Advancing Placement

Andrew B. Kahng
UCSD Depts. of CSE and ECE, La Jolla, CA 92093-0404 USA
abk@eng.ucsd.edu https://vlsicad.ucsd.edu/~abk/

ABSTRACT

Placement is central to IC physical design: it determines spatial embedding, and hence parasitics and performance. From coarse- to fine-grain, placement is conjointly optimized with logic, performance, clock and power distribution, routability and manufacturability. This paper gives some personal thoughts on futures for placement research in IC physical design. Revisiting placement as optimization prompts a new look at placement requirements, optimization quality, and scalability with resources. Placement must also evolve to meet a growing need for co-optimizations and for cooperation with other design steps. "New" challenges will naturally arise from scaling, both at the end of the 2D scaling roadmap and in the context of future 2.5D/3D/4D integrations. And, the nexus of machine learning and placement optimization will continue to be an area of intense focus for research and practice. In general, placement research is likely to see more flow-scale optimization contexts, open source, benchmarking of progress toward optimality, and attention to translations into real-world practice.

ACM Reference Format:
Andrew B. Kahng, UCSD Depts. of CSE and ECE, La Jolla, CA 92093-0404 USA, abk@eng.ucsd.edu https://vlsicad.ucsd.edu/~abk/. 2021. Advancing Placement. In *Proceedings of the 2021 International Symposium on Physical Design (ISPD '21), March 22–24, 2021, Virtual Event, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3439706.3446884

1 INTRODUCTION

Placement is at the heart of physical implementation of hardware systems because it defines spatial embedding. Figure 1 (left) shows the convergence of physical implementation through traditional flow stages: design syntheses converge logic and spatial embedding, which enables estimates of parasitics, coupling, timing and power to correspondingly converge in design analyses and verifications. *Optimization* of the design outcome is both a backdrop and an overarching goal: the IC design process itself is a complex optimization that must be performed within a given "box" of resources (servers, licenses, people, weeks of schedule).

Placement is conjointly optimized with logic, performance, clock and power distribution, routability, and manufacturability. At a next level of detail, there are many interleaved steps. Just a few examples: buffering and sizing (for slack, slew and cap load), multi-bit flop clustering, useful skew, hold padding, low-power (clock and power gating, voltage islands, etc.), and pin access and sub-metal improvements. The co-optimization of placement with logic synthesis and timing has led to numerous placement-based synthesis and timing-driven layout techniques seen in today's *physical synthesis* [5]. The co-optimization of placement with routability and

ISPD '21, March 22–24, 2021, Virtual Event, USA.
ACM ISBN 978-1-4503-8300-4/21/03.
https://doi.org/10.1145/3439706.3446884

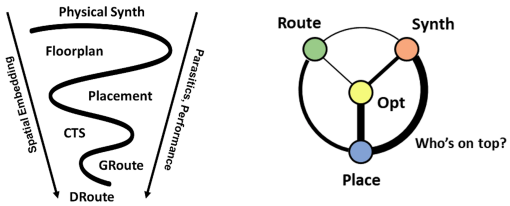

Figure 1: Left: convergence of spatial embedding, parasitics and performance in physical implementation. Right: The interlock of synthesis, placement, routing and optimization.

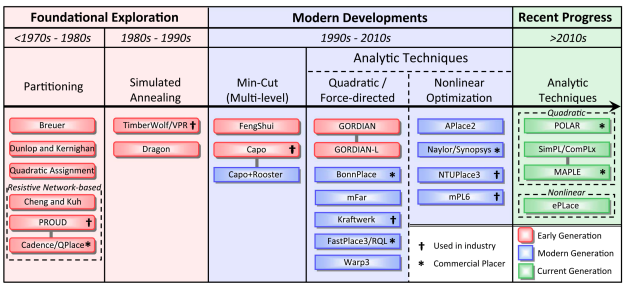

Figure 2: A history of placement methods, from [79].

manufacturability, most recently in light of pin access and design rule checks that the router alone cannot resolve, has led to tighter connections between placement and routing engines in modern place-and-route (P&R) tools. Figure 1 (right) shows the complete graph of interlocks between synthesis, optimization, placement and routing, with stronger links depicted by wider traces. ***[Challenge 1:** Improve understanding of the question, "Who's on top?"]*[1]

Scope, history and key challenges for VLSI placement research are superbly summarized in "Progress and Challenges in VLSI Placement Research" [79] [78] by Markov, Hu and Kim. [79] reviews 50 years of progress in the field, according to a taxonomy of wirelength-driven placement, mixed-size placement, routability-driven placement, timing- and power-driven placement, and physical synthesis integration. Landmark works across several eras are visualized in Figure 2, reproduced from [79]. Tables 1 through 7 of [79] summarize literatures for legalization and detailed placement; mixed-size placement; congestion estimation and routability-driven placement; timing- and power-driven placement; and placement for physical synthesis. The paper also discusses benchmarking and open challenges for the field.

[1] In other words, who will ultimately need to "drive"? ([11] suggested that routing would ultimately need to drive.) The left part of the figure reflects orthogonalization/separation of concerns and localized "A-B-C" loops. The right part reflects a "co-equals" power-sharing perspective that begins with unified and common incremental syntheses, analyses, and underlying data structures. (The trace widths also reflect gradual unifications over the past 25+ years.) There is a long-standing tension between realities of R&D organization and product architecture in commercial EDA (left) and "philosophy" (right).

The remainder of this paper offers some personal thoughts on future directions and challenges for placement research. The discussion broadly buckets these directions and challenges into Optimization; Co-Optimization and Co-Operation; "New" Problems; Learning and Placement; and Support for Placement Research. Section 7 comments on how the review of [79] might be updated, and gives other conclusions as well.

2 PLACEMENT OPTIMIZATION

Future placement research will see renewed attention to placement as *optimization* that lives in a given "box" of resources. Optimization within resource limits is central to wringing design quality and value out of a given design enablement. And, because placement optimization determines spatial embedding, it is central to (accurate) design space exploration that will increase efficiency of the design process without leaving product quality on the table.

There are two basic, intertwined needs. (i) First, design process efficiency must be scaled by "seeing ahead", i.e., predicting outcomes of current and downstream design optimization steps. Inability to predict means guardbanding or divergence, both of which are undesirable.[2] In particular, we need new placement optimizations that are more predictable. (ii) Second, placement research must reexamine "where corners were cut" in the deployment of optimization methods. The goal is to recover solution suboptimality that has been left on the table over the course of decades, as the EDA industry and research were driven by turnaround time requirements (and, as IC designs and EDA were backstopped by process scaling). Going forward, any solutions to (i) and (ii) will require exploitation of modern (cloud, multi-core, GPU) compute substrates as well as application of machine learning.

2.1 Predictability and Stability

Predictors within the physical implementation flow (see also Section 6 below) are useful for identifying and terminating "doomed" floorplans and placements. Other predictors shift the cost-accuracy tradeoff of analysis, which tightens guardbands in optimization. A number of "high-value targets" for prediction [52] center around placement and the need for convergent co-optimizations. These include (i) floorplan quality evaluation, (ii) power delivery network synthesis that maximizes performance while maintaining routability, and (iii) prediction of a block's PPA change when the floorplan (shape, IO placement) context changes.[3] These targets remain open due to a confounding aspect of today's placement and other physical design optimizations: *predictability of outcomes decreases with increased solution quality.* In other words, heuristics and tools become noisy and *chaotic* when they are pushed to their limits [53].[4] This manifests as "dancing floorplans" or other placement instability that is not well-received by tool users. Prediction is even more challenging because structurally dissimilar solutions can have similar quality metrics such as total wirelength or total negative slack, but *which* paths are critical or *which* layout regions are congested can differ greatly between these solutions.

[2]Today's flows and methodologies do not tolerate big loops (e.g., "redo the floorplan" or "redo the clock tree"). Because optimism risks loops, in-built estimates and predictions of parasitics, timing, congestion, etc. already have significant pessimism. Hard "max" criteria (fixed-die routability, path timing) also induce pessimism.

