	18.85	111	17.91	29	16.40	82	13.60	31.69	20.59	16	1
ibm09	19.59	66	17.52	178	19.88	29	18.60	84	13.83	26.68	34.49	15	1
ibm10	60.72	229	53.58	490	45.46	116	43.60	172	37.48	21.29	16.33	22	17
ibm11	28.49	106	26.47	69	29.40	45	26.50	112	19.96	32.62	32.77	21	2
ibm12	51.74	675	55.12	119	55.79	25	44.30	153	35.57	45.46	24.54	23	3
ibm13	39.39	151	33.56	88	37.73	53	37.70	151	24.95	34.51	51.10	26	2
ibm14	56.19	286	52.67	333	50.26	155	43.50	276	38.48	30.61	13.05	52	5
ibm15	70.48	237	64.69	264	65.00	195	65.50	385	52.14	24.07	25.62	87	10
ibm16	N/A	N/A	83.14	580	90.01	162	72.40	436	61.33	35.56	18.05	93	15
ibm17	92.38	443	91.50	249	89.17	188	78.50	606	70.60	26.30	11.19	104	16
ibm18	54.90	378	54.11	397	51.84	127	50.70	437	45.05	15.07	12.54	114	18
Avg										29.16	25.84		

**Table 2: Half perimeter wire length (HPWL) and runtime comparisons for the IBM benchmarks between Capo, mPG, and our tool. For ratio comparisons with Capo, we used their best result. Run times cannot be directly compared: our experiments use 2.5GHz Linux/Pentium 4 workstations, Capo I used 1GHz Linux/Pentium 3 workstations, Capo II and III used 2GHz Linux/Pentium 4 workstations, and mPG used 750MHz Sun Blade 1000 workstations. All run times are in minutes, with the exception of our legalization step, which is in seconds.**

- layout. *Integration, the VLSI Journal*, 21:1–94, 1996.
- [10] A. E. Dunlop and B. W. Kernighan. A procedure for placement of standard-cell VLSI circuits. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, CAD-4(1):92–98, January 1985.
- [11] H. Eisenmann and F. M. Johannes. Generic global placement and floorplanning. In *Proc. Design Automation Conf*, pages 269–274, 1998.
- [12] GSRC. Bookshelf slot. <http://www.gigascale.org/bookshelf>.
- [13] D. Hill. US patent 6,370,673: Method and system for high speed detailed placement of cells within an integrated circuit design, 2002.
- [14] G. Karypis, R. Aggarwal, V. Kumar, and S. Shekhar. Multilevel hypergraph partitioning: Application in VLSI domain. In *Proc. Design Automation Conf*, pages 526–529, 1997.
- [15] J. Kleinhans, G. Sigl, F. Johannes, and K. Antreich. GORDIAN: VLSI placement by quadratic programming and slicing optimization. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 10(3):356–365, 1991.
- [16] C. Li and C.-K. Koh. On improving recursive bipartitioning-based placement. Technical Report TR-ECE-03-14, Purdue University ECE, 2003.
- [17] J.-M. Lin and Y.-W. Chang. TCG: A transitive closure graph based representation for non-slicing floorplans. In *Proc. Design Automation Conf*, 2001.
- [18] H. Murata, K. Fujiyoshi, S. Nakatake, and Y. Kajitani. VLSI module placement based on rectangle-packing by the sequence pair. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 15(12):1518–1524, 1996.
- [19] R. H. J. M. Otten. What is a floorplan? In *Proc. Int. Symp. on Physical Design*, pages 201–206, 2000.
- [20] Yingxin Pang, Florin Balasa, Koen Lampaert, and Chung-Kuan Cheng. Block placement with symmetry constraints based on the o-tree non-slicing representation. In *Proc. Design Automation Conf*, pages 464–467, 2000.
- [21] C. Sechen. Chip-planning, placement, and global routing of macro/custom cell integrated circuits using simulated annealing. In *Proc. Design Automation Conf*, pages 73–80, 1988.
- [22] A. Shanbhag, S. Danda, and N. Sherwani. Floorplanning for mixed macro block and standard cell designs. In *Proc. Great Lakes Symposium on VLSI*, pages 26–29, 1994.
- [23] W. Swartz and C. Sechen. Timing driven placement for large standard cell circuits. In *Proc. Design Automation Conf*, pages 211–215, 1995.
- [24] W. P. Swartz. Automatic layout of analog and digital mixed macro/standard cell integrated circuits. *Yale Thesis, Chapter 4*, 1993.
- [25] M. Upton, K. Samii, and S. Sugiyama. Integrated placement for mixed macro cell and standard cell designs. In *Proc. Design Automation Conf*, pages 32–35, 1990.
- [26] Maogang Wang, Xiaojian Yang, and Majid Sarrafzadeh. Dragon2000: Standard-cell placement tool for large industry circuits. In *Proc. Int. Conf. on Computer Aided Design*, pages 260–263, 2000.
- [27] H. Yu, X. Hong, , and Y. Cai. Mmp: a novel placement algorithm for combined macro block and standard cell layout design. In *Proc. Asia South Pacific Design Automation Conf.*, pages 271–276, 2000.