

Benchmarking For Large-scale Placement and Beyond*

Saurabh N. Adya[†], Mehmet C. Yildiz[‡], Igor L. Markov[†],

Paul G. Villarrubia[‡], Phiroze N. Parakh[‡] and Patrick H. Madden[‡]

[†] The University of Michigan, Department of EECS, Ann Arbor, MI 48109-2122

[‡] SUNY Binghamton, Computer Science Department, P.O. Box 6000, Binghamton, NY 13902

[‡] IBM Corp., 11400 Burnet Road, Austin, TX 78758

[‡] Monterey Design Systems, 894 Ross Drive, Sunnyvale, CA 94089

ABSTRACT

Over the last five years the VLSI Placement community achieved great strides in the understanding of placement problems, developed new high-performance algorithms, and achieved impressive empirical results. These advances have been supported by non-trivial benchmarking infrastructure, and future achievements are set to draw on benchmarking as well. In this paper we review motivations for benchmarking, especially for commercial EDA, analyze available benchmarks, and point out major pitfalls in benchmarking. We outline major outstanding problems and discuss the future of placement benchmarking. Furthermore, we attempt to extrapolate our experience to circuit layout tasks beyond placement.

Categories and Subject Descriptors

B.7.2 INTEGRATED CIRCUITS / Design Aids

C.5.4 VLSI Systems

J.6 Computer-aided design (CAD).

General Terms

Algorithms, Measurement, Documentation, Performance, Experimentation, Human Factors, Standardization.

Keywords

Layout, Placement, Placer, Benchmark, Performance, Routing, Congestion, Signal delay, Timing, Wirelength, Comparison, Evaluation.

1. INTRODUCTION

Progress in VLSI placement research over the last five years has been tremendous. High-performance free placement tools such as Kraftwerk [24], Capo [10], Dragon [54] and Feng Shui [58] are now widely available [13] and used. They have been successfully tested on ever-increasing circuits and are on par with commercial tools as far as simple placement objectives are concerned. More importantly, we now have much better understanding of such important issues as *a priori* interconnect prediction [52], routing congestion [47, 55] and timing [31, 56, 35]. Given that VLSI placement is largely an empirical field, much of this progress would have been impossible without the public availability of large circuit benchmarks [13], such as the 18 ISPD-98 circuits released by IBM [4] and their derivatives [54, 56]. The availability of open-source placers and public placement benchmarks leads to new synergies by allowing researchers to modify the tools and the benchmarks, analyze

*Contact author: Prof. Igor Markov imarkov@umich.edu

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'03, April 6-9, 2003, Monterey, California, USA.
Copyright 2003 ACM 1-58113-650-1/03/0004 ...\$5.00.

tool performance in depth and combine tools to solve new design problems [1]. Many open problems in placement remain, and benchmarking issues, such as independent replication of reported results, are integral to further progress.

The ability to accurately measure the impact of any technique is crucial to scientific advancement. Recently, the physics community had a surprising revelation: a number of papers from a well known researcher were based on fraudulent data [5, 37]. Apparently driven by the desire to publish, experimental results were fabricated, with theory being passed off as fact. Had it not been for the failure of other research groups to replicate experiments, the fraud might have gone undetected. Incorrect published results are damaging to subsequent research; while there is no indication in the Physical Design community of *intentional* misrepresentation of results, reported results often cannot be reproduced [40]. Even the most basic metrics are freely interpreted by different authors, and it is nearly impossible to determine the best approach for a given problem.

As circuits become larger and more complex, the need to improve design automation tools becomes more urgent. According to recent literature, existing tools may produce layouts several times worse than what is achievable. This has been independently shown (i) by comparing manually designed circuits to those laid out by commercial tools [21], and (ii) by placing specially designed circuits with known good layouts [27, 15]. Thus, much room for improvement remains in circuit layout, and even the slowdown of Moore's law would emphasize possible improvements due to EDA tools. Yet, if we cannot reliably compare EDA tools and relevant research results, significant progress is unlikely.

This paper reviews basic motivations for placement benchmarking in Section 2. Major available tools and benchmarks are analyzed in Section 3. Routability is discussed in Section 4 and timing in Section 5. We attempt to extrapolate the lessons learned to layout tasks beyond placement in Section 6.

2. LAYOUT SOPHISTICATION MOTIVATES OPEN BENCHMARKING FOR EDA

As VLSI chips grow in size and complexity, large-scale placement is becoming integral to achieving multiple design objectives. Some of the most important goals are the minimization of wirelength, routing congestion, cycle-time and power dissipation. These objectives may correlate, e.g., wirelength and power, but in some cases conflict with each other. For example, to avoid routing congestion in certain areas of ASIC designs, one may need to spread out collections of cells. This reduces wiring density, but increases wirelength.¹ Similarly, techniques that minimize timing in many cases increase wirelength and congestion. While experiences with specific optimization techniques provide only circumstantial evidence of conflicting objectives, for practical purposes this evidence is strong.

The multiplicity of conflicting objectives makes large-scale VLSI placement extremely complex. Additional complexity is due to design constraints, e.g., signal integrity guidelines, chip die and floorplan constraints, pre-designed on-chip intellectual property, etc.

¹In principle, there may be congestion minimization techniques that do not increase wirelength. These should be considered too.

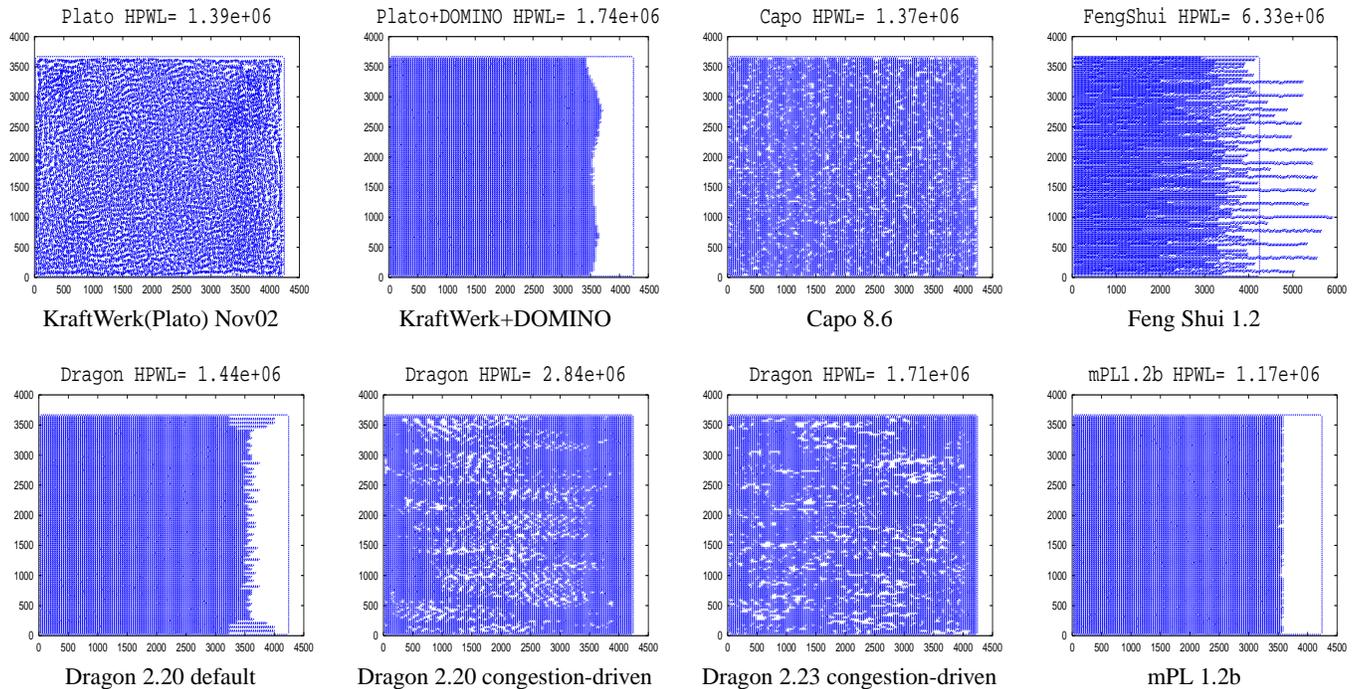


Figure 1: Placements of the 12.5K-cell benchmark PEK001 produced by placers KraftWerk, Capo, Dragon, FengShui, and mPL. The KraftWerk placement has many cell overlaps and must be further legalized with DOMINO, which costs 20% wirelength. The Feng Shui 1.2 placement is not legal either, primarily due to a misinterpretation of row constraints. Feng Shui 1.5 with tuned parameters fixes this problem, and produces a legal placement with wirelength 1.6e6. Capo and KraftWerk tend to distribute whitespace uniformly across the fixed die to generically improve routability, but variable-die placers pack cells in rows to the left to minimize wirelength. In congestion-driven mode, Dragon behaves like a fixed-die placer and allocates whitespace according to an internal congestion map. Wirelengths in Table 2 may differ slightly for randomized placers such as Capo and Dragon.

