electronics

Review

Progress of Placement Optimization for Accelerating VLSI Physical Design

Yihang Qiu [1,2], Yan Xing [1,*], Xin Zheng [1], Peng Gao [1], Shuting Cai [1] and Xiaoming Xiong [1]

[1] School of Integrated Circuits, Guangdong University of Technology, Guangzhou 510006, China
[2] School of Automation, Guangdong University of Technology, Guangzhou 510006, China
[*] Correspondence: yanxing@gdut.edu.cn

Abstract: Placement is essential in very large-scale integration (VLSI) physical design, as it directly affects the design cycle. Despite extensive prior research on placement, achieving fast and efficient placement remains challenging because of the increasing design complexity. In this paper, we comprehensively review the progress of placement optimization from the perspective of accelerating VLSI physical design. It can help researchers systematically understand the VLSI placement problem and the corresponding optimization means, thereby advancing modern placement optimization research. We highlight emerging trends in modern placement-centric VLSI physical design flows, including placement optimizers and learning-based predictors. We first define the placement problem and review the classical placement algorithms, then discuss the application bottleneck of the classical placement algorithms in advanced technology nodes and give corresponding solutions. After that, we introduce the development of placement optimizers, including algorithm improvements and computational acceleration, pointing out that these two aspects will jointly promote accelerating VLSI physical design. We also present research working on learning-based predictors from various angles. Finally, we discuss the common and individual challenges encountered by placement optimizers and learning-based predictors.

Keywords: VLSI placement; large-scale optimization; physical design; machine learning; prediction; placement challenges

Citation: Qiu, Y.; Xing, Y.; Zheng, X.; Gao, P.; Cai, S.; Xiong, X. Progress of Placement Optimization for Accelerating VLSI Physical Design. *Electronics* **2023**, *12*, 337. https://doi.org/10.3390/electronics12020337

Academic Editor: Antonio G. M. Strollo

Received: 11 December 2022
Revised: 5 January 2023
Accepted: 5 January 2023
Published: 9 January 2023

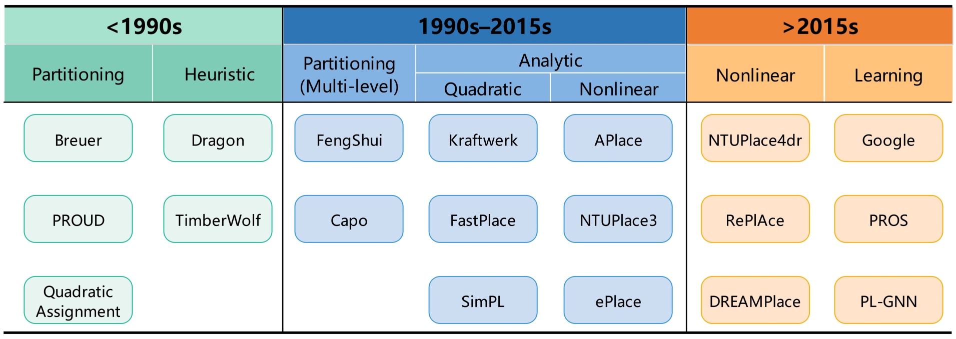
Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Moore's law [1] has long guided the development of the semiconductor industry as a rule of thumb. In the post-Moore era, however, the semiconductor industry faces severe challenges due to increasing design size and complex objectives and constraints. These challenges make the automated very-large-scale integration (VLSI) physical design (PD) difficult and even worse, leading to longer design cycles. At the beginning of VLSI PD, the circuit cells' location needs to be determined. There are two critical stages called macro (e.g., RAMs and ROMs) placement and standard cell (logic gates, such as NOR and NAND) placement.