[3]Figure 1 (left) may be recast in the light of accuracy versus information. More information about the design is known further along in the flow. Predictions offer the potential of shifting the accuracy vs. information curve.

[4]Notions of chaos have been called out in [51] [31], and implications of a "noise floor" on prediction accuracy are noted in [18].

Due to the above, it is difficult to make *actionable* predictions in today's regime of optimizers being pushed to their limits, and users being unwilling to give up even 1% of solution quality. The presence of black-box downstream steps such as CTS or routing only adds to the difficulty. ***[Challenge 2:** Develop more predictable placement optimization methods.]* Formal criteria for stability or predictability must be developed, as well. ***[Challenge 3:** Orchestrate placement methods on modern (parallel, distributed, AI) compute substrates with predictability of outcomes (from ensembles of optimization runs) as the driving criterion.]* When placement is performed using thousands of threads, this by itself can confer predictability – if we can model and predict the distributions and order statistics of optimization outcomes [56]. Orthogonally, evolutionary or "parallel problem-solving from nature" classes of metaheuristics map well to modern substrates. Such methods as "go with the winners" [3], adaptive multistart [12], particle swarm optimization [64], NGSA-II [34], etc. will need to be revisited.

Chaos in placement optimization refers to non-smoothness: very small changes to the input result in very large changes to the output. ***[Challenge 4:** Develop placement optimizers that have (provably) smooth behavior.]* Here, *smooth* is in the sense of a Lipschitz criterion: if two placement instances differ by at most some distance δ, then their solutions should differ by at most some distance $f(\delta)$. The definition of "distance" is itself an open question. Useful stability of outcomes may also be obtained if we cast placement optimization as a trajectory of incremental optimizations [60] [30]. The key is that a sequence of incremental optimizations should not gradually lose solution quality. ***[Challenge 5:** Develop incremental and re-entrant placement methods that maintain from-scratch solution quality.]* In addition, smoothness or stability can be achieved by primitive "tethering": specified parts of the placement solution (e.g., for macros or latches) must remain within some ball of radius r with respect to locations in some given (previous) solution.[5] ***[Challenge 6:** Develop methods to solve the tethered placement problem.]*

We may also ask, *"What should be predicted?"* and *"How will predictions be used?"* For the former, just predicting achievable solution quality (e.g., total power, or zero post-route DRCs) is relatively useless without a certificate, such as an explicit tool setup and run-script that will achieve such a solution. As noted in Section 5 below, we still lack understanding of which within-flow figures of merit signify good future outcomes. For the latter, [18] points out how having oracular knowledge of a *part* of an optimizer's solution (e.g., the eventual placed locations for half the macros in a P&R block, or half of the swap-cell list at the end of leakage optimization) can easily degrade placement solution quality. I.e., actions based on prognostication change the outcome. Hence: "be careful what you ask for (and how you will use it)" when pursuing modeling, prediction and/or machine learning in placement.

2.2 Closing the Suboptimality Gap

In optimization, the reality is "Better, faster, cheaper — pick any two." Curiously, in the EDA field we tend to insist (as customer to EDA vendor, or as academic peer reviewer to someone with a new approach to placement) on "I want all three". We are trained to formulate and attack difficult optimizations using ILPs, min-cost flows, assignment, dynamic programming, satisfiability, and so on [71]. But then: "We need the answer overnight", "The approach is

[5]Note that tethered placement is not placement with fixed instances (which are known to disrupt current methods). It is more akin to extreme fencing (which is also not handled well today).

impractical due to its runtime", or "While the method improves wirelength by 1%, this is not a fair comparison because runtimes are three times longer". Such messages drive us to cut corners, and to add more heuristics on top of existing stacks of heuristics.[6]

In Figure 3 (left), heuristic B clearly dominates heuristic A. Heuristic B also seems to dominate heuristic C – until the computational resource is expanded, and C pulls ahead. Thus, B and C *together* define the quality-runtime Pareto (cf. studies such as [15]). It is unfortunate when heuristic C never sees the light of day – particularly since we have little idea how close any of these are to reaching optimality. Fields such as machine vision advance solution quality with a culture of benchmarking and with computational resource arguably as a secondary concern (Figure 3 (right)).

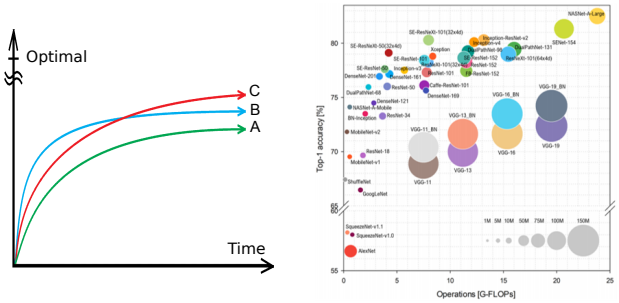

Figure 3: Left: Optimization quality is monotone in resources applied, and multiple optimizers together define the resource-quality Pareto. Right: Excerpt from Figure 1 of [10], showing advance of deep learning for ImageNet-1k.

Multiple vectors are needed to close the suboptimality gap. To begin with, research must have proper aiming points: optimality *in what sense*. Formulations (objectives, constraints) and benchmarks should be clean, "core", long-lived drivers of relevant research. In this light, several past criteria (metrics, evaluation scripts and tools, contest enablements) live on today but likely need to be refreshed. *[**Challenge 7:** Update core placement optimization formulations, and develop long-lived benchmarks and solution criteria.]*

The fraught topic of *benchmarks* merits revisiting with more urgency than in the past. Absent any authoritative source (a la Geekbench or Underwriters Laboratories) since the 1990s (MCNC), benchmarks for placement have been episodically created and passively received. "Real" benchmarks are obfuscated (e.g., module hierarchy stripped), incomplete (e.g., no clock), non-vertical (only a placer needs to be fed), and past any competitive relevance (i.e., old). Recent initiatives such as RISC-V, POSH, FOSSi, CHIPS Alliance, and OpenHW Group hold the promise of more complete and modern benchmarks. Scale and diversity will also be needed to drive research toward next-generation capabilities. And crucially, with real benchmarks the remaining suboptimality gap is unknown.

Artificial netlist constructions have balanced between realism, known optimal solution quality, scalability and other desiderata. Real designs can be perturbed to retain some known achievable solution cost [39] [62], or perturbed by scaling to yield bootstrapped lower bounds on heuristic suboptimality [41]. Alternatively, by sacrificing realism, optimal solutions can be planted such that heuristic suboptimality can be exactly measured [22] [29]. Other constructions produce artificial netlists that try to match prescribed topological criteria (in/outdegree distributions, IO counts, path depths, sequential/combinational instance counts, etc.) [32] [93] [50] [70] [67]. *[**Challenge 8:** Develop and widely adopt generators and suites of artificial testcases that are representative (to optimizers) of diverse design types and future instance complexities.]*

Another facet of closing the suboptimality gap is the use of (modern) computational resources. Past focus has been on single-threaded or single-server turnaround times. Future solution quality gains will draw on massive increases in compute, through use of cloud, distributed/federated, GPU, etc. platforms. *[**Challenge 9:** Develop and improve placement methods with solution quality that increases monotonically with the given "footprint" (threads × runtime) of computational resource.]* This recalls and extends the notion of *anytime* optimization [103].

Improved research infrastructure such as open-source codes, PDKs and benchmarks (see Section 6) will also help advance placement optimizations. The urgency of closing optimization quality gaps suggests that we might aim higher. *[**Challenge 10:** Establish an "Underwriters Laboratories" for measurement, benchmarking of IC design automation and designs.]* Advances can also be supported by a roadmap of optimization requirements and capabilities.