Modern ASIC designs are laid out in the *fixed-die context*, where the layout area, routing tracks and power lines are fixed before placement starts [10]. Minimized are congestion and timing.² Fixed-die layout is relevant for processes with over-the-cell routing on three or more metal layers and often applied to design blocks rather than whole chips.

Placement density is a new concern implied by the fixed-die context. We define placement density *in a region* as the ratio of (i) total area of movable cells in that region to (ii) the area available for placement of movable cells in the region. Another related term is *whitespace* = $100\% - \text{density}\%$. Similarly, the *average density*, also known as *row utilization*, is defined as the ratio of the total area of *all* movable cells to the total amount of area available for placement of movable cells. It cannot be changed by the placer. If all movable cells are uniformly spread throughout the layout area, the average density will be achieved at all locations. To improve yield, placers may be required to limit density in any given region (uniform distribution of whitespace is one of many ways to satisfy such a constraint). Observe that maximum density values (per region) below the average density are not feasible, and if the maximum density per region equals the average density, then whitespace must be distributed equally. However, placers are often allowed to allocate significant “free space” rather than distribute cells uniformly. Depending on the type of design, this may be an important part of the overall problem. Figure 1 shows how several academic placers handle 15% whitespace in the PEK001 benchmark [15].

Industrial placement instances, e.g., at IBM, can be classified into ASICs, SOCs, and Microprocessor RLMs (random logic macro). Each of these categories presents unique difficulties. ASIC chips generally contain a large number of fixed I/O ports that may be

²Fixed-die placement is a departure from textbook physical design where routing tracks and chip area are determined during placement and routing. As such, it invalidates older benchmarks because area is not minimized anymore.

perimeter-restricted or pervasive throughout the core area of the chip (area-array I/O). ASIC chips frequently contain a handful (1-20) of pre-placed large macros that are fixed, a moderate number (100s) of movable large multi-row cells, and many small movable cells — up to several million and increasing. ASIC chips come in a variety of average densities typically ranging from 40% to 80%.

SOC designs are similar to ASIC designs, but with many more large macros fixed in the placement area. In extreme cases the bulk of the design is concentrated in standard pre-designed library cores, RAMs, etc, with only a small fraction of movable logic providing minor control functions. Such placement instances tend to have extremely low densities on the order of 20%, and in some cases less than 5%. Placement algorithms developed for nearly-full designs often do not handle such extremes well. Therefore one seeks algorithms specifically developed for this context and tested on low-density designs [2].

Microprocessor designs are generally laid out hierarchically, and this approach often leads to many small partitions. Some of these partitions are small standard-cell placement instances with very few fixed cells, a large number of fixed I/O ports, and a small number of movable cells (< 10000). Because densities tend to be high, averaging 80% and reaching 98%, specialized techniques are needed to produce good placements that are also legal [57, 12]. Also, due to the small number of movable objects and the large number of fixed ports [8], multi-level partitioning [3, 32] is no better during placement than simpler “flat” FM partitioning [25]. Related results have been reported in [7].

Given the complexity and the variety of VLSI placement problems, one should not expect a single closed-form algorithm or even a commercial tool to work well in all circumstances. Comprehensive evaluation of algorithms and tools is non-trivial, and so is testing the applicability to each of the relevant domains. Often, algorithms that perform well in the case of an RLM do not perform well for an ASIC (e.g., recursive bisection with “flat” FM partitioning). Likewise, al-

gorithms such as recursive bisection with multi-level partitioning that show significant promise on very large netlists may provide little or no value for small structured components of a microprocessor chip. Some algorithms may perform well on dense designs while others perform well on sparse designs. Other differentiating factors include the diversity of cell sizes and the presence of fixed and movable macros, — these are increasingly important in modern designs.

The sophistication and variety of layout problems, as well as the multitude of performance factors, make the case for public benchmarking in Physical Design. Indeed, we observe a similar situation in computer architecture, where different design decisions may favor different applications, and the variety of microprocessors feeds the need for comprehensive comparisons. To this end, industry-standard evaluation of desktop and server hardware performance across a variety of tasks is based on SPECmark benchmarks [51]. Originally, the Standard Performance Evaluation Corporation (SPEC) published a suite of 10 benchmarks that test a computer's integer and floating point computation. The suite includes slightly hacked versions of well-known FORTRAN and C codes. The performance measure of one SPECmark is comparable to that of a VAX-11/780. Additional SPECmark suites have been published in recent years.

Similarly, an appropriate set of placement benchmarks could become a standard for measuring and categorizing the behavior of placement algorithms. Understanding how algorithms behave across the entire problem space is important for selecting and developing the best techniques, and such an effort seems beyond any single industrial or academic placement group. The industry needs placement benchmarking to improve internal tool development, to measure potential vendor tool offerings, and to communicate important issues to academic researchers. Such benchmarks can greatly enhance the efficiency of communication between all parties involved.

Besides total wirelength and congestion, layout tools must optimize timing and ensure signal integrity. Such design objectives and constraints must be reflected in public placement benchmarks so that one can compare layout tools in a variety of placement contexts. In particular, one would like to see trustworthy empirical data for academic placers that lend themselves to timing closure flows and produce good timing results while maintaining routability and signal integrity. Such community benchmarks are relevant for users of commercial EDA tools, and an investment into producing them is justified in the long term. An attempt at such benchmarks is being made at CMU [49].

3. AVAILABLE OPEN BENCHMARKS AND PLACEMENT TOOLS

As noted in [40], published results for wirelength-driven placement of MCNC benchmarks differ wildly due to creative but poorly explained interpretation of input files by authors. Similarly, we show in Section 5 that more recent published results on timing-driven placement exhibit alarming contradictions. Less drastic differences may go unnoticed, or even be falsely advertised as algorithmic improvements, especially when reported implementations are not available for independent evaluation. For example, some works on placement and floorplanning claim good wirelength, but their placements have many cell overlaps. Another source of discrepancies in published results is poorly specified benchmarks that require pre-processing and additional information, e.g., timing-driven placement benchmarks from [18] published only in Verilog. Limited parsers also cause difficulties, e.g., as of February 2003, the second suite of IBM-Dragon benchmarks in LEF/DEF has two variants — one for Dragon and one for Capo. To this end, we advocate using common, recent benchmarks and standard parsers, e.g., Cadence's open-source LEF/DEF parsers downloadable for free from <http://www.openeda.org>. Below is a survey of existing families of placement benchmarks as well as downloadable placer implementations. We typically cite the publication where a given contribution was first described, but download links for software and circuits can be found in the GSRC Bookshelf [13], specifically in the Wirelength-Driven Placement and Circuit Design Examples slots. Based on empirical data for various pairings of benchmarks and placers, we observe interesting trends.

3.1 Benchmarks

Artificially-generated netlists may be useful for regression-testing and sanity-checking, but placement algorithms are typically validated and compared using netlists derived from real designs. Material in Section 2 explains why.

MCNC benchmarks date back to late 1980s; they are small and outdated [4]. The multitude of their interpretations makes any reported numbers meaningless. Placement variants of MCNC benchmarks converted to the GSRC Bookshelf format are still available in [13] and are sometimes referred to as GSRC-MCNC benchmarks.

In 1998 IBM released 18 netlists with 10K-220K modules as benchmarks for hypergraph partitioning [4]. Despite all design information being sanitized, two out of twenty original netlists were not cleared for public release. However, the remaining 18 benchmarks were soon adapted to placement.

IBM-Dragon [54, 55] benchmarks come with the Dragon placer discussed below and are referred to as IBM-Place by their authors. Several suites of these benchmarks are available on-line. The first suite released in 2000 sizes cells based on the node area in the IBM circuits, and removes all large cells. Because of that, the resulting netlists contain disconnected pads. The second suite maps cells to an Artisan library and adds enough routing information to run Cadence WarpRoute on a given placement. A suite of timing-driven placement benchmarks recently posted by the authors of Dragon is not related to their previous benchmarks, but rather derived from the ISPD '01 suite [18] discussed below. No timing constraints are given (February 03), and other important information is missing.