Getting a good placement result is essential and challenging. Poor placement quality may cause routing failure and thus the need to re-iterate the design. Placement is a multi-objective optimization problem and also an NP-complete problem [2]. Due to its complexity, standard cell placement is often divided into global placement (GP), legalization (LG), and detailed placement (DP). GP determines the approximate position of each cell, and the result allows partial overlap. LG eliminates all the overlaps between cells and removes all design rule violations. DP makes further adjustments to improve the final placement quality. GP is more important than LG and DP because it determines the general location of the circuit cells. When discussing standard cell placement, we are concerned with GP.

Given the importance and difficulty of placement, it has been extensively studied over the past 50 years. As shown in Figure 1, various algorithms are used to solve the placement problem, including partitioning-based methods, heuristic algorithms, analytical solvers, and learning-based algorithms. Even with all these efforts, however, the quality and efficiency of placements still need to be fully satisfied, and further optimization is still needed. In addition, the shrinking feature size and increasing design scale bring growing complexity to the placement problem, making it a hot topic all the time.



<1990s		1990s–2015s			>2015s	
Partitioning	Heuristic	Partitioning (Multi-level)	Analytic		Nonlinear	Learning
			Quadratic	Nonlinear		
Breuer	Dragon	FengShui	Kraftwerk	APlace	NTUPlace4dr	Google
PROUD	TimberWolf	Capo	FastPlace	NTUPlace3	RePlAce	PROS
Quadratic Assignment			SimPL	ePlace	DREAMPlace	PL-GNN

Figure 1. Description of placement algorithms and the corresponding placers in academia. As the design size increased, the dominant placement algorithms shifted from the early partitioning-based method to heuristic algorithms. Later, the partitioning method began prevailing again due to the application of the multi-level framework. In some recent academic contests, such as ISPD [3–7], DAC [8], and ICCAD [9–11], the analytical methods provided better results than the previous algorithms. Nowadays, the leading placers mainly use nonlinear rather than quadratic methods because of the higher quality obtained with nonlinear methods. Furthermore, learning-based algorithms are gaining more and more attention, and a well-known example is Google's automatic macro placer [12].

Scholars have recently strived to achieve higher-quality placement results in less time. Related studies have been done to organize the development of placement optimization. G. Huang et al. [13] presented a comprehensive review of existing machine learning (ML) techniques for electronic design automation (EDA) studies. They organized the recent developments in placement optimization into three categories based on the degree of automation: traditional placer enhancement, routing information prediction, and placement decision making. Hao et al. [14] provided a review of existing ML techniques for placement studies and categorized them as intelligent placement and kernelized placement. There is a little discussion on accelerated placement computing in [13]. While [14] notes the importance of accelerated computing platforms, there is less discussion of algorithms—i.e., only partial ML techniques are discussed.

Inspired by [13,14], we observe that recent developments in placement optimization mostly have the ultimate goal of accelerating VLSI PD. For example, some have worked through algorithmic improvement to obtain better placement quality (e.g., routability and timing), which may have non-negligible overhead, but will eventually bring time benefits due to the reduction of flow iterations. Some researchers have directly accelerated the computation of a placer (e.g., applying a GPU-accelerated platform). Some have added predictions of the quality of candidate placement result to avoid unnecessary flow iterations. From this perspective, we group the placement studies into two categories: placement optimizers and learning-based predictors. Figure 2 depicts the new trends of VLSI PD flow. Figure 3 further classifies the optimizers and predictors, the details of which will be presented in the subsequent section. The optimizers perform online placement optimization, emphasizing algorithmic optimization and accelerated computation to get good placement quickly. The predictors perform offline placement optimization, highlighting using placement results to make predictions about downstream metrics and feeding back to guide better placement. To narrow our discussion, we focus on the placement of ASICs and exclude FPGA, analog circuits, etc. The reason is that the placement studies for these different types of circuits are similar, with ASICs being the most studied.