3 CO-OPTIMIZATION AND CO-OPERATION

Tight integration of more optimization engines on common structures and more common analysis engines together enable on-demand constructive estimations and reduced miscorrelation. This reflects a steady trend toward *co-optimization* (multiple objectives) and *co-operation* (multiple engines).[7] Even with a fixed netlist, the placer co-optimizes multiple objective function terms: wirelength (power), timing, legality (density spreading), IR and routability (congestion). Routability is qualitatively more difficult to capture: a placement is known to be routable only after it has been successfully routed, but this depends on a specific router's long-running heuristic in a fixed-die context. *[**Challenge 11:** Develop improved congestion formulations for use in analytic placement.]*

The Art of Co-Optimization. How the placer co-optimizes multiple objectives, subject to constraints, is still very much an art. (i) The importance of objectives can shift across iterations, and across global-detailed-legalization stages. E.g., wirelength, timing, density, congestion, non-overlap and pin accessibility might each take the spotlight in turn. (ii) Constraints such as non-overlap or setup timing are fungible with objectives – via weights, penalties, and other forms of "softness" or relaxations. (iii) Considerations such as edge-type conflict can be masked by whitespace (bloating or padding) that is removed during later placement steps. (iv) Other considerations such as local pin access pattern might be ignored until last-mile fix-ups at the end of detailed placement.

A first observation is that in co-optimization, there is an overarching issue of "What should the placer know, and when should the placer know it?" What can be ignored in global placement, and safely fixed up afterward – with what solution quality gain or loss? When should soft constraints be reverted to hard constraints – along what schedule? When should (constructive, under the hood) predictions be invoked – such as virtual buffering/sizing or timing-driven global routing? *[**Challenge 12:** Improve foundational understanding*

[6] This has come at a cost. Arguably, we are as ill-informed about suboptimality gaps for classical EDA optimizations, and about the potential benefits of long-running distributed CAD optimizations, as we were 20+ years ago [14] [8]. But in today's era of optimization, 1% matters.

[7] From the designer standpoint, a holy grail is to know "the cost of X", where X = incremental area, speed, robustness, low-power, etc., in a high-dimensional Pareto. Placers and other optimizers in PD do not directly support this yet.

of how objective function estimation accuracy, constraint relaxation and instance evolution determine solution quality.][8]

A second observation is that required amounts of "masked" or "ignored" guardbanding in global placement depend on the strength of detailed placement optimizations, e.g., for sub-metal rule compliance or pin accessibility. Weaker detailed placement requires larger upstream guardbanding. ***[Challenge 13:** Develop stronger, distributable "heavy optimizations" in detailed placement.]* Use of branch-and-bound, ILP/SAT/SMT, and/or high-dimensional DP [87] [15] [43] [33] [42] can better optimize routability and other detailed placement objectives. This reduces upstream guardbanding while improving chances of routing success.

Co-Optimization Through Co-Operation. Co-optimizations can require co-operation between placement and other engines. Examples include (i) physical floorplanning, where macro placement and global placement together determine block placements, channel widths and halos, fences and density screens; and (ii) physical synthesis (place-opt), where synthesis and global placement together optimize buffer trees and other parts of the netlist as spatial embedding and timing evolve. Other partnerships are between (iii) placement and global routing, (iv) placement and power delivery network synthesis, (v) placement and clock distribution synthesis, and (vi) detailed placement and detailed routing.

Co-operation is typically via coarse interleaving, reflecting the need to reach a fixed point in a "chicken-egg loop". In the netlist-placement loop, buffering, sizing, and various local transforms are interleaved with incremental placement. There is considerable art in the orchestration of objectives, granularities and degrees of freedom. For (ii), (iii) and (iv) especially, the interleaving is tedious when high-quality (as opposed to quick-and-dirty) solutions are needed at each step. ***[Challenge 14:** Develop methods to directly solve for the fixed point of simultaneous "chicken-egg" optimizations.]* Tightly coupled co-operations are seen in (i) and (v), where quality of results is still a key challenge. Especially: ***[Challenge 15:** Develop improved conjoint optimization of placement and clock distribution.]*[9] The example of (vi) reflects the difficulty in advanced nodes for either the placer or the router to ensure success of the P&R outcome.

A final comment: Placement and other engines can co-evolve into evolutionary niches, whether via academic contests or commercial R&D. Anecdotally, this can be seen when companies each have well-performing P&R platforms, but transplanting any individual {synthesis, placement, CTS, routing} step into another platform worsens results. ***[Challenge 16:** Develop foundational understanding of placement-centered co-optimizations.]*

4 "NEW" FOCI FOR PLACEMENT

Apart from what is pointed out in other sections, several placement topics are likely to receive focused attention in the near term.[10] Lack of attention and investment has allowed several topics (analog, datapath, 3D, power planning, IO planning, ...) to become bottlenecks today, even though their inevitability was well-understood decades ago. Also, several topics listed can be seen as consequences of the late-stage 2D scaling roadmap and recent scaling boosters. For example, density scaling with low-track height cell architectures while preserving PPA shines a spotlight on puzzle-fitting and "island" constraints (multi-height cells, multi-row heights, voltage domains, well structures, and sub-metal proximity). And, when 2D scaling is no longer optimal for the product, this induces 2.5D, 3D and 4D (reconfiguration) placement contexts. Finally, and perhaps most crucially: "New" topics can return to foreground when we decide to require automation with (super)human-quality solutions. Examples of this include analog placement and macro placement.

4.1 Physical Context

Several near-term foci arise from the physical context of placement. Two examples are true 3D placement and "island-heavy" placement. (Also: die-package power and floorplan co-optimization.)

3D Placement. Heterogeneous More-Than-Moore integration, as well as the monolithic 3D-VLSI end of More-Moore (1.0nm, 0.7nm "equivalent" nodes), bring multiple 3D placement challenges. Existing approaches typically apply partitioning into tiers, 2D placement per tier, and some interleaving of incremental improvement. Opportunities for cross-tier optimization (placement of inter-tier connections, power delivery, clocking, performance and power/thermal) are lost early in these flows. Yet, our intuition is that the z-axis should fundamentally change traditional co-optimizations due to severe gradients (temperature, IR, device quality) and costs (inter-tier connection). ***[Challenge 17:** Develop "true 3D" placement.]* True 3D placement would directly consider all objectives and constraints (timing, power, routability, etc.) in the full tiered 3D placement resource, throughout global placement, detailed placement, legalization and optimization.

Island-Heavy Placement. Low-track height libraries are used for density scaling even as multi-bit flip-flops [58] and high-drive, low output-resistance cells are needed to deliver high performance with minimum power. This results in standard-cell placement with a significantly higher proportion of multi-height cells (e.g., 1-, 2- and 3-row cell heights). In addition, rows with three or more distinct heights (cf. [35]) may be laid down in alternation or in other patterns. The different-height rows are analogous to voltage islands and conventional- vs. flip-well regions in FDSOI [37]. That is to say, each row has a specific spatial extent and site type, and determines the performance model of every cell placed in it. ***[Challenge 18:** Develop techniques for "island-heavy" placement.]* Mixed row-height and multi-height cell "puzzle-fitting" constraints must be introduced and enforced – at the right junctures – during the placement flow.[11] Ubiquity of multiple row heights and other "island" types opens new placement-floorplan co-optimizations, and adds new complexities to placement-CTS co-optimization.

4.2 Design Context

Other near-term foci arise from the design context. Two examples are macro placement for designs with extreme memory-dominance, and system interconnect-savvy SOC floorplan assessment.

[8] Tuning the evolution of objectives, the softness or relaxation of constraints, and overall "flow control" in the placer are all at the nexus of learning and placement, as noted in Section 5.

[9] This conjoint optimization should address clock sink clustering into multi-bit flip-flops, bottom-up sink clustering that accurately comprehends CRPR impacts, mitigation of hold padding, useful skew – as well as low-power design and variation-robustness. There are many chicken-egg loops here, e.g., [19].

[10] In 2012 and 2015, Markov et al. [78] [79] called out as open challenges the scaling of flat placement (for physical synthesis, more macros and more clocks), 3D placement, and five main topics: (i) datapath layout, (ii) layout-friendly high-level synthesis, (iii) integrated timing and power optimization, (iv) lithography-aware physical synthesis, and (v) "quantifying the impact". These are all still open. (i) is revisited here. (ii) is still improving accuracy and scalability of constructive and learning-based predictions to inform (ESL, architecture) design space exploration (cf. Sec. V of [55]). (iii) and (iv) are types of co-optimizations (Section 3 above), with (iv) arguably having turned out to be a non-issue. Progress on (v) may be along axes set out in Sections 2.2 and 6.