IBM-Floorplacement [1] benchmarks are also derived from the IBM netlists, but include all original macros. They are available in LEF/DEF and in the GSRC format. Improved wirelengths have been reported recently [16]. Several known placers produce many cell overlaps on these benchmarks and cannot fit all macros inside the layout areas. Therefore we recommend visualizing placements and checking for overlaps before reporting wirelength.

PEKO [15] benchmarks reflect the net-degree distribution of the 18 IBM netlists, but are otherwise generated artificially, with randomization, offering astronomically large irregular netlists to test the scalability of placers [27]. PEKO stands for Placement Examples with Known Optimal wirelength. In particular, in optimal placements each net independently achieves the smallest possible wirelength, making all wires local. The authors conclude that existing placers are 40-100% away from optimal solutions. However, there is a lingering concern that PEKO benchmarks are not representative of industry circuits. PEKO benchmarks have 15% whitespace and come in two suites.

Grids with four fixed vertices and n^2 1x1 movables, such as the one in Figure 2/Capo 8.6, are used in our work to test the behavior of placers on datapath-like circuits, on which many commercial layout tools perform poorly [21]. A simple induction argument shows that there is only one optimal placement for each such "netlist," where each of $2n^2 - 2n + 4$ nets has length 1 (cf. PEKO benchmarks). More importantly, sub-optimality can be visualized to drive debugging efforts. The benchmarks are on-line at [13].

Vertical benchmarks [49] created at CMU attempt to remedy the lack of design information in public circuit benchmarks. They provide multiple representations of real circuits at different stages of design process, including have non-trivial layout features, such as fixed macros. However, most of those circuits have under 50K cells. As of February 2003, details are missing to evaluate signal delay and there are no definitions of timing constraints, clocks or acceptable transition times. The netlists are mapped to a $0.35\mu\text{m}$ -library, thereby making interconnect effect negligible.

Non-benchmarks. The ISPD 01 suite from [18] is available in a hierarchical (not flattened) gate-level Verilog format. There are no timing constraints or gate libraries available. There are multiple top-level signals with prefix "CLK", and it is not clear how clock nets are represented. The authors suggest that a series of proprietary tools be applied to their benchmarks before and after timing-driven placement. However, because of differences in tools (versions, options, cell libraries etc) such pre-processing may lead to different numbers even if the same timing-driven placer is used.

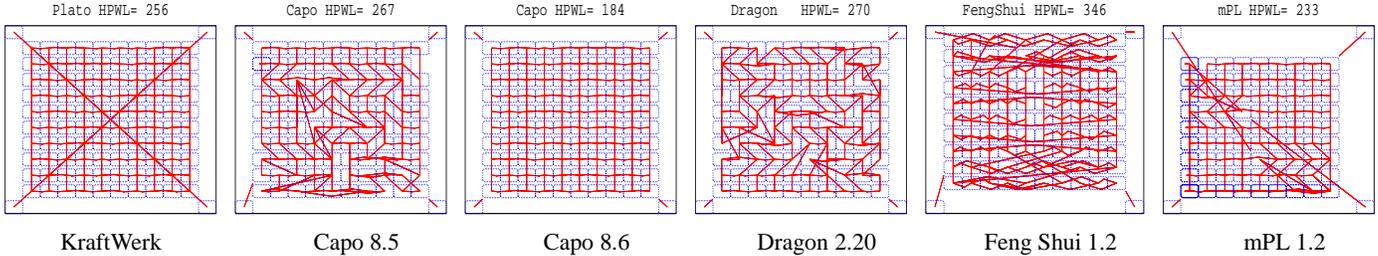


Figure 2: Placements of a 10x10 grid-graph produced by academic placers. All pictures are produced using a utility that ships with Capo and gnuplot. mPL 1.2 places cells with many overlaps, and Capo 8.5 produces one overlap. mPL 1.2b (not shown) and Capo 8.6 both find the unique optimal placement. We ran Feng Shui 1.2 with command-line option `-capo15` to avoid uneven rows as in Figure 1. Unfortunately, that causes Feng Shui to shift locations of all movable cells up by half a row. KraftWerk produces good placements, but the block of movable cells has a wrong orientation relative to fixed pads. For larger grids, none of the placers produce optimal wirelength. The smallest wirelength is achieved by KraftWerk, then Capo 8.6, closely followed by mPL 1.2b. All three are generally within 25% from optimal HPWL. Dragon 2.20 and Capo 8.5 double wirelength.

Feature	Kraftwerk	Capo 8.6	Dragon	Feng Shui	mPL 1.2b
Fixed Terminals	+	+	+	+	+
Macros	±	±	-	-	-
Terminals of arbitrary size	+	+	-	+	-
Cells of arbitrary size	+	+	+	+	-
Subrows	-	+	-	-	-
Orientation constraints	-	+	-	-	-
LEF/DEF	-	+	±	±	-
Whitespace Management	±	±	+	±	-
Detail Placement	-	±	+	±	+
Timing-driven Placement	+	±	+	-	-

Table 1: Features supported by different academic placers. KraftWerk and Capo support fixed macros, but often fail to remove overlaps of movable macros [1]. Being a global placer, Capo comes with a very simple detail placer. CapoT, a timing-driven version of Capo is not open-source at the moment.

3.2 Placers

We describe large-scale standard-cell placers available to multiple research groups; the order reflects when the tools were developed and reported in publications. All placers we use except for KraftWerk directly support the GSRC Bookshelf [13] placement format that is considerably simpler than LEF/DEF.³ Some of them also support subsets of Cadence LEF/DEF, but input problems are common with industrial circuits.⁴

KraftWerk [24] is a force-directed placer. Yet, rather than moving one cell at a time, it solves the Poisson equation (a PDE from mathematical physics). This analytical algorithm often leaves cell overlaps and requires a separate follow-up legalization step. However, overlaps are typically well distributed over the core area and can be removed by a simple built-in legalization algorithm, perhaps at a cost of increased wirelength. This simple legalization may fail and the user is advised to call the variable-die DOMINO [22] detail placer shipped with KraftWerk. DOMINO is based on network-flow algorithms. Both tools target fixed-die layout and tend to distribute whitespace fairly uniformly. KraftWerk is deterministic.

Not having access to source codes of KraftWerk or DOMINO, we obtained Linux executables in November 20002 directly from the authors, who mentioned that so far KraftWerk has not improved wirelength minimization beyond the original 1998 version. The binary for KraftWerk is called Plato.

³In our experiments, Feng Shui 1.2 misinterpreted row information in PEKO01 benchmarks. Newer versions fix that problem.

⁴Capo is now shipped with Cadence’s official LEF/DEF parser, and the users can switch to it from the default native parser.

PE KO # [15]	Optimal WL $\times e^6$	KraftWerk WL $\times e^6$	+DOMINO WL $\times e^6$	Capo 8.6 WL $\times e^6$	Dragon 2.20 WL $\times e^6$	Feng Shui 1.2 WL $\times e^6$	mPL 1.2b WL $\times e^6$
01	0.81	1.39	1.74	1.29	1.46	3.18	1.17
02	1.26	1.98	2.61	2.03	2.43	5.26	1.78
03	1.50	3.02	3.78	2.66	2.93	6.82	2.33
04	1.75	3.25	4.25	3.12	3.87	8.09	2.35
05	1.91	3.92	4.79	3.16	3.79	9.08	2.77
06	2.06	4.07	5.38	3.57	4.35	10.14	2.98
07	2.88	5.73	7.56	5.07	6.24	12.55	4.29
08	3.14	5.87	8.17	5.57	6.79	13.78	4.95
09	3.64	8.52	10.74	6.47	7.72	16.24	5.42
10	3.73	8.9	12.03	8.0	8.49	20.12	7.01

Table 2: HPWL on 10 out of 18 PEKO benchmarks in suite 1. KraftWerk placements are legalized by DOMINO. Feng Shui 1.2 is used with the option `-capo15` that improves results.

Capo [10, 11] is a global fixed-die placer based on recursive min-cut bisection. It uses a built-in multi-level Fiduccia-Mattheyses partitioner [7] written from scratch for this application. All source codes are available in [13]. Capo has a built-in LEF/DEF interface and has been tuned on proprietary benchmarks from Cadence, with successful routing in mind (using WarpRoute or any other tool). Capo uniformizes whitespace to generically improve routability, but may produce unroutable placements for challenging circuits.