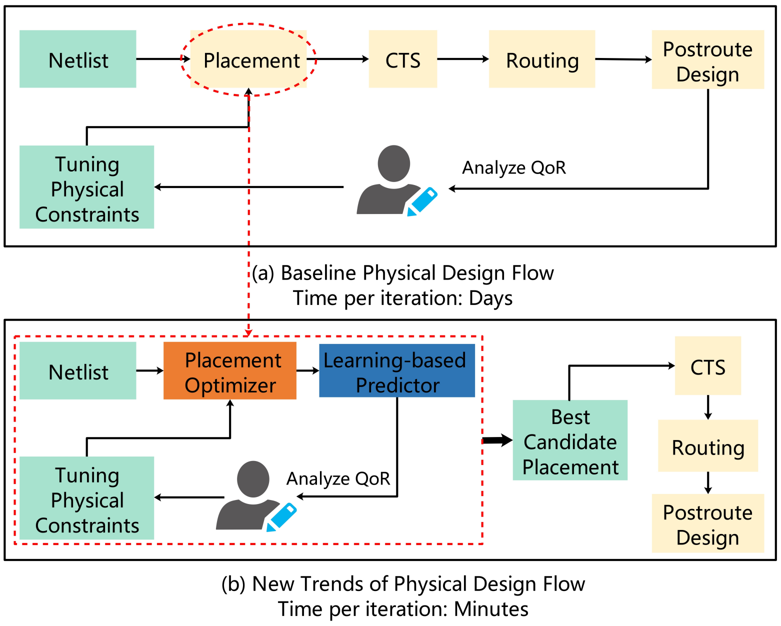


Figure 2. New trends of placement-centric VLSI physical design flow. This flow contains two key technologies: (i) a placement optimizer, which contains algorithm improvement and computational acceleration, and (ii) a learning-based predictor, which is used to classify postplacement results.

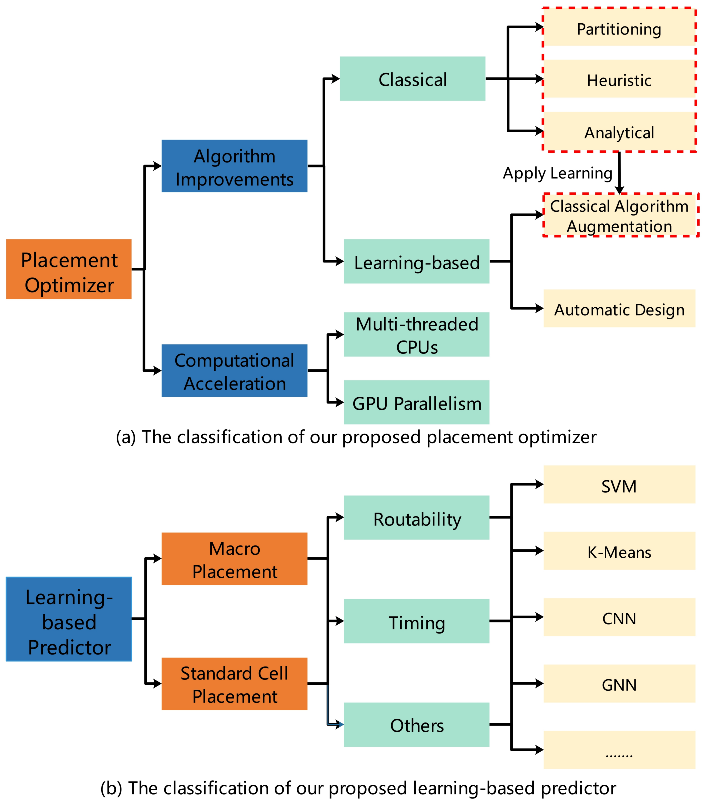


Figure 3. The classification of the placement optimizers and learning-based predictors. The optimizers perform online placement optimization and the predictors perform offline placement optimization. The details will be introduced in the subsequent section.

The characteristics of this paper are listed in the following:

1. We provide a comprehensive overview of placement optimization from the perspective of accelerating VLSI PD. Problem definition, classical placement algorithms,

computational acceleration, learning techniques, new trends, and new challenges are all well organized and discussed. The above are essential for understanding placement and advancing research but may have been simplified or ignored in previous reviews.