[11] Before puzzle-fitting can begin, different cell types must be distributed with even density. Displacement must be well-controlled in detailed placement.

Extreme Memory-Dominance. AI and machine learning accelerator architectures can have very high fractions of die or block area occupied by SRAM and register file instances. When hundreds of macros occupy 90% or more of area in performance-critical blocks, only human experts can find viable placements plus collaterals such as clock and power distribution strategies, density screens and IO placements. *[**Challenge 19:** Develop performance-driven macro packing and floorplan automation that matches or surpasses human solution quality.]* In this context, use of deep RL to achieve "placements that are superhuman or comparable" [81] is a huge breakthrough, but this capability is not yet broadly available.

System Interconnect-Savviness. System-, architecture-, and SOC-level design space exploration all require fast and accurate assessment of achievable PPAC envelopes; this is a function of placement (partitioning, shaping and packing). Such assessment is increasingly dominated by system interconnects such as standard buses and NOCs, which challenges even human experts. *[**Challenge 20:** Develop system interconnect-savvy SOC floorplan optimizations with expert-human solution quality.]* For example, timing- and global interconnect-aware shaping and packing must comprehend the number of pipeline stages needed for long-distance communication [80], in addition to clock periods and routability.

4.3 Specializations

Additional foci arise in "domain-specific" specializations, e.g., *datapath* placement has growing impact in AI, communications and signal processing architectures. Sectors such as aerospace (rad-hard) or automotive (harsh) bring extreme requirements that challenge placement. For instance, reliability physics of nanoscale devices are acknowledged by architectural features such as multi-modular redundancy. This requires placers to optimize physical separation of replicated or spare copies of key sub-blocks (in so-called master-checker configurations). Other "new" foci include areas where human expertise and productivity are long-standing bottlenecks: analog placement [36] [69] [100], IO placement, and die-package planning. *[**Challenge 21:** Develop niche placement automations that remove human expertise bottlenecks.]* Many research opportunities also exist at the blurred border between placement and partitioning (FPGAs, multi-FPGA emulation, chiplet-based integration, etc.).

Datapath Placement. Regular structures in RTL and netlist can be identified and used to guide structured placement of datapaths during global placement [102] [86] [6]. Commercial efforts of 25+ years ago (Cadence SmartPath, Synopsys ModuleCompiler, Arcadia Mustang, etc.) never sought to supplant the human designer. Rather, these tools put the burden on the designer to manage and manually place datapaths within the overall P&R block, while allowing for global parameters such as feedthroughs. As global placer quality improved, the need for specialized datapath placement decreased.

With today's tiled architectures, loss or gain in datapath placement solution quality is replicated hundreds of times. And, manual design steps scale poorly. *[**Challenge 22:** Develop datapath placement automation that surpasses human-directed solution quality.]* As timing-critical paths are repeated across multiple bit slices, there is opportunity for speedup by abstraction and bundling of these paths and their components.

5 LEARNING AND PLACEMENT

The nexus of machine learning (ML) and physical design has seen tremendous activity in recent years. The first IEEE CEDA DAWN seminar [113], IEEE CASS webinars, the MLCAD workshop [114], the ICCAD20 keynote of [89], and the excellent survey of Huang et al. [49] together give a solid picture of this fast-moving area. Commercial EDA has also taken up the challenge of scaling design quality and schedule with ML. Following are several directions that are expected to advance rapidly.

Learning objectives. ML models can provide improved optimization objectives (cf. Challenge 11) and provide insight into the "What should the placer know, and when should the placer know it?" question noted in Section 3. ML can also characterize correlations between different objectives (HPWL, density, TNS etc.) and apply such correlations in prediction and optimization.

Smart flow and flow control. Many commands and options are available to users of commercial placers. Placement heuristics have tunable parameters such as weight schedules, step size or number of passes. And, the placer-internal flow can initiate and terminate various hidden features at different junctures of a given tool run. In this context, few-shot or one-shot learning can enable design-specific "smart flows", ranging from recipe (runscript) recommendations to hidden control of numerical solvers and placer-internal flow. The concept of "smart flow" also encompasses the "target sequence" challenge in [52]: determine PPA and other constraints that will steer a multi-step optimization to a best-possible outcome.

Embedding and clustering. Embedding and clustering provide rich links between ML and placement. Statistical learning of low-dimensional representations can cast placement into already-tractable forms, and be reused across search spaces. Vector node embeddings and associated distance measures can be applied in placement optimizations (revisiting IO placement through this lens may be fruitful). Clustering criteria from ML such as modularity have been successfully adapted for problem size reduction and "blob placement".

Deep learning for placement. High-quality combinatorial optimization via deep learning is an active research area [9]. Today, there are open questions around (i) methods and resources needed for model training, and (ii) reproducible demonstrations of solution quality, generalization and transferability, and scalability. On the other hand, we have seen breakthrough deep-RL macro placement [81] and generally more rapid advances toward "Stage 4: Reinforcement Learning, Intelligence" [52] than originally foreseen.

Domain expertise. The complexity and hard-won capability of today's placers suggest leveraging rather than rediscovering domain expertise. This highlights methods such as imitation learning, algorithm alignment and transfer learning. On the other hand, the "Bitter Lesson" [94] might be kept in mind as ML is fused with traditional placement approaches.

AI hardware. Exploitation of AI hardware has turned a corner in global and detailed placement (DREAMPlace [73], ABCDPlace [74]). These methods make high-quality constructive placement prediction feasible in iterative floorplan and PDN optimization. Logical next steps include combining with RL-based macro placement [81], and developing RL-based standard-cell placement.

Open source. Machine learning for placement-centric optimizations may benefit from at-scale use of open-source platforms such as [106]. This motivates: *[**Challenge 23:** Develop accurate predictors of closed-source tool outcomes using open-source tools.]* Interestingly, co-evolution and convergence of closed- and open-source engines could result: (i) open source would seek to reduce deltas from closed source, and (ii) closed source would seek to be more predictable.

More. *[**Challenge 24:** Find quantum leaps in solution quality and speed at the nexus of ML and placement.]* Along the path to "direct inference" or "end-to-end", various practical realities and small

steps should be worked out. Deployable ML would need to provide robustness, explainability and model debuggability when results are poor. [89] points out real-world aspects of AI automation such as few-shot learning, data augmentation and 3-layer model architecture that help deal with small data and/or user-confidential data (cf. Sections 2 and 6). In addition, "target lists" will evolve and require curation. Today, routability hotspot prediction is being commoditized, but placement that can predict and preemptively mitigate SI or IR hotspots is an open challenge (cf. target lists in [52] and Section 2). Advancing performance predictions from GBA-accuracy to {PBA,SI,MCMM}-accuracy is still the missing key to better correlation between timing- and power-driven placement optimizations and post-route outcomes.

6 SUPPORT FOR PLACEMENT RESEARCH

Any research community continually seeks to accelerate progress, attract new talent, and achieve real-world impact. Given the breadth and difficulty of the above challenges, this need is now more critical than ever before. Directions to keep an eye on include flow-scale contexts for optimization; open-source research enablement; and metrics collection to support machine learning.

6.1 Flow-scale Context and Open Source

The ISPD Contests (and the earlier ISPD-98 partitioning benchmark suite) have been key drivers for the entire physical design field. Half of the contests from ISPD-2005 to ISPD-2020 have been on placement, reflecting the central and evergreen nature of placement challenges. Overall, 14 placement contests have been run at major conferences (Table 1).

Table 1: History of academic placement contests.