Most of the results we report are for Capo 8.6 which somewhat outperforms Capo 8.0 from 2000 [10] but may run slower.⁵ A small number of overlaps is possible after Capo 8.0, therefore the authors of [10] run a commercial placer in a fast ECO mode to fix overlaps before routing. Later versions have a fast greedy built-in overlap remover and a simple detail placer based on optimal placement of small groups of cells [11]. Capo does not use Simulated Annealing at any point, but it is randomized — the best of five independent runs is typically better than the average [6]. The executable used for benchmarking was called `MetaPlacerTest0.exe`.

Dragon [54, 55, 56] performs recursive min-cut partitioning using hMetis libraries [32] and periodically improves global wirelength using Simulated Annealing. In our experiments Dragon sometimes achieves better wirelength than Capo, but may be an order or magnitude slower. In default mode, Dragon packs cells in rows left to right, which practically makes it a variable-die placer. In 2002, Dragon was extended with a congestion-driven mode [55] that distributes whitespace unevenly to mitigate congestion at the price of larger wirelength. Dragon has been tested and tuned on IBM-Place benchmarks in the same tool flow that was used to evaluate Capo 8.0 [10]. Figure 1 shows that congestion-driven mode of Dragon increases wirelength compared to the normal mode. Dragon supports a subset of LEF/DEF. Since the source code is not available, we downloaded Dragon 2.20 binaries in the Fall 2002. The latest version 2.23 is primarily a bug-fix release. Most recently, timing-driven Dragon has been released [56].

⁵Capo can be ran in faster modes [10] via command-line parameters. However, we only report results for the default configuration.

Bnchmrks	Dragon [55]	KraftWerk [24]	Capo 8.6 [10]	mPL 1.2b [15]	FengShui 1.2 [58]
IBM [55]	Good	Poor	Med	N/A	N/A
PEKO [15]	Med	Poor	Med	Good	Poor
Grids	Poor	Good	Good	Med	Poor
Cadence-Capo [10]	Med	Med	Good	N/A	Poor

Table 3: Half-perimeter wirelengths achieved by major academic placers on four benchmark suites. N/A means that no placements were produced due to input problems, crashes or unsupported features. In particular, mPL 1.2b does not support uneven cell sizes and Feng Shui 1.2 could not read IBM benchmarks (fixed in Feng Shui 1.5).

FengShui 1.2 [58] uses *hMetis* libraries [32] for recursive min-cut partitioning and attempts to further improve wirelength by using a native multi-way partitioner. *FengShui* is a variable-die placer and always packs cells in rows to the left. No data were published for *Feng Shui* describing routability with respect to major commercial routers. *Feng Shui 1.2* reported at [58] is available in source code, and *Feng Shui 1.5* is available in executable form (April 2003). We use version 1.2 unless indicated otherwise.

mPL 1.2 [20] is a new multi-level placer that, at the top level, uses a fairly expensive analytical optimization [14] that directly handles non-overlapping constraints. At lower levels mPL uses slot assignment and enumerates permutations of small subsets of cells [26]. At the end, cells are packed to the left by sorting their locations (this is typical of a variable-die placer). A more recent version mPL 1.2b integrates detail placement. Unlike in this paper, mPL wirelength is sometimes reported after the external detailed placer *DOMINO* [22] is applied. We noticed that those versions of mPL are deterministic and always produce the same placement if input is unchanged. mPL 1.2 and, later, mPL 1.2b binaries for Sun/Solaris were provided to us by the authors. We ran them on a 750MHz Sparc Ultra-III processor, whereas all other placers were ran on a 2GHz Pentium4-Xeon running Linux.

Table 1 is a check-list for common features found in placers. *Dragon* and *Capo* have more features than other placers.

3.3 Empirical Analyses

Figure 1 plots the outputs of six placers on the PEKO01 benchmark [15] and suggests that *Dragon* in congestion-driven mode, *KraftWerk* and *Capo* behave like fixed-die placers. The other three simply pack cells in rows to the left, which is typical of variable-die placers. Even when this produces good wirelength (e.g., in mPL), such placements may be unroutable. On the other hand, *Dragon*'s congestion-driven mode doubles wirelength for PEKO01 and seems wasteful as well. As we discovered, *FengShui 1.2* interprets this benchmark incorrectly. The problem is currently fixed in *Feng Shui 1.5* and all cells are placed into the die, the resulting wirelength is much improved and is 2 times away from optimal. Additional empirical data for PEKO benchmarks are given in Table 2 and can be compared to results on benchmarks from the proprietary Cadence-Capo suite [10] shown in Table 4. *Dragon* and *Feng Shui* apparently mishandle multiple sub-rows in a row split by a vertical power stripe.

Grid placements in Figure 2 suggest that (i) it may be difficult for an annealer (in *Dragon*) to place regular structures, (ii) despite good performance, *KraftWerk* seems to ignore connections to fixed terminals. A summary comparison of existing placers using data from multiple benchmark suites is given in Table 3. We hypothesize that the analytical placer *KraftWerk* did not do well on IBM netlists because they have numerous multi-pin nets.

4. BENCHMARKING FOR ROUTABILITY

In the 1980s and early 1990s, works on circuit layout often considered both placement and routing [50, 53]. However, as those tasks became more complex, they were often considered separately in the late 1990s. Such a separation of concerns makes evaluation easier and decreases benchmarking runtime, however, the results may be inconclusive and misleading. Even when placement results

test # [10]	Dragon 2.20			KraftWerk +ECO			Capo 8.6 +ECO		
	WL $\times e^8$	WL $\times e^8$	sec	WL $\times e^8$	WL $\times e^8$	sec	WL $\times e^8$	WL $\times e^8$	sec
1	2.91	4.03	383	2.85	2.95	25	2.64	2.64	73
2	I	I	I	4.01	6.38	185	3.08	3.22	268
3	a/oc	a/oc	a/oc	5.9	6.01	149	5.55	5.56	158
4	oc	oc	oc	10.5	11.4	95	10.5	10.7	163
5	a/5.58	a/5.74	a/2132	t-o	t-o	t-o	5.68	5.72	326
6	a	a	a	t-o	t-o	t-o	155	155	251

Table 4: HPWL results and runtimes of academic placers on Cadence-Capo benchmarks (2.0GHz Pentium4-Xeon). The results for Feng Shui are similar to Dragon's, except for test1 (2.48, 2.73, 110). To legalize placements, we apply Cadence QPlace in the -ECO mode and report wirelength before and after. If a placement is legal, no cells are moved. QPlace -ECO moves almost every cell after KraftWerk, but the wirelength increase is modest. "I" stands for failure to read the input, "a" denotes failure to produce a placement, "oc" reflects numerous cells outside the core region, "t-o" stands for 24-hour time-outs. Dragon aborts in fixed-die mode on three testcases (complaining about the lack of whitespace) and finishes only in variable-die mode on two (test3 and test5). test5 has 33.9K cells and 29.4% whitespace, test6 has 35.5K cells and 0.1% of whitespace.

are evaluated by running a commercial router [10], this is far from explicit routability improvement during placement [47, 55].

The narrow focus on placement, together with attempts at wirelength prediction, lead to the popularity of wirelength-based metrics that *roughly* model routability, are easy to calculate, and can be integrated into an optimization engine. Errors in such metrics can sometimes be tolerated. Indeed, in variable-die designs using channel-based routing model (common when very few metal layers were available), even a poor placement could be routed, although at potentially high cost. Modern fixed-die designs with high utilization, many metal layers and over-the-cell routing model lead to the new phenomenon of *unroutable placements*. The fact that low-wirelength placements are not necessarily routable, motivated recent studies of the routability of different placement methods.