2. We observe a new trend of placement-centric VLSI PD flow and classify the placement development into optimizers and predictors. We discuss the deficiencies of classical placement algorithms in advanced technology nodes and point out future development directions for placement to consider accelerating VLSI PD.

This paper is organized as follows. Section 2 defines the placement problem. Section 3 reviews classical placement algorithms and discusses their application bottlenecks in advanced technology nodes. Sections 4 and 5 detail the placement optimizer and the learning-based predictor. Section 6 outlines placement challenges for accelerating VLSI physical design, and Section 7 summarizes the work of this survey.

2. Problem Definition

Placement is a multi-objective optimization problem that requires a rational determination of the locations of macros or standard cells. The problem can be described as follows [15]: Given the placement region P, the input netlist $G = (V, E)$ (V denotes the macros or standard cells to be placed; E denotes the connections between circuit cells), and the objective function $F = (V, E)$, find the location (x_i, y_i) of each $v_i \in V$ such that (1) the objective function $F = (V, E)$ is minimized and (2) the design constraints are satisfied. The objectives are usually one or more of these: wirelength [3], routability [8], timing [10], power [16], thermal [17], and manufacturability [18]. The design constraints typically include cells not overlapping, cells being placed within the placement region, placement density being less than a threshold, fence region constraints [7], and so on. The visualization of the placement process is given in Figure 4.

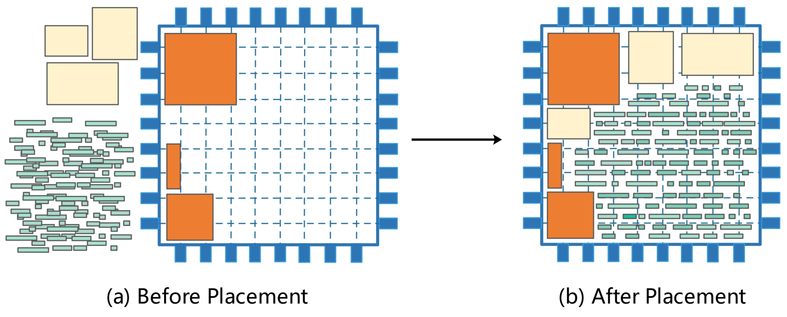


Figure 4. Description of the placement process. The placement region is usually rectangular and can be divided into several bins. The blue rectangles form the placement region. The orange, milk-white, and green rectangles represent fixed-position macros, movable macros, and movable standard cells, respectively.

Previous placement studies focused on optimizing the algorithm itself and less on both accelerated computation and ML-aided to accelerate VLSI PD. We attribute these studies to classical placement algorithms. As they are essential for understanding the development of placement and advancing research, classical placement algorithms are presented in Section 3.

3. Classical Placement Algorithms

This section reviews classical placement algorithms, including partitioning-based algorithms, heuristic algorithms, and analytical methods. Table 1 summarizes the core ideas, strengths, and weaknesses of the three types of algorithms and the corresponding typical placers.

Table 1. The core ideas (*), strengths (+), and weaknesses (−) of the three types of algorithms and the corresponding typical placers ($).

Type	Description
Partitioning-based Algorithms	(*) Divide and conquer (+) Efficient and scalable, can be used to solve large-scale circuits (−) Poor placement quality due to lack of global or local information (−) Harder to handle multiple objectives simultaneously ($) Capo [19] and FengShui [20]
Heuristic Algorithms	(*) Stochastic/hill-climbing methods (+) Simple to implement, high quality for small designs (+) Easier to handle multiple objectives simultaneously (−) Slower and less scalable for large-scale circuits ($) Dragon [21] and TimberWolf [22]
Analytical Methods	(*) Mathematical programming (+) More efficient and scalable (+) Better quality for even large-scale designs (+) Easier to consider multiple objectives simultaneously (−) Harder to optimize macro orientations ($) Quadratic: FastPlace [23], Polar [24], SimPL [25] ($) Nonlinear: NTUplace [26], ePlace [27], RePlAce [28]