Year	Title
ISPD05	Placement
ISPD06	Placement
ISPD11	Routability-Driven Placement
DAC12	Routability-Driven Placement
ICCAD12	Design Hierarchy Aware Routability-Driven Placement
ICCAD13	Detailed Placement
ISPD14	Detailed Routing-Driven Placement
ICCAD14	Incremental Timing-Driven Placement
ISPD15	Blockage-Aware Detailed Routing-Driven Placement
ICCAD15	Incremental Timing-Driven Placement
ISPD16	Routability-Driven FPGA Placement
ISPD17	Clock-Aware FPGA Placement
ICCAD17	Multi-Deck Standard Cell Legalization
ISPD20	Wafer-Scale Deep Learning Accelerator Placement

Many significant methods and publications have directly resulted from placement contest entries. However, contests in general do not span multiple flow steps, even as the placement context continues to shift toward co-optimization and co-operation. Furthermore, contest enablements often have proprietary (and/or "disabled") elements [23], and are thus difficult to propagate or extend. As a result, progress and relevance have been hampered by a lack of platforms for integration, interoperability, and in-context evaluation of academic research. Past efforts to overcome these challenges include the MARCO/GSRC Bookshelf [16], OpenAccess Gear [99], and A2A "horizontal benchmark extension" [57].[12]

[12]These works have all shared the same basic motivations. E.g., the abstract of [99] begins: "Physical design EDA research in academia has historically been based on infrastructure developed independently by individual contributors. This has led to fragmentation in the community, where interaction, data interchange and comparison of results between tools are difficult. We discuss our early experience with the OpenAccess Gear system, an open source software initiative intended to provide pieces of the critical integration and analysis infrastructure that are taken for granted in proprietary tools, but often wholly absent in research tools."

A significant ongoing initiative, dating from ICCAD-2016, is the IEEE CEDA DATC *Robust Design Flow* (RDF). As stated in [23], the RDF aims to *"... facilitate research on flow-scale methodology and cross-stage optimizations"* and *"seeks to (i) provide an academic reference flow from logic synthesis to detailed routing based on existing contest results; (ii) construct a database for design benchmarks and point tool libraries; and (iii) connect academic research to industry practitioners and designs by using industry-standard design input/output formats"*. As of its most recent iteration [24] [104], the RDF spans all of RTL-to-GDS and includes multiple options at several stages, notably global placement. In 2020, the RDF also added open *analysis calibrations*.

Another initiative, dating from mid-2018, is the DARPA IDEA program [111], which launched multiple *open-source* EDA projects. For digital RTL-to-GDS automation, OpenROAD [106] integrates 20+ engines with the OpenDB physical implementation database[13] and the OpenSTA timer. It has been part of the IEEE CEDA RDF initiative since 2019. Figure 4 shows the tool's incremental shared netlist architecture, which enables in-memory communication between tools and tight incremental optimization loops. This provides an available backplane for research on flow-level optimizations, and for assessment of algorithm and engine improvements, in an industry-compatible, open-source context. The past year has also seen the open-sourcing of a *manufacturable* PDK and libraries (130nm, SkyWater Technology Foundry [108]), along with the ASAP7 advanced-node research PDK and 7.5T cell library [28] [109]. Together, these developments remove many roadblocks for research noted in [79], and facilitate more rapid transfer of academic research into real-world practice. How the trajectory of research will change as a result remains to be seen.

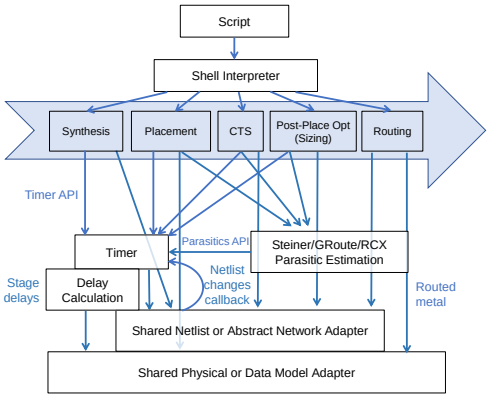

Figure 4: OpenROAD shared netlist architecture.

6.2 Metrics Collection and ML Enablement

Standards and infrastructure are needed to support data collection and data-enabled machine learning [52] [38] [59]. On the one hand, tools must produce harvestable metrics using standard terms and semantics. In the placement context, such metrics include TNS, WNS, wirelength, overcongestion, etc. – sometimes referred to as

[13]OpenDB's underlying data model is similar to that of LEF/DEF and OpenAccess [110].

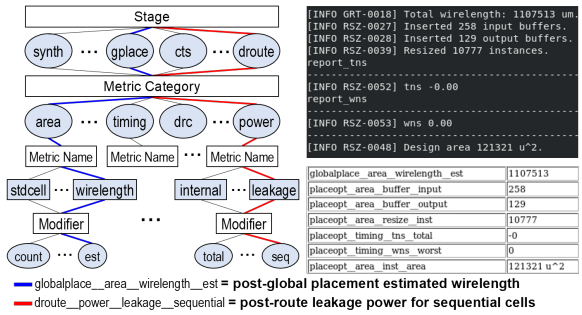

Figure 5: OpenROAD metrics naming and extraction from logging.

"derived data" since it is derived from the state of the physical design. Furthermore, data generation for machine learning requires a data model, an implementation, and an API to extract derived data [112]. OpenROAD with its OpenDB database provides a working open-source database, data model and implementation. All tools use a common spdlog-based messaging package, with consistent message types and tool namespaces. Figure 5 (left) illustrates metrics naming in the current OpenROAD, based on universes of *flow stage* names, *nouns* at run-level and tool metric-level, and *modifiers*. The figure shows how run metrics or tool metrics would have canonical names. For example, the blue path shows the name for post-global placement estimated wirelength. (The metrics can be tied to corners, runs, designs, etc.) In Figure 5 (right), logging in the tool is shown above, and metrics extracted and recorded using Python and JSON are shown below. Using such infrastructure, the research community can collaboratively amass data for machine learning, draw interest via Kaggle contests, etc. ***[Challenge 25:** Launch organized, multi-entity metrics collection to support ML studies and contests.]* The related challenge is to get the ball rolling with first results and first learnings.

7 CONCLUSIONS

Looking back, three personal observations re [79] are the following. First, there are new highlights: (i) deep reinforcement learning for macro placement [81]; (ii) game-changing exploitation of AI hardware [73] [74]; (iii) the confirmed influence of the 2005 ISPD Contest [83] and its successors, with ripple effects seen in Table 1, the DATC RDF, and the hotbeds of activity at NTU, Iowa State, CUHK, Michigan, UCLA and elsewhere; and (iv) the advent of an open-source, industry-compatible, full-flow platform for placement research. Second, there are directions that should be advancing faster: (v) floorplacement [90] [85], multilevel [17], distributable heavy optimization (Challenge 13), and artificial testcase generation (Challenge 8) paradigms; and (vi) clearing the haze over timing, routability, incremental/reentrant usage, and stability in placement optimization (Challenges 4-6, 11-12). Third, more notes from "the sweep of history" seem apparent: (vii) the longevity of sequence pair [82] and O-tree/B*-tree [40] [25] based annealing [98] [1] in floorplanning; (viii) the place of early antecedents such as regularization [4] [7] [65] as backdrop to [84], or [48] as backdrop to [96]; and (ix) the impact of multi-generation efforts that produced mPL6 [17], FastPlace3.0 [97] and IPR [88], NTUplace4h [47], late versions of Capo [91] [92], and ePlace-MS/RePlAce [77] [26]. A personal reading list is at [115].

Looking forward, placement research will need to match itself to what CAD optimization in practice will need to look like in the future: more intelligent and autonomous; deployed on distributed and cloud resources; and optimizing expectations and Paretos. This means serious investment in research on learning-enabled optimization (i.e., at a new nexus of ML and optimization), distributed and federated methods, the interplay between discrete-combinatorial and continuous methods, and related directions. Last but not least, culture changes (open sourcing, benchmarking, and focus on narrowing of quantified suboptimality gaps) are also needed. As seen in AI/ML fields, advancing optimization comes with benchmarks, measured progress to reduce suboptimality, and tight translation paths between research and leading-edge practice -– all of which are also part of the next chapter for research in VLSI placement and physical design.

8 ACKNOWLEDGMENTS

Many thanks are due to colleagues who generously provided thoughts on futures for placement which have been incorporated here: David Chinnery, Dennis Huang, Ilgweon Kang, Stefanus Mantik, Igor Markov, Siddhartha Nath, David Pan, Ravi Varadarajan and Joe Shinnerl. Thanks are also due to ISPD-2021 organizers for inviting this paper and talk.