In [10], the *Capo* placement tool was used in a set of experiments on proprietary commercial circuits, for most of which *Capo* placements could be routed using a commercial tool without a great deal of difficulty. While one might conclude that *Capo* placements are *generally routable*, a different conclusion could be drawn from recently published empirical results. In [55], the *Capo* and *Dragon* placement tools were compared using ISPD-98 circuits from IBM [4] that were originally published as hypergraph partitioning benchmarks. The authors of [55] removed large macros, mapped the circuits using an academic $0.18\mu\text{m}$ cell library from Artisan and added artificial routing grids. The resulting benchmarks are publicly available at [13], and when they are placed with *Capo* the same commercial global router used in [10] frequently fails. We were able to reproduce those experiments with *Capo 8.0* (*Capo 8.6* tends to produce better-routable placements, but not as good as *Dragon*). Additional data are given in Table 4, where six out of seven Cadence-Capo benchmarks are placed by *Capo*, *KraftWerk*, *FengShui* and *Dragon*. These results suggest that *Dragon* is tuned to IBM-*Dragon* benchmarks, while *Capo* is tuned to Cadence-Capo benchmarks. Also, it appears that *when optimizing wirelength, one cannot predict if routing will succeed or fail, and prior successes or failures on other circuits are not an indication of future performance*. This has serious repercussions for commercial design teams: the routability of a placement approach may be unknown until actual routing. As for improving placement algorithms, the success or failure gives little insight into what was right or wrong with a placement, or how it may be improved. We need metrics that are good at predicting routability, especially at the early stages of placement.

Before suggesting routability metrics for placement, we note that benchmarking of routing itself is problematic. No consensus exists on global routing objectives, and there are no large, widely used public benchmarks. Despite the availability of many well-known placement benchmarks (MCNC, IBM-*Dragon* and PEKO) in [13]

ibm #	Feng Shui 1.5			Capo 8.5			Dragon 1.13		
	HPWL	%h	%v	HPWL	%h	%v	HPWL	%h	%v
01	6.04e6	52	48	6.23e6	55	45	5.62e6	49	51
02	1.73e7	63	37	1.74e7	63	37	1.49e7	50	50
03	1.58e7	55	45	1.59e7	54	46	1.42e7	51	49
04	1.95e7	58	42	2.09e7	58	42	1.81e7	46	54
05	4.51e7	61	39	4.72e7	61	39	3.92e7	52	48
06	2.30e7	56	44	2.55e7	56	44	2.23e7	46	54
07	4.18e7	68	32	4.08e7	63	37	3.49e7	51	49
08	4.17e7	66	34	4.22e7	64	36	3.73e7	50	50
09	3.63e7	59	41	3.47e7	57	43	3.25e7	55	45
10	6.99e7	70	30	6.78e7	63	37	6.16e7	58	42

Table 5: Half-perimeter wirelengths for placements by Feng Shui, Capo, and Dragon. Results for remaining eight IBM benchmarks are similar. The distribution into horizontal and vertical components reveals large differences — Feng Shui and Capo, based on bisection, produce significantly more horizontal wiring. The use of routing resources on metal layers is a key routing-related concern.

there is nothing comparable for routing. Most papers on global routing use proprietary benchmarks, or test cases that were generated by the authors themselves. Comparisons are frequently made with naïve implementations, or with unnamed commercial tools.

We do not believe that one can easily re-target a layout tool from wirelength-driven metrics to timing and/or power minimization after successful global and detail routing. Therefore we describe an evolutionary transition through a series of simpler metrics that can be incorporated into current work, and provide greater insight into the routability of a placement.⁶

METRIC 1: Simple Congestion Metrics. We explain the reported variations in routability via a detailed examination of results produced by two bisection-based placers, Capo [10] and *Feng Shui* [58], and the annealing based placer *Dragon* [54]. Table 5 shows half-perimeter wirelengths for the three placers on the IBM-derived placement benchmarks [13]. Besides total wirelength, we decompose the wiring into horizontal and vertical components. *While total wirelengths may only differ by 5% or 10% per benchmark, horizontal and vertical demand may differ by a large margin.*

ASIC routing is normally performed using “preferred direction” wiring; clearly, Capo and Feng Shui target significantly higher horizontal demand and significantly lower vertical demand (incidentally, industrial benchmarks often have more horizontal routing resources than vertical). If routing fails for Capo or Feng Shui, but not for *Dragon*, the router likely cannot find a location where a wire can travel horizontally. If routing fails for *Dragon*, but not for Capo or Feng Shui, it is likely that the router cannot find a location where a wire can travel vertically.⁷ The difference in routing demand also suggests how to deal with interconnect layers. If the number of metal layers is odd or differences in routing pitches bias routing supply in one direction, the placer should bias the routing demand accordingly.

Evaluating vertical and horizontal wirelength is easy and helps explain apparently contradictory results. Commercial tools report the two numbers and their sum. Academic tools should do the same.

Extending the simple separation between horizontal and vertical components is a metric similar to channel density. Sweeping through the layout either vertically or horizontally, one can track horizontal and vertical routing demand. “Best-case” and “worst-case” congestion levels for the H and V routing layers can be found. When horizontal and vertical placement densities are compared for Feng Shui, Capo, and *Dragon*, the overall results are similar to those in Table 5.

Our final suggestion for “simple” congestion metrics are those based on “probabilistic” routing models [47, 38, 33] as follows.

- The core area is decomposed into a regular grid of routing tiles.

⁶More sophisticated models for achievable routing were proposed and validated [30], but the community has yet to produce consistent and independently verifiable results even for simple metrics.

⁷If routing succeeds, it may take an unusually long time, indicating layout problems. To avoid congestion, a route may need to detour. Such detours increase path lengths, substantially slowing down maze search in modern routers.

- Each signal net is decomposed into Steiner or spanning trees.
- The “probability” that a given tree edge uses a given tile is computed based on fast combinatorial enumeration of shortest paths.

An open-source implementation of probabilistic congestion maps from [38] is distributed with the Capo placer in [13] and can produce picture files as well as scripts for Matlab and gnuplot — see Figure 3. This estimation method is reasonably fast, can obtain results that are close to those of global routing tools [47, 33]. These estimates should be used with caution, however: good global routing tools may introduce slight routing detours to eliminate congestion problems. Probabilistic models might be considered pessimistic; if the estimates are used to influence the placement process, we may be addressing problems which do not actually exist, and suffer unnecessary wirelength increases.

METRIC 2: Global (and Detail) Routing. Presently, relatively few routers are available publicly. Global routers *Labyrinth* [34] and the *Force-Directed Router* [41] are both downloadable from [13] in source code (in C++ and Java respectively), but their behavior on large circuits may not be representative of commercial routers. Some research groups use commercial tools [10, 55], most frequently Cadence *WarpRoute*. However, commercial tools are impossible to tweak and difficult to integrate with. In particular, commercial tools typically do not save global routing results (which would be convenient for evaluating global placement) but rather offer a monolithic global+detail routing optimization. Furthermore, commercial routers may obscure results by performing sophisticated optimizations. To summarize, we believe that an open infrastructure for global routing should be developed by academic researchers and populated with open-source routers of reasonable quality, tested against commercial tools (similarly to how major academic global placers have been tested). A fast global router can be then embedded into a placer [46] as an estimator.

5. TOWARD OPEN BENCHMARKING FOR TIMING-DRIVEN PLACEMENT

The development of scalable, powerful and robust algorithms for circuit delay minimization during placement is a key challenge in Circuit Layout. It is mentioned regularly in the requests from industry and government funding agencies, but few replicable results have been reported in the literature. While we discuss timing, parallels can be made with power minimization. Barriers to research in timing-driven placement can be summarized as follows.

Lack of non-trivial placement benchmarks with enough information to perform accurate timing analysis. The MCNC benchmarks which have signal direction information use an extremely simple and outdated timing model. Meanwhile, benchmarks derived from academic work are viewed by industrial groups as small and meaningless. “Synthetic” benchmarks are criticized for not accurately modeling “real” circuits.

Accurate circuit-level timing analysis is non-trivial, and accurate device-level timing analysis is computationally expensive.

Actual design parameters are closely guarded industrial secrets, and profoundly influence interconnect delay.

Differences in interpretation that have plagued wirelength-based placers [40] are more problematic in the context of timing optimization. The timing-driven annealing-based placer from [53] reports the longest path delay 798ns for the MCNC benchmark *avqsmall*. In [24], the longest path was improved to only 80ns. A quadrisection driven placer [29] reported a result of 71ns, and most recently, a result of 59.6ns was reported [45]. The improvement in delay for the same circuit *by more than a factor of 10* seems beyond belief, especially considering that the approach of [53] was by no means naïve and their placer implementation has been validated independently. Also note that the *avqsmall* circuit was released in 1989, and clock frequencies of 16.6MHz were not realistic for standard-cell ASICs at that time. At this time, while interconnect delay was important, it by no means dominated system delay. Even if all interconnect delay was eliminated, it is unlikely that the delay of the longest path could be affected to this extent. Further investigation revealed that some path delays reported in [45] are smaller than the sum-of-gate-delays reported in [28] — for the testcase *fract* [45]

computes a path delay $11.91ns$, while [28] produces a lower bound of $18.5ns$ by entirely ignoring interconnect delays.