3.1. Partitioning-Based Algorithms

The partitioning-based algorithm uses the divide-and-conquer strategy to recursively partition the input netlist and placement region, and then assigns smaller subnetlists to subregions in a top-down fashion. The partitioning of the netlist is usually based on a cost function that minimizes the number of cut edges that pass between adjacent subregions, so the partitioning method is also often referred to as the min-cut algorithm.

According to the different underlying computational methods, the partitioning methods can be divided into four categories [29], which are move-based methods, geometric representation-based methods, combinatorial formulations, and clustering methods. Move-based algorithms improve the initial feasible solution by repeated local perturbations, such as swapping two cells, and the relevant typical algorithms are Kernighan–Lin (KL) [30] and Fiduccia–Mattheyses (FM) [31]. Geometric-representation-based methods use geometric embedding of circuits to improve the clustering quality and runtime [32]. Combinatorial formulations can capture complex objective functions and constraints, and related techniques include network flow and mixed-integer linear programming [33]. Clustering algorithms use a multilevel paradigm [34], usually categorized as bottom-up or top-down.

The partitioning-based algorithm is efficient and scalable. However, the minimum cut-based algorithm does not take wirelength as the optimization objective, and the minimum cut is not equal to the minimum wirelength. Its independent optimization of each subproblem leads to an inadequate grasp of the overall problem, which can easily lead to routing congestion. The above two factors make the final placement quality obtained by the method poor. In addition, the optimization objective of the partitioning method is the number of cut edges between sub-regions, which leads to the limited ability to solve multi-objective optimization problems. Currently, the partitioning method is seldom used for placement directly but is often used in the preprocessing of placement to reduce the problem size.

Typical partitioning-based placers are Capo [19] and FengShui [20].

3.2. Heuristic Algorithms

The heuristic algorithms used to solve the placement problem are simulated annealing (SA) [35], the genetic algorithm [36], particle swarm algorithm [37], the bee colony algorithm [38], etc., among which, SA is more widely used because of its simplicity.

SA randomly perturbs the macro or standard cell and iterates several times to get an optimized placement result. The algorithm first needs to generate an initial solution and then randomly swap the positions of circuit cells to accept a better solution while accepting a worse solution with a certain probability. The algorithm iterates this process until the temperature drops to a minimum, at which point the objective function converges to the optimal solution. The probability of accepting a worse solution decreases as the temperature decreases.

SA is straightforward, has high solution quality, can solve different optimization problems with only slight modifications to the cost function, and is generally for modeling various circuit problems. However, when the chip size becomes larger, the search space of SA increases dramatically, resulting in unacceptable runtime.

Typical placers based on heuristics are Dragon [21] and TimberWolf [22].

3.3. Analytical Methods

The current leading placers are based on analytical methods. This approach models the placement problem with mathematical programming and uses optimization methods to minimize the cost function. The analytical methods can be divided into quadratic and nonlinear placement, depending on the cost function. The following is a review of the quadratic and nonlinear methods, using wirelength-driven placement as an example, as wirelength has the advantage of correlating with other vital metrics, such as power and timing. The wirelength is usually defined as the total half-perimeter wirelength (HPWL):

$$\text{HPWL}(\mathbf{x},\mathbf{y}) = \sum_{e\in\mathcal{E}} \left\{ \max_{i,j\in e} |x_i - x_j| + \max_{i,j\in e} |y_i - y_j| \right\} \tag{1}$$

Since HPWL is not everywhere differentiable, the quadratic placers require a quadratic approximation of HPWL using various net models, such as Bound2Bound [39]. After the approximation, the wirelength of the quadratic placers can be defined as follows.