REFERENCES

[1] S. N. Adya and I. L. Markov, "Fixed-outline Floorplanning Through Better Local Search", *Proc. ICCD*, 2001, pp. 328-333.
[2] T. Ajayi, D. Blaauw, T.-B. Chan, C.-K. Cheng et al., "OpenROAD: Toward a Self-Driving, Open-Source Digital Layout Implementation Tool Chain", *Proc. GOMACTech*, 2019, pp. 1105-1110.
[3] D. Aldous and U. Vazirani, ""Go with the Winners" Algorithms", *Proc. FOCS*, 1994, pp. 492–501.
[4] C. J. Alpert, T. Chan, D. J. Huang, A. B. Kahng, I. Markov, P. Mulet and K. Yan, "Faster Minimization of Linear Wirelength for Global Placement", *Proc. ISPD*, 1997, pp.4-11.
[5] C. J. Alpert, S. K. Karandikar, Z. Li, G.-J. Nam, S. T. Quay, H. Ren, C. N. Sze, P. G. Villarrubia and M. C. Yildiz, "Techniques for Fast Physical Synthesis", *Proc. IEEE* 95(3) (2007), pp. 573-599.
[6] S. Arikati and R. Varadarajan, "A Signature Based Approach to Regularity Extraction," *Proc. ICCAD*, 1997, pp. 542-545.
[7] R. Baldick, A. B. Kahng, A. Kennings, and I. L. Markov, "Function smoothing with applications to VLSI layout", *Proc. ASP-DAC*, 1999, pp. 225-228.
[8] R. S. Barr, B. L. Golden, J. P. Kelly, M. G. C. Resende and W. R. Stewart, "Designing and Reporting on Computational Experiments with Heuristic Methods", *J. Heuristics* 1 (1995), pp. 9-32.
[9] Y. Bengio, A. Lodi and A. Prouvost, "Machine Learning for Combinatorial Optimization: a Methodological Tour d'Horizon", *Eur. J. Oper. Res.* 290(2) (2021), pp. 405-421.
[10] S. Bianco, R. Cadene, L. Celona and P. Napoletano, "Benchmark Analysis of Representative Deep Neural Network Architectures", *IEEE Access*, Oct. 2018, DOI: 10.1109/ACCESS.2018.2877890.
[11] R. Brashears and A. B. Kahng, "Advanced Routing for Deep Submicron Technologies", *Computer Design*, May 1997. Available at http://vlsicad.ucsd.edu/Presentations/talk/supporting.html
[12] K. D. Boese, A. B. Kahng and S. Muddu, "A New Adaptive Multistart Technique for Combinatorial Global Optimizations", *OR Letts.* 16(2) (1994), pp. 101-113.
[13] F. Brglez, "Design of Experiments to Evaluate CAD Algorithms: Which Improvements Are Due to Improved Heuristic and Which are Merely Due to Chance?", *technical report* CBL-04-Brglez, NCSU Collaborative Benchmarking Laboratory, April 1998.
[14] A. E. Caldwell, A. B. Kahng, A. A. Kennings and I. L. Markov, "Hypergraph Partitioning for VLSI CAD: Methodology for Heuristic Development, Experimentation and Reporting", *Proc. DAC*, 1999, pp. 349-354.
[15] A. E. Caldwell, A. B. Kahng and I. L. Markov, "Optimal Partitioners and End-case Placers for Standard-cell Layout", *IEEE TCAD* 19(11) (2000), pp, 1304-1313.
[16] A. E. Caldwell, A. B. Kahng and I. L. Markov, "Toward CAD-IP Reuse: The MARCO GSRC Bookshelf of Fundamental CAD Algorithms", *IEEE Design and Test of Computers*, 2002, pp. 70-79. http://vlsicad.eecs.umich.edu/BK/
[17] T. Chan, J. Cong, J. Shinnerl, K. Sze and M. Xie, "mPL6: Enhanced Multilevel Mixed-size Placement", *Proc. ISPD*, 2006, pp. 212-214.
[18] T.-B. Chan, A. B. Kahng and M. Woo, "Revisiting Inherent Noise Floors for Interconnect Prediction", *Proc. SLIP*, 2020.
[19] T.-B. Chan, A. B. Kahng and J. Li, "NOLO: A No-Loop, Predictive Useful Skew Methodology for Improved Timing in IC Implementation", *Proc. ISQED*, 2014, pp. 504-509.
[20] W.-T. J. Chan, Y. Du, A. B. Kahng, S. Nath and K. Samadi, "BEOL Stack-Aware Routability Prediction from Placement Using Data Mining Techniques", *Proc. ICCD*, 2016, pp. 41-48.
[21] W.-T. J. Chan, P.-H. Ho, A. B. Kahng and P. Saxena, "Routability Optimization for Industrial Designs at Sub-14nm Process Nodes Using Machine Learning", *Proc. ISPD*, 2017, pp. 15-21.
[22] C.-C. Chang, J. Cong and M. Xie, "Optimality and scalability study of existing placement algorithms", *Proc. ASP-DAC*, 2003, pp. 612-627.
[23] J. Chen, I. H.-R. Jiang, J. Jung, A. B. Kahng, V. N. Kravets, Y.-L. Li, S.-T. Lin and M. Woo, "DATC RDF-2019: Towards a Complete Academic Reference Design Flow", *Proc. ICCAD*, 2019, pp. 1-6.
[24] J. Chen, I. H.-R. Jiang, J. Jung, A. B. Kahng, V. N. Kravets, Y.-L. Li, S.-T. Lin and M. Woo, "DATC RDF-2020: Strengthening the Foundation for Academic Research in IC Physical Design", *Proc. ICCAD*, 2020, pp. 1-6.