Aside from inaccurate reporting of design parameters used in timing-driven placement (such as the spacing between cell rows), discrepancies in results are due to the dearth of infrastructure necessary to support timing-driven placement (TDP). While it is easy to verify net cuts reported by partitioning engines and confirm half-perimeter wirelength reported by global placers, it is practically impossible to independently verify timing improvements reported by new TDP algorithms, even if placements produced by them are available. Any consistent public infrastructure for benchmarking in timing-driven placement should address such concerns and, in particular, implement several different path-delay computations. If a newly developed placement tool does not find the expected critical path in a reference placement, this is a clear sign that there is an error in the approach. Being able to easily identify the existence of a problem would be invaluable to the academic researcher. In fact, wirelength reported by most academic placers is consistent with a public evaluator available in the GSRC Bookshelf [13].

One could suggest that the setup-slack (the difference between path arrival time and path required time) reported by a static-timing-analysis (STA) engine should be the final arbiter of the “goodness” of a TDP. Indeed, among recent TDP papers [45, 28, 18, 31, 56, 35] one half [18, 31, 56] do just that. However, some groups may find it difficult to obtaining valid timing constraints, gate models (delay library) and an appropriate technology-file to correctly compute the setup-slack. To overcome the obstacles in using an industrial STA engine, authors frequently report “path-delays” through some gate-delay computation coupled with internally developed STA engines. Such timing analyzers can be simplified by ignoring path-exceptions, multiple-clock domains, delays on primary I/O, and gate-delay modeling and net modeling details. The impact of slope (signal-transition time) on gate-delay is typically ignored, likely making path-delay results erroneous as shorter paths with long nets appear more critical than paths with more stages of logic.

At the heart of TDP lies an inherent compromise between optimization and simulation. Ideally each decision made by the placement engine must be guided by exact setup-slack. However, even one pass of an accurate STA may be prohibitively slow in some cases. In an extreme case, embedding a timing update into passes of a Fiduccia-Mattheyses (FM) partitioner, raises the complexity per pass from linear to quadratic (in the number of movable objects) if each move would have to perform a timing update on the entire fan-in and fan-out cone of the relocated cell. Thus, TDP engines must *approximate* their timing gain. Practical trade-offs are biased toward optimization. Also note that maximizing setup-slack in a circuit is equivalent to maximizing setup-slack on all possible paths, whose number may be exponential in the number of movable objects. Thus a TDP engine is forced into implicit traversals or further approximations, further complicated by false paths. Classical minimization of half-perimeter wirelength does not capture this, and path-unaware net-weighting schemes are inadequate. Path-counting schemes [35] can do better.

Even a single-stage delay along a path cannot be quickly calculated with adequate accuracy and fidelity. First, gate-delay and output-transition time are functions of input-transition time (poor transition time typically affects 2-3 stages downstream). Second, the net topology and the presence of buffers may not be certain at placement. Some researchers approximate net delay using the star model, others use minimum spanning trees [31], easily-computable single-trunk Steiner tree or derivatives [17]. Many papers use Elmore delay for the star net-model and intrinsic slope-independent gate delay. These simplifications would be acceptable if the results were correlated with (in a relative sense) or at least were representative of the actual setup-slack. However, that is often not the case. The notion of “path-potential” was introduced in [24] as a method of demonstrating the timing driven properties of a placement engine in the absence of relevant TDP benchmarking infrastructure. A lower bound for path-delay can be found by running an STA with zero interconnect delays (i.e., just gate delays). Two placements can then be compared by subtracting this lower bound from maximal path delays. However, this would ignore transition times!

Timing constraints add more variety to the TDP problem. Today typical designs non-trivial boundary conditions, false-paths, multi-cycle paths, etc. A placer ignoring these design features may focus on paths irrelevant to the actual clock period. Multiple clock domains with different periods raise new issues. Is the $-0.5ns$ setup-slack on a path clocked at $250MHz$ *more critical* than a similar slack on a path clocked at $50MHz$?

While various design considerations make it extremely difficult to evaluate timing accurately, academic works typically address geometric and graph-theoretic aspects that are also challenging for commercial tools. Indeed, signal paths that detour a lot typically have greater delay than “straight” paths. A simple but non-trivial objective function is given by the total geometric path length (gate delays can be added easily to such formulations). To this end, algorithms that directly attempt to “straighten” critical paths by optimizing geometric path lengths have been proposed [31] and extended to more realistic delay objectives. During such optimization they must ensure that sub-critical paths do not overtake currently-critical paths. These algorithms need only the infrastructure to evaluate the criticality of paths and are accessible to academic groups.

Physical synthesis is a synergistic attempt at design closure via simultaneous placement and logical transforms [23, 42, 39]. While interesting work on Physical Synthesis, with empirical results, already appeared at conferences [36], no replicable timing results are given. In Physical Synthesis, concerns about ignoring transition time are alleviated by interleaving placement transforms with calls to netlist buffering [23]. However, this raises two additional concerns: the netlist or gate sizes may change from one iteration to the next, and regions of the chip may become over utilized, thus requiring powerful legalization methods. An alternative method is to perform the placement optimization within a “virtual buffering” mode [42]. This allows the placement engine to operate on a constant netlist (buffers are not inserted) with a timing analysis mode that minimizes excessive slope effects and correctly accounts for buffer delays. In a gain-based synthesis environment [39] this problem is converted into the task of maintaining the gain on each cell. While the netlist does not change, the sizes of the cells may change (to maintain gain), leading to the need for strong legalization techniques. In either case, transition-time effects may lead to 5-10% larger gate area and further challenges for the TDP engine.

It may be unrealistic to develop a Physical Synthesis environment in academia in the near future because the narrower task of timing-driven placement seem to be hitting serious roadblocks. However, we do envision a set of benchmarks with valid timing constraints, multiple clock-domains and of representative size. This requires access to gate-delay libraries (.lib) and technology files (LEF). Finally, there needs to be a way to independently verify the timing results of the placements. Some necessary infrastructure may be provided by recent efforts at Si^2 that resulted in downloadable software such as OLA [44] and OpenAccess [43], but path-based STA is still missing. The descriptions below are adapted from [44, 43].

OLA is an Application Procedural Interface (API) that can be used by EDA tools for the determination of cell and interconnect characteristics of very deep submicron ICs. OLA is an extension to the Standard for Delay and Power Calculation System, the IEEE 1481-1999 standard. Target applications include timing-driven placement and routing, and OLA attempts to eliminate inconsistent timing data between different EDA tools by using the library vendor’s “golden” delay calculator in all OLA compliant tools.

The OpenAccess API is a C++ program interface to IC design data. The associated reference database is a technology donation from Cadence Design Systems, who is also a member of the OpenAccess Coalition. The API and the reference implementation provide a high performance, high capacity electronic design database with architecture designed for integration and fast application development. Access to the reference database source code is provided to allow companies and academic institutions to contribute to future database enhancements and add proprietary extensions. The database can, in principle, be used in production environments where software maintenance is critical.

6. BEYOND PLACEMENT

To seriously address the huge sub-optimality of existing placement tools [27, 21, 15], one needs to ascertain improvements on industrial circuits. However, published empirical data show that even when two research groups use the same source data, *there are often differences of interpretation, resulting in incompatible numbers and no useful conclusions made from the data*. For example, timing-driven placement benchmarks posted in Verilog [18] prevent reliable comparisons to published numbers, e.g., in [56]. To remedy such incomplete benchmarks, the Vertical Benchmarking project at CMU [49] offers multiple representations of the same design. However their benchmarks still do not have sufficient timing data. On the positive side, recent placement benchmarks better agree in terms of row spacing, pin positions, etc and researchers are more conscientious about such design aspects [40].