$$W(\mathbf{x},\mathbf{y}) = \frac{1}{2}\mathbf{x}^T Q_x \mathbf{x} + \mathbf{c}_x^T \mathbf{x} + \frac{1}{2}\mathbf{y}^T Q_y \mathbf{y} + \mathbf{c}_y^T \mathbf{y} + \text{ const.} \tag{2}$$

where both Q_x and Q_y in the equations are symmetric and positive-definite, so minimizing the wirelength is equivalent to solving the following system of two linear equations, which can be solved quickly by various Krylov-subspace methods [40]:

$$\begin{aligned} Q_x \mathbf{x} &= -\mathbf{c}_x \\ Q_y \mathbf{y} &= -\mathbf{c}_y \end{aligned} \tag{3}$$

However, if only wirelength optimization is considered, eventually, all cells overlap. Based on the method of reducing cell overlap, quadratic placers can be classified into three categories [39]: partitioning-based, force-directed, and warping-based. Partitioning-based quadratic placers are based on a divide-and-conquer approach that minimizes the cost at each level [41,42]. The force-directed method uses forces to disperse the individual cells [23], and the warping-based approach deforms the chip area and indirectly moves the cells [43].

The quadratic method is quite efficient, mainly due to the simplicity of the wirelength model and the speed of the quadratic programming solution. However, its placement quality is often worse compared to the nonlinear method.

Typical quadratic placers are FastPlace [23], Polar [24], and SimPL [25].

The nonlinear placer uses a smooth wirelength model to approximate HPWL and computes the gradients analytically. The common wirelength models used are *Log-Sum-Exp (LSE)* [44] and *Weighted-Average (WA)* [45], denoted, respectively, as follows.

$$W_{LSE}(\mathbf{e}) = \gamma \left(\ln \sum_{i \in e} \exp\left(\frac{x_i}{\gamma}\right) + \ln \sum_{i \in e} \exp\left(\frac{-x_i}{\gamma}\right) \right) \tag{4}$$

$$W_{WA}(\mathbf{e}) = \left(\frac{\sum_{i \in e} x_i \exp(x_i/\gamma)}{\sum_{i \in e} \exp(x_i/\gamma)} - \frac{\sum_{i \in e} x_i \exp(-x_i/\gamma)}{\sum_{i \in e} \exp(-x_i/\gamma)} \right) \tag{5}$$

where γ is the smoothing parameter used to control the model's accuracy. Here, we take the horizontal part of the wirelength smoothing model as an example, and the vertical part can be obtained similarly.

In nonlinear placement, the density constraint can be defined as:

$$D_b(\mathbf{x}, \mathbf{y}) = \sum_{v \in V} P_x(b, v) P_y(b, v) \tag{6}$$

where $D_b(x, y)$ is the total area of movable cells in bin b, P_x and P_y are the overlap functions of bin b and cell v along the x and y directions, respectively. As $D_b(\mathbf{x}, \mathbf{y})$ is neither smooth nor differentiable, there are various models used to smooth it, such as the Bell-shaped function [44], the sigmoid model [46], the Helmholtz function [47], and the eDensity model [48].

Then, the density constraint is merged into the objective function by the Lagrangian penalty function or the relaxation method:

$$\min \quad W(\mathbf{x}, \mathbf{y}) + \lambda \sum_{b} (D_b(\mathbf{x}, \mathbf{y}) - M_b)^2 \tag{7}$$

where M_b is the maximum allowable area of movable cells in bin b. This unconstrained problem can subsequently be solved by convex optimization, e.g., with the conjugate gradient method [49] or Nesterov method [48].