[25] T.-C. Chen and Y.-W. Chang, "Modern Floorplanning Based on B*-tree and Fast Simulated Annealing", *IEEE TCAD* 25(4) (2006), pp. 637-650.
[26] C. Cheng, A. B. Kahng, I. Kang and L. Wang, "RePlAce: Advancing Solution Quality and Routability Validation in Global Placement", *IEEE TCAD* 38(9) (2019), pp. 1717-1730.
[27] W.-K. Chow, J. Kuang, X. He, W. Cai and E. F. Y. Young, "Cell Density-driven Detailed Placement with Displacement Constraint", *Proc. ISPD*, 2014, pp. 3-10.
[28] L. T. Clark, V. Vashishtha, L. Shifren, A. Gujja, S. Sinha et al., "ASAP7: A 7-nm FinFET Predictive Process Design Kit", *Microelectronics J.* 53 (2016), pp. 105–115.
[29] J. Cong, M. Romesis and M. Xie, "Optimality, Scalability and Stability Study of Partitioning and Placement Algorithms", *Proc. ISPD*, 2003, pp. 88-94.
[30] O. Coudert, J. Cong, S. Malik and M. Sarrafzadeh, "Incremental CAD", *Proc. ICCAD*, 2000, pp. 236-244.
[31] N. Dakwala, "Data Mining STA Reports", *Proc. SNUG Silicon Valley*, 2016, pp. 1-13.
[32] J. Darnauer and W. Dai, "A Method for Generating Random Circuits and Its Application to Routability Measurement", *ACM Int'l Symp. on FPGAs*, 1996, pp. 66-72.
[33] P. Debacker, K. Han, A. B. Kahng, H. Lee, P. Raghavan and L. Wang, "Vertical M1 Routing-Aware Detailed Placement for Congestion and Wirelength Reduction in Sub-10nm Nodes", *Proc. DAC*, 2017, pp. 1-6.
[34] K. Deb, A. Pratap, S. Agarwal and T. Meyarivan, "A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II", *IEEE Trans. Evol. Comp.* 6(2) (2002), pp. 182-197.
[35] S. Dobre, A. B. Kahng and J. Li, "Design Implementation with Non-Integer Multiple-Height Cells for Improved Design Quality in Advanced Nodes", *IEEE TCAD* 37(4) (2018), pp. 855-868.
[36] R. Dreslinski, D. Wentzloff, M. Fayazi, K. Kwon, D. Blaauw, D. Sylvester, B. Calhoun, M. Coltella and D. Urquhart, "Fully-Autonomous SoC Synthesis using Customizable Cell-Based Synthesizable Analog Circuits", 2019, https://apps.dtic.mil/sti/citations/AD1075802 https://github.com/idea-fasoc/fasoc
[37] H. Fatemi, A. B. Kahng, H. Lee and J. Pineda de Gyvez, "Heuristic Methods for Fine-Grain Exploitation of FDSOI", *IEEE TCAD* 39(10) (2020), pp. 2860-2871.
[38] S. Fenstermaker, D. George, A. B. Kahng, S. Mantik and B. Thielges, "METRICS: A System Architecture for Design Process Optimization", *Proc. DAC*, 2000, pp. 705-710.
[39] D. Grant and G. Lemieux, "Perturb+Mutate: Semisynthetic Circuit Generation for Incremental Placement and Routing", *ACM TRETS* 1(3) (2008), pp. 1-24.
[40] P.-N. Guo, C.-K. Cheng and T. Yoshimura, "An O-tree Representation of Non-slicing Floorplan and its Applications", *Proc. DAC*, 1999, pp. 268-273.
[41] L. Hagen, J. H. Huang and A. B. Kahng, "Quantified Suboptimality of VLSI Layout Heuristics", *Proc. DAC*, 1995, pp. 216-221.
[42] C. Han, A. B. Kahng, L. Wang and B. Xu, "Enhanced Optimal Multi-Row Detailed Placement for Neighbor Diffusion Effect Mitigation in Sub-10nm VLSI", *IEEE TCAD* 38(9) (2019), pp. 1703-1716.
[43] K. Han, A. B. Kahng and H. Lee, "Scalable Detailed Placement Legalization for Complex Sub-14nm Constraints", *Proc. ICCAD*, 2015, pp. 867-873.
[44] S. S. Han, A. B. Kahng, S. Nath and A. Vydyanathan, "A Deep Learning Methodology to Proliferate Golden Signoff Timing", *Proc. DATE*, 2014, pp. 1-6.
[45] S. Heo, A. B. Kahng, M. Kim and L. Wang, "Diffusion Break-Aware Leakage Power Optimization and Detailed Placement in Sub-10nm VLSI", *Proc. ASP-DAC*, 2019, pp. 550-556.
[46] S. Heo, A. B. Kahng, M. Kim, L. Wang and C. Yang, "Detailed Placement for IR Drop Mitigation by Power Staple Insertion in Sub-10nm VLSI", *Proc. DATE*, 2019, pp. 824-829.
[47] M. Hsu, Y. Chen, C. Huang, S. Chou, T. Lin, T. Chen and Y. Chang, "NTUplace4h: A Novel Routability-Driven Placement Algorithm for Hierarchical Mixed-Size Circuit Designs", *IEEE TCAD* 33(12) (2014), pp. 1914-1927.
[48] B. Hu and M. Marek-Sadowska, "FAR: Fixed-points Addition and Relaxation Based Placement", *Proc. ISPD*, 2002, pp.161–166.
[49] G. Huang, J. Hu, Y. He, J. Liu, M. Ma, Z, Shen, J. Wu, Y. Xu, H. Zhang, K. Zhong, X. Ning, Y. Ma, H. Yang, B. Yu, H. Yang and Y. Wang, "Machine Learning for Electronic Design Automation: A Survey", *ACM TODAES* (2021), pp. 1-44.
[50] M. D. Hutton, J. Rose, and D. Corneil, "Automatic Generation of Synthetic Sequential Benchmark Circuits", *IEEE TCAD* 21(8) (2002), pp. 928-940.
[51] K. Jeong and A. B. Kahng, "Methodology From Chaos in IC Implementation", *Proc. ISQED*, 2010, pp. 885-892.
[52] A. B. Kahng, "Machine Learning Applications in Physical Design: Recent Results and Directions", *Proc. ISPD*, 2018, pp. 68-73.
[53] A. B. Kahng, "Reducing Time and Effort in IC Implementation: A Roadmap of Challenges and Solutions", *Proc. DAC*, 2018, pp. 1-6.
[54] A. B. Kahng, "Looking Into the Mirror of Open Source", *Proc. ICCAD*, 2019, pp. 1-8.
[55] A. B. Kahng, "Open-Source EDA: If We Build It, Who Will Come?", *Proc. VLSI-SoC*, 2020, pp. 1-6.
[56] A. B. Kahng, "MLCAD Today and Tomorrow: Learning, Optimization and Scaling", *MLCAD Workshop*, Nov. 2020. https://www.youtube.com/watch?v=o__VF3yUhyc
[57] A. B. Kahng, H. Lee and J. Li, "Horizontal Benchmark Extension for Improved Assessment of Physical CAD Research", *Proc. GLSVLSI*, 2014, pp. 27-32.
[58] A. B. Kahng, J. Li and L. Wang, "Improved Flop Tray-Based Design Implementation for Power Reduction", *Proc. ICCAD*, 2016, pp. 1-8.
[59] A. B. Kahng and S. Mantik, "A System for Automatic Recording and Prediction of Design Quality Metrics", *Proc. ISQED*, 2001, pp. 81-86.
[60] A. B. Kahng and S. Mantik, "On Mismatches Between Incremental Optimizers and Instance Perturbations in Physical Design Tools", *Proc. ICCAD*, 2000, pp. 17-21.
[61] A. Kahng and S. Mantik, "Measurement of Inherent Noise in EDA Tools", *Proc. ISQED*, 2002, pp. 206-211.
[62] A. B. Kahng and S. Reda, "Evaluation of Placer Suboptimality Via Zero-Change Netlist Transformations", *Proc. ISPD*, 2005, pp. 208-215.
[63] A. B. Kahng and Q. Wang, "Implementation and Extensibility of an Analytic Placer", *IEEE TCAD* 5(24) (2005), pp. 734-747.
[64] J. Kennedy and R. Eberhart, "Particle Swarm Optimization", *Proc. ICNN* Vol. 4, 1995, pp. 1942-1948.
[65] A. A. Kennings and I. L. Markov, "Analytical Minimization of Halfperimeter Wirelength", *Proc. ASP-DAC*, 2000, pp. 179-184.
[66] M. Kim, J. Hu, J. Li and N. Viswanathan, "ICCAD-2015 CAD Contest in Incremental Timing-driven Placement and Benchmark Suite", *Proc. ICCAD*, 2015, pp. 921-926.
[67] D. Kim, H. Kwon, S.-Y. Lee, S. Kim, M. Woo and S. Kang, "Machine Learning Framework for Early Routability Prediction with Artificial Netlist Generator", *Proc. DATE*, 2021, to appear.
[68] M.-C. Kim, N. Viswanathan, C. J. Alpert, I. L. Markov and S. Ramji, "MAPLE: Multilevel Adaptive Placement for Mixed-Size Designs", *Proc. ISPD*, 2012, pp. 193–200.