Lessons from placement benchmarking are summarized below:

1. Evaluation methods must be explicit to leave minimum room for misinterpretation. Simple open-source evaluation tools should be used to verify the accuracy and correctness of any published result. For example, open-source plotters of placement and congestion, as well as evaluators of wirelength and congestion are distributed with the Capo placer in the GSRC Bookshelf [13]. Linux and Solaris binaries are posted in the *Placement Utilities* slot. Benchmarks should be explicit too, and no preprocessing by user should be assumed. The same input files should be used for all tools compared. When conversion cannot be avoided, standard publicly available converters should be used — we posted such converters in the *Placement Utilities* slot of the GSRC Bookshelf.
2. Raw experimental results are very useful and should be posted on-line. This simplifies the verification of results, and may lead to insights into what a tool did “right” or “wrong” on various problems. In the same vein, the version of each tool should be reported (it’s easy!) or at least the time when each tool was downloaded and the source. This can resolve potential confusion about outdated versions of public EDA tools.
3. Visualizations, especially on small benchmarks, help identifying and diagnosing problems. In the course of our work, the performance of Capo, Feng Shui and mPL was improved through step-by-step analysis of placement process on grid benchmarks. A bug in Dragon 2.20 fixed in Dragon 2.23 is illustrated in Figure 1. We recommend placement results be sanity-checked by plotting (are all cells in the core area, do macros overlap?).
4. Regressions are common when bugs are fixed. Last-minute placer bugfixes sent to us by developers occasionally produced worse results than prior versions. For example, mPL 1.2b placed the PEKO01 benchmark with wirelength 1.17e6 versus 1.09e6 achieved by mPL 1.2. We suspect that this deterministic implementation uses a randomized algorithm with a fixed seed, making the results somewhat chaotic. One could expose randomization, as in Capo and Dragon, to stabilize evaluation via averaging [6].
5. Open-source tools are very valuable as they enable interesting experiments via slight modifications. For example, terminal propagation is not described adequately in placement literature, and the best way to learn successful approaches to it is to look at open-source codes [13]. The same applies to many implementation details of high-performance min-cut partitioning algorithms [9]. Open source also lowers barriers to entry and leads to more meaningful research work. Instead of writing new parsers and basic algorithms, researchers should focus on key aspects of EDA tools.
6. Despite the overall preference for real design benchmarks, artificial testcases with known optimal solutions [27, 15] are becoming popular. Instead of known optimal solutions, bounds on optimal costs will do. Such benchmarks (BEKU) are proposed in [19] for min-cut hypergraph partitioning.

As we focus on more difficult problems, the community must support open benchmarking and tool availability, otherwise we cannot expect much progress.

Benchmarking For Routing Tools.

With variable-die channel-based standard-cell designs, comparing global routing tools was relatively easy. Channel density can be computed directly, and channel routing tools can often achieve the lower-bound target. Feed-throughs are inserted in cell rows; given the length of the longest row and the total channel density, we can obtain a very accurate estimate of chip area after detail routing.

Fixed-die, multilayer over-the-cell global routing is more difficult to evaluate because detail routing is non-trivial and must be decoupled. Technology-specific constraints, e.g., antenna rules, make it impossible to predict successful routing for dense designs [48].

Reasonable metrics for global routers were proposed in [47, 34]:

- Each edge of the global routing graph has a fixed maximum capacity; this is a hard physical constraint, and any routing which exceeds this is infeasible.

- When routing demand is below capacity, successful detail routing is more likely. In [47], 70-80% was proposed as a good objective. If a routing solution exceeds this level for a given edge, the edge is “over capacity”. Reducing the total amount by which all edges exceed the target capacity is a reasonable goal.

- If capacity constraints are met, reduce the total wirelength.

A number of global routing benchmarks were made available in [13] by the authors of [34]. As the community moves toward wider usage of benchmarks, these can be suggested as a reasonable next step. For detail routing, very little is available for benchmarking. Only a few research groups are actively working on detail routing tools, and the problem is made extremely complex due to differing design rules, numbers of routing layers, and performance objectives such as crosstalk, delay, and even lithography related issues.

Delay, Power and Temperature.

Incompatible data published for the MCNC benchmarks `fract` and `avqsmall` suggest wide-ranging interpretations and modeling of signal delay, rise and fall times, etc. Given a placement and routing solution, two researchers may come up with “delay” or “power” numbers that are off by an order of magnitude. If the community is to actively pursue timing-, power- and temperature-driven layout, common frameworks are required to evaluate these objectives. We hope that [43, 44] may provide such frameworks. As for public benchmarks with enough information to evaluate signal delay, we are currently negotiating with our colleagues in the industry and hope to post new benchmarks in the GSRC Bookshelf [13]. However, detailed comparisons including delay will require much more effort and finesse than the comparisons presented in this paper.

Wider Benchmarking Context.

When we consider layout problems identified in “research needs” documents from funding agencies, many areas appear in need of benchmarks, even to reliably verify results of one’s research by experiment. We feel that aside from identifying important problems the community must develop evaluation methods and agree upon them. To be specific, we mention several sample areas where benchmarking could help. *Mixed digital-analog design for SOC and 3-dim integration* raise new layout issues. *The X-routing architecture* with 45-degree wiring may affect basic placement and routing algorithms. *Multiple-voltage* systems are now being developed to reduce power consumption without sacrificing performance. Public benchmarks are lacking for such non-traditional designs despite their relevance to next-generation circuitry. Also *physical verification, reliability and yield issues* are becoming more important every year.

In summary, we propose that the physical design community adopt standards for empirical evaluation and best practices similar to those in the placement community. This could improve the quality of ongoing work on circuit layout as well as the interaction among researchers, practitioners and funding agencies.

Acknowledgments

This work was supported by the Semiconductor Research Corporation, the Gigascale Silicon Research Center, an IBM University Partnership award, an IBM Faculty Partnership award, and equipment grants from Intel. We would also like to thank to Prof. Frank Johannes (TU Munich), Prof. Jason Cong, Prof. Joseph Shinnerl, Min Xie (UCLA), Prof. Andrew Kahng (UCSD) and Xiaojian Yang (Synplicity) for technical discussions and help with placement tools.

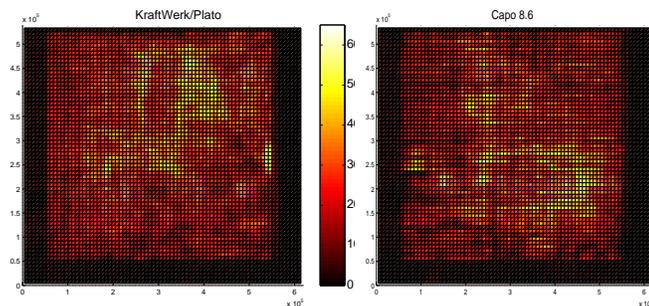


Figure 3: Probabilistic congestion maps for Cadence-Capo test3 (20.5K cells). The Capo placement (right) has higher peak congestion, but lower average congestion than KraftWerk (left). Cadence WarpRoute succeeded in both cases: 6 violations for Capo and 123 for KraftWerk. The plotter program is in the GSRC Bookshelf under Placement Utilities.