We use RePlAce [28] as an example to introduce the design parameters that determine the placement and how nonlinear algorithms work towards resolving it. Many design parameters determine the placement, including the bin granularity, the maximum target density, and the overflow threshold. In the initial stage, all standard cells are placed in the chip center. In this case, some bins' density is much higher than the maximum target density, thereby failing the requirement. Subsequently, standard cells are slowly dispersed under a force similar to the attraction and repulsion in the electric field under multiple iterations of the Nesterov algorithm so that the density of each bin tends to be uniform. As the iteration progresses, the parameter λ becomes larger and larger, and the primary optimization goal of Equation (7) will shift from the total wirelength to the density. The algorithm stops when the density overflow is less than the overflow threshold, such as 0.1. The smaller the bin granularity setting, the finer the placement, but the longer the runtime. The higher the maximum target density setting, the more concentrated the final cell distribution will be, resulting in a smaller total wirelength, though other metrics, such as routability, may be worse.

Nonlinear placers provide a more accurate representation of wirelength and thus achieve better placement quality. It can be well suited for solving multi-objective optimization problems by simply modifying the constraints [50]. However, the nonlinear optimization solution is very time-consuming compared with quadratic optimization. Thus, the nonlinear placer often uses a multilevel approach to reduce complexity, which in turn may degrade placement quality.

Typical nonlinear placers are NTUplace [26], ePlace [27], and RePlAce [28].

In general, analytical methods are more efficient and scalable. They can achieve better quality for even large-scale designs, and they more easily consider multiple objectives simultaneously. However, due to the nature of mathematical programming, analytical methods struggle to optimize macro orientations. When comparing the quadratic method and the nonlinear method, the conclusion is that the quadratic method solves quickly but sacrifices part of the solution's quality, and the nonlinear solution has higher quality but takes more time to solve the problem. The jury is still out on which analytical method is best. An applied intuition is to use the quadratic method if runtime is more important and the nonlinear method if solution quality is more essential. RePlAce [28] gives an application example: use the quadratic method to obtain the initial placement quickly, and then use the nonlinear method to obtain more excellent solution quality.

3.4. Discussions

In advanced technology nodes, VLSI PD faces a large design scale, and complex optimization objectives and constraints (such as signal integrity, power consumption, and timing) must be considered. In this case, classical placement algorithms face application bottlenecks, mainly as follows:

1. Classical placement algorithms have unsatisfactory solution quality. Limited by the huge solution space due to the large design scale, classical placement algorithms often have to compromise between runtime and solution quality, which brings suboptimal solutions.
2. Classical placement algorithms such as heuristics and nonlinear methods are very time-consuming when solving large-scale netlists. The vast time overhead does not help to complete VLSI PD quickly.
3. Classical placement algorithms lack foresight. Classical algorithms become complex or powerless when more optimization objectives and constraints are considered. Most classical algorithms in academic research only use wirelength as a metric for placement quality, which is far from predictive for downstream metrics. A poorly foresighted placement can lead to routing failures, costing time for design-flow iterations.

Through research on placement-related work in recent years, we believe that the future of placement in advanced technology nodes considering accelerating VLSI PD can be summarized in the following directions:

1. Algorithm improvements, including the improvement of classical placement algorithms and the newly proposed learning-based algorithms.
2. Computational acceleration, including the application of multi-threaded CPUs and GPUs to accelerate the placement-solution finding.
3. Learning-based predictors: application of ML or deep learning (DL) methods for predicting downstream metrics.

The new trend of placement-centric VLSI PD flow is shown in Figure 2. As depicted in Figure 3, we attribute directions 1 and 2 to the placement optimizer, which is introduced in Section 4. We present direction 3 in Section 5.

4. Placement Optimizer

The progress related to the placement optimizer includes algorithm improvements and computational acceleration. We present each of them in this section.

4.1. Algorithm Improvements

We divide the improvement directions of placement algorithms into classical and learning-based, as shown in Figure 3. The significant difference between them is whether they can learn, i.e., optimize a new problem using prior knowledge or experience. Classical algorithms do not have an explicit training or testing process, and they must solve a new issue from scratch each time. Learning-based algorithms, in contrast, introduce ML/DL to complete a placement. Learning algorithms are expected to reduce the design cycle further [12,51]. Due to their learning capability, only the trained models need to be fine-tuned and can be applied to solve a new problem.

4.1.1. Classical

Continuous optimization of classical placement algorithms is still essential for accelerating VLSI PD. The new developments of classical placement algorithms are summarized in Table 2.

Table 2. Recent developments in classical placement algorithms.

Type	Description
Partitioning-based Algorithms	Improve the existing multilevel framework [52,53] Identify placement relevant cell clusters [29]
Heuristic Algorithms	Narrow the search space [54,55] Design new heuristic algorithms [56–58]
Analytical Methods	Optimize mathematical models [27,28,48,50,59,60] Consider more new objectives and constraints [18,28,61–69]

Partitioning-based algorithms use divide-and-conquer to reduce the problem's complexity, but the lack of local or global information decreases placement quality. Chang et al. [52] proposed a damped-wave multilevel framework that uses macro-clustering to improve scalability and constructive unclustering of macros to assist a standard-cell placer to get better solutions. This new framework permits a more effective optimization processes due to considering both local and global information at an early stage. Another novel multilevel framework for macro placement has also been proposed in [53]. Aside from improving the multilevel approach, Fogaca et al. [29] explored the use of modularity-driven clustering to identify natural clusters in a given graph without the parameter tuning and size balancing constraints typically required by VLSI CAD partitioning methods. They showed that modularity-based clustering correlates better with actual netlist placements than traditional partitioning methods (hMETIS [70]).

Heuristic algorithms can get high placement quality, but when the problem size becomes large, the solution is time-consuming and thus not applicable. Narrowing the search space helps reduce the runtime. In [54,55], the authors proposed a simulated evolution algorithm that perturbs a solution according to the quality of macro placement, and only partial macros need to be repositioned in each iteration, which facilitates its fast convergence to a good solution. Aside from the SA algorithm, other heuristics are also effective at solving placement problems, such as improved discrete particle swarm algorithms [57], improved genetic algorithms [56], and some hybrid heuristics [58].

As for analytical methods, we summarize their development in two directions, which are optimizing mathematical models [27,28,48,50,59,60] and considering new optimization objectives and constraints [18,28,61–69].

The mathematical model is optimized in three aspects: wirelength model, density function, and optimization algorithm. A novel wirelength model called BiG is proposed in [59], which combines the advantages and avoids the disadvantages of existing wirelength models to improve the placement quality. Lu et al. [27,48] developed a nonlinear placer called ePlace, which builds a new density function called eDensity and uses the Nesterov method to achieve faster convergence. Based on [48], Zhu et al. [60] proposed Pplace, which uses an analytical method to calculate the potential energy of electrostatic systems and achieve a lower wirelength than ePlace. Later, Cheng et al. [28] developed RePlAce, which further modifies the density function and extends the global placer to address routability. Another work in [50] proposed a new optimization algorithm, namely, the generalized augmented Lagrangian method, to solve nonlinear placement, and they showed that the method achieves good quality and is robust for handling different objectives.

Classical analytical methods in academic research often simply minimize wirelength and satisfy density constraints. More optimization objectives and design constraints are now to be considered for better placement quality. These optimization objectives mainly include routability [28,61,63,67,69], timing [62,68], manufacturability [18], and thermal [66]; and the constraints mainly include technology and fence region constraints [61,64] and 3D-IC constraints [65]. We discuss this topic with a recent work [69] on routability-driven placement. The authors of [69] applied an analytical method and a multilevel framework to address global and local routing congestion. To remove global routing congestion, they represented each net as a movable soft module and proposed a congestion-aware net penalty model integrated into the optimization objectives. To reduce local routing congestion, previous works relied on the accuracy of a congestion map. However, the congestion map is inaccurate at a high level of the multilevel framework due to the simple net connectivity between clusters. To this end, they proposed an inflation technique to expand each cluster according to the congestion value occupied by the cluster and its internal connectivity intensity. Experimental results on industrial datasets described