[69] K. Kunal, M. Madhusudan, A. K. Sharma, W. Xu, S. M. Burns, R. Harjani, J. Hu, D. A. Kirkpatrik and S. Sapatnekar, "ALIGN: Open-Source Analog Layout Automation from the Ground Up", *Proc. DAC*, 2019, pp. 1-4, https://github.com/ALIGN-analoglayout/ALIGN-public.
[70] P. D. Kundarewich and J. Rose, "Synthetic Circuit Generation Using Clustering and Iteration", *IEEE TCAD*, 2004, pp. 869-887.
[71] T. Lengauer, *Combinatorial Algorithms for Integrated Circuit Layout*, Wiley-Teubner, 1990.
[72] T. Lin, C. Chu, J. R. Shinnerl, I. Bustany and I. Nedelchev, "POLAR: Placement Based on Novel Rough Legalization and Refinement", *Proc. ICCAD*, 2013, pp. 357-362.
[73] Y. Lin, S. Dhar, W. Li, H. Ren, B. Khailany and D. Z. Pan, "DREAMPlace: Deep Learning Toolkit-Enabled GPU Acceleration for Modern VLSI Placement", *Proc. DAC*, 2019, pp. 1-6.
[74] Y. Lin, W. Li, J. Gu, H. Ren, B. Khailany and D. Z. Pan, "ABCDPlace: Accelerated Batch-Based Concurrent Detailed Placement on Multithreaded CPUs and GPUs", *IEEE TCAD* 39(12) (2020), pp.5083-5096.
[75] Y. Lin, B. Yu, X. Xu and J. Gao, N. Viswanathan, W. Liu, Z. Li, C. J. Alpert and D. Z. Pan, "MrDP: Multiple-row Detailed Placement of Heterogeneous-sized Cells for Advanced Nodes", *Proc. ICCAD*, 2016, pp. 1-8.
[76] J. Lu, P. Chen, C.-C. Chang, L. Sha, D. J.-H. Huang, C.-C. Teng and C.-K. Cheng, "ePlace: Electrostatics Based Placement Using Fast Fourier Transform and Nesterov's Method", *ACM TODAES* 20(2) (2015), article 17.
[77] J. Lu, H. Zhuang, P. Chen, H. Chang, C. Chang, Y. Wong, L.Sha, D. Huang, Y. Luo, C. Teng and C. Cheng, "ePlace-MS: Electrostatics-Based Placement for Mixed-Size Circuits", *IEEE TCAD* 34(5) (2015), pp.685-698.
[78] I. L. Markov, J. Hu and M. Kim, "Progress and Challenges in VLSI Placement Research", *Proc. ICCAD*, 2012, pp. 275-282.
[79] I. L. Markov, J. Hu and M. Kim, "Progress and Challenges in VLSI Placement Research", *Proc. of the IEEE*, 103(11) (2015), pp. 1985-2003.
[80] D. Matzke, "Will Physical Scalability Sabotage Performance Gains?", *IEEE Computer* 30(9) (1997), pp. 37-39.
[81] A. Mirhoseini, A. Goldie, M. Yazgan, J. Jiang, E. Songhori, et al., "Chip Placement with Deep Reinforcement Learning", *arXiv* 2004.10746, 2020. https://arxiv.org/abs/2004.10746
[82] H. Murata, K. Fujiyoshi, S. Nakatake and Y. Kajitani, "VLSI Module Placement Based on Rectangle-packing by the Sequence-pair", *IEEE TCAD* 15(12) (1996), pp. 1518-1524.
[83] G.-J. Nam, C. J. Alpert, P. Villarrubia, B. Winter and M. Yildiz, "The ISPD 2005 Placement Contest and Benchmark Suite", *Proc. ISPD*, 2005, pp. 216-220.
[84] W. Naylor, R. Donelly, and L. Sha, "Non-Linear Optimization System and Method for Wire Length and Delay Optimization for an Automatic Electric Circuit Placer", *U.S. Patent 6301693*, Oct. 2001.
[85] A. N. Ng, I. L. Markov, R. Aggarwal and V. Ramachandran, "Solving Hard Instances of Floorplacement", *Proc. ISPD*, 2006, pp. 170-177.
[86] R. X. T. Nijssen and C. A. J. van Eijk, "Regular Layout Generation of Logically Optimized Datapaths", *Proc. ISPD*, 1997, pp. 42-47.
[87] H. Onodera, Y. Taniguchi and K. Tamaru, "Branch-and-bound Placement for Building Block Layout" *Proc. DAC*, 1991, pp. 433-439.
[88] M. Pan and C. Chu, "IPR: An Integrated Placement and Routing Algorithm", *Proc. DAC*, 2007, pp. 59-62.
[89] R. Puri, "Engineering the Future of AI for Enterprises", *ICCAD keynote*, 2020.
[90] J. A. Roy, S. N. Adya, D. A. Papa and I. L. Markov, "Min-cut Floorplacement", *IEEE TCAD* 25(7) (2006), pp. 1313-1326.
[91] J. A. Roy, D. A. Papa, S. N. Adya, H. H. Chan, A. N. Ng, J. F. Lu and I. L. Markov, "Capo: Robust and Scalable Opensource Min-cut Floorplacer", *Proc. ISPD*, 2005, pp. 224-226.
[92] J. A. Roy, D. A. Papa and I. L. Markov, "Capo: Congestion-driven Placement for Standard-cell and RTL Netlists with Incremental Capability", *Modern Circuit Placement*, Springer, 2007.
[93] D. Stroobandt, P. Verplaetse and J. van Campenhout, "Generating Synthetic Benchmark Circuits for Evaluating CAD Tools", *IEEE TCAD* 19(9) (2000), pp. 1011-1022.
[94] R. Sutton, "The Bitter Lesson" blog post, March 13, 2019. http://www.incompleteideas.net/IncIdeas/BitterLesson.html
[95] N. Viswanathan, C. J. Alpert, C. Sze, Z. Li, G.-J. Nam, and J. A. Roy, "The ISPD-2011 Routability-Driven Placement Contest and Benchmark Suite", *Proc. ISPD*, 2011, pp. 141-146.
[96] N. Viswanathan and C. Chu, "FastPlace: Efficient Analytical Placement Using Cell Shifting, Iterative Local Refinement, and a Hybrid Net Model", *IEEE TCAD* 24(5) (2005), pp. 722-733.
[97] N. Viswanathan, M. Pan and C. Chu, "FastPlace 3.0: A Fast Multilevel Quadratic Placement Algorithm with Placement Congestion Control", *Proc. ASP-DAC*, 2007, pp.135-140.
[98] D. F. Wong and C. L. Liu, "A New Algorithm for Floorplan Design", *Proc. DAC*, 1986, pp.101-107.
[99] Z. Xiu, D. A. Papa, P. Chong, C. Albrecht, A. Kuehlmann, R. A. Rutenbar and I. L. Markov, "Early Research Experience With OpenAccess Gear: An Open Source Development Environment For Physical Design", *Proc. ISPD*, 2005, pp. 94-100.
[100] B. Xu, K. Zhu, M. Liu, Y. Lin, S. Li, X. Tang, N. Sun and D. Z. Pan, "Magical: Toward Fully Automated Analog IC Layout Leveraging Human and Machine Intelligence.", *Proc. ICCAD*, 2019, pp. 1-8. https://github.com/magical-eda/MAGICAL
[101] J. Z. Yan and C. Chu, "DeFer: Deferred Decision Making Enabled Fixed-outline Floorplanner", *Proc. DAC*, 2008, pp. 161-166.
[102] T. T. Ye and G. D. Micheli, "Data Path Placement with Regularity", *Proc. ICCAD*, 2000, pp. 264-270.
[103] S. Zilberstein, "Using Anytime Algorithms in Intelligent Systems", *AI Magazine*, Fall 1996, pp. 73-83.
[104] DATC-RDF, https://github.com/ieee-ceda-datc/datc-robust-design-flow.
[105] *International Technology Roadmap for Semiconductors* Design Chapter, 2009 edition. www.itrs2.net/itrs-reports.html and https://www.dropbox.com/sh/ia1jkem3v708hx1/AAB1fo1HrYIKClJNk0dB7YrCa?dl=0
[106] OpenROAD, https://github.com/The-OpenROAD-Project.
[107] RePlAce, https://github.com/The-OpenROAD-Project/RePlAce.
[108] SkyWater 130nm Open Source PDK, https://github.com/google/skywater-pdk.
[109] ASAP7 PDK and 7.5-Track Cell Library, https://github.com/The-OpenROAD-Project/asap7.
[110] OpenAccess, https://si2.org/openaccess/
[111] DARPA IDEA, https://www.darpa.mil/program/intelligent-design-of-electronic-assets
[112] Silicon Integration Initiative, Artificial Intelligence / Machine Learning Public Documents, https://si2.org/ai-ml-downloads/
[113] IEEE CEDA Design Automation WebiNar (DAWN) Event 1: Machine Learning For EDA. https://ieee-ceda.org/presentation/webinar/dawn-machine-learning-eda
[114] ACM/IEEE Workshop on Machine Learning for CAD (MLCAD). http://mlcad.itec.kit.edu/
[115] A Placement Reading List. https://bit.ly/3adMqUZ