7. REFERENCES

- [1] S. N. Adya and I. L. Markov, "Consistent Placement of Macro-Blocks using Floorplanning and Standard-Cell Placement", *ISPD 2002*, pp. 12-17.
- [2] C. J. Alpert, G.-J. Nam and P. G. Villarrubia, "Free Space Management for Cut-Based Placement", *ICCAD 2002*, pp. 746-751.
- [3] C. J. Alpert, J.-H. Huang and A. B. Kahng, "Multilevel Circuit Partitioning", *DAC 1997*, pp. 530-533.
- [4] C. J. Alpert, "The ISPD98 Circuit Benchmark Suite," *ISPD 1998*, pp. 80-85. <http://vlsicad.cs.ucla.edu/~cheese/ispd98.html>
- [5] M. R. Beasley, S. Datta, H. Kogelnik, H. Kroemer, and D. Monroe. "Report of the Investigation Committee on the Possibility of Scientific Misconduct in the Work of Hendrik Schon and Coauthors," 2000. http://www.lucent.com/news_events/researchreview.html
- [6] A. E. Caldwell, A. B. Kahng, A. A. Kennings, and I. L. Markov, "Hypergraph Partitioning for VLSI CAD: Methodology for Reporting, and New Results," *DAC 1999*, pp. 349-354.
- [7] A. E. Caldwell, A. B. Kahng and I. L. Markov, "Improved Algorithms for Hypergraph Bi-partitioning," *ASPDAC 2000*, pp. 661-666.
- [8] A. E. Caldwell, A. B. Kahng and I. L. Markov, "Hypergraph Partitioning With Fixed Vertices," *IEEE Trans. on CAD*, vol. 19, no. 2, 2000, pp. 267-272.
- [9] A. E. Caldwell, A. B. Kahng and I. L. Markov, "Design and Implementation of Move-based Heuristics for VLSI Hypergraph Partitioning", *ACM Journal of Experimental Algorithms*, vol. 5, 2000 <http://www.jea.acm.org/volume5.html>
- [10] A. E. Caldwell, A. B. Kahng and I. L. Markov, "Can Recursive Bisection Alone Produce Routable Placements?" *DAC '00*, pp.477-82.
- [11] A. E. Caldwell, A. B. Kahng, I. L. Markov, "Optimal Partitioners and End-case Placers for Standard-cell Layout", *IEEE Trans. on CAD*, vol. 19, no. 11, 2000, pp. 1304-1314
- [12] A. E. Caldwell, A. B. Kahng, I. L. Markov, "Hierarchical Whitespace Allocation", to appear in *IEEE Trans. on CAD 2003*.
- [13] A. E. Caldwell, A. B. Kahng, I. L. Markov, "VLSI CAD Bookshelf" <http://vlsicad.eecs.umich.edu/BK>
- [14] T. Chan, J. Cong, T. Kong, and J. Shinnerl, "Multilevel Optimization for Large-scale Circuit Placement," *ICCAD 2000*, pp. 171-176.
- [15] C. C. Chang, J. Cong and M. Xie, "Optimality and Scalability Study of Existing Placement Algorithms," *ASP DAC 2003*, pp. 621-627.
- [16] C.-C. Chang, J. Cong, and X. Yuan, "Multi-level Placement for Large-Scale Mixed-Size IC Designs," *ASPDAC 2003*, pp. 325-330.
- [17] H. Chen, C. Qiao, F. Zhou and C.K. Cheng, "Refined Single Trunk Tree: A Rectilinear Steiner Tree Generator For Interconnect Prediction," *Intl. Workshop on System-Level Interconnect Prediction (SLIP) 2002*, pp. 85-89.
- [18] Y.-C. Chou and Y.-L. Lin, "A Performance-driven Standard Cell Placer based on a Modified Force-directed Algorithm," *ISPD 2001*, pp. 24-29.
- [19] J. Cong, M. Romesis, and M. Xie, "Optimality, Scalability and Stability Study of Partitioning and Placement Algorithms," *ISPD '03*.
- [20] J. Cong and J. R. Shinnerl, "Multi-level Optimization in VLSI CAD," *Kluwer*, Boston, 2002.
- [21] W. J. Dally and A. Chang, "The Role of Custom Design in ASIC Chips", *DAC 00*, p. 643-647.
- [22] K. Doll, F. M. Johannes and K. J. Antreich, "Iterative Placement Improvement By Network Flow Methods". *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol.13, (no.10), Oct. 1994, pp. 1189-1200.
- [23] W. Donath et al., "Transformational Placement and Synthesis", *DATE 2000*, pp. 194-201.
- [24] H. Eisenmann and F. M. Johannes, "Generic Global Placement and Floorplanning", *DAC 1988*, p. 269-274.
- [25] C. M. Fiduccia, R. M. Mattheyses, "A Linear-Time Heuristic For Improving Network Partitions", *DAC 1982*, pp. 171-181.
- [26] S. Goto, "An Efficient algorithm for the Two-Dimensional Placement Problem in Electrical Circuit Layout," *IEEE Trans. on Circuits and Systems*, vol. 28 no. 1, 1981, pp. 12-18.
- [27] L. Hagen, J. H. Huang, and A. B. Kahng, "Quantified Suboptimality of VLSI Layout Heuristics", *DAC 1995*, pp. 216-221.
- [28] W. Halpin, C. Y. Roger Chen, and N. Sehgal, "Timing driven placement using physical net constraints," *DAC 2001*, pp. 780-783.
- [29] D. J.-H. Huang and A. B. Kahng, "Partitioning based standard cell global placement with an exact objective," *ISPD 1997*, pp. 18-25.
- [30] A. B. Kahng, S. Mantik and I. L. Stroobandt, "Requirements for Models of Achievable Routing," *ISPD 2000*, pp. 4-11.
- [31] A. B. Kahng, S. Mantik and I. L. Markov, "Min-max Placement For Large-scale Timing Optimization" *ISPD 2002*, pp. 143-148.
- [32] G. Karypis, R. Agarwal, V. Kumar, and S. Shekhar, "Multilevel Hypergraph Partitioning: Applications in VLSI Design", *DAC 1997*, pp. 526-529.
- [33] P. Kannan, S. Balachandran, and D. Bhatia, "On Metrics for Comparing Routability Estimation Methods for FPGAs," *DAC 2002*, pp. 70-75.
- [34] R. Kastner, E. Bozogzadeh, and M. Sarrafzadeh, "Predictable Routing," *ICCAD 2000*, pp. 110-113.
- [35] T. Kong, "A Novel Net Weighting Algorithm for Timing-Driven Placement", *ICCAD 2002*, pp. 172-176.
- [36] P. Kudva, A. Sullivan and W. E. Dougherty, "Metrics for Structural Logic Synthesis", *ICCAD 2002*, pp. 551-556.
- [37] E. Lerner, "Fraud Shows Peer-review Flaws," *The Industrial Physicist*, 8(2), 2002.
- [38] J. Lou, S. Krishnamoorthy, H. S. Sheng, "Estimating Routing Congestion using Probabilistic Analysis," *ISPD 2001*, pp 112-117.
- [39] Magma Design Automation Inc., "White Papers," <http://www.magma-da.com/whitepapers.html>
- [40] P. H. Madden, "Reporting of Standard Cell Placement Results," *IEEE Trans. on CAD*, 21(2), Feb. 2002, pp. 240-247.
- [41] F. Mo, A. Tabbara, and R. K. Brayton, "A Force-directed Maze Router," *ICCAD 2001*, pp. 404-408.
- [42] Monterey Design Systems, "Dolphin," <http://www.mondes.com/products/dolphin.htm>
- [43] Open Access, <http://www.si2.org/openaccess/>
- [44] Open Library Architecture (OLA), <http://www.si2.org/OLA/>
- [45] S.-L. Ou and M. Pedram, "Timing-driven Placement based on Partitioning with Dynamic Cut-net Control", *DAC 2000*, pp. 472-476.
- [46] P. N. Parakh, R. B. Brown, K. A. Sakallah, "Congestion Driven Quadratic Placement", *DAC 1998*, pp. 275-278.
- [47] A. Rohe and U. Brenner, "An Effective Congestion Driven Placement Framework," *ISPD 02*, pp. 6-11.
- [48] L. Scheffer and E. Nequist, "Why interconnect prediction doesn't work," *Intl. Workshop on System-Level Interconnect Prediction (SLIP) 2000*, pp. 139-144.
- [49] H. Schmit, "Vertical Benchmarks," http://www.ece.cmu.edu/~herman/html/benchmark_slot.html
- [50] G. Sigl, K. Doll and F. M. Johannes, "Analytical Placement: A Linear or Quadratic Objective Function?" *DAC 1991*, pp. 57-62.
- [51] The Standard Performance Evaluation Corporation (SPEC), "SPECmark benchmarks," <http://www.specbench.org/>
- [52] D. Stroobandt, "A Priori Wire Length Estimates for Digital Design," 324 pages, *Kluwer*, ISBN 0-7923-7360-X, 2001.
- [53] W. Swartz and C. Sechen, "Timing-Driven Placement For Large Standard-Cell Circuits," *DAC 1995*, pp. 211-215.
- [54] M. Wang, X. Yang and M. Sarrafzadeh, "Dragon2000: Standard-cell Placement Tool for Large Industry Circuits," *ICCAD 2000*, pp. 260-263.
- [55] X. Yang, B.-K. Choi and M. Sarrafzadeh, "Routability Driven White Space Allocation for Fixed-Die Standard-Cell Placement," *ISPD 2002*, pp. 42-50.
- [56] X. Yang, B.-K. Choi and M. Sarrafzadeh, "Timing-Driven Placement using Design Hierarchy Guided Constraint Generation," *ICCAD 2002*, pp. 177-184.
- [57] X. Yang, B.-K. Choi, and M. Sarrafzadeh, "A Standard-Cell Placement Tool for Designs with High Row Utilization," *ICCD 2002*, p. 45-47.
- [58] M. C. Yildiz and P. H. Madden, "Improved Cut Sequences for Partitioning Based Placement," *DAC*, 2001, pp. 776-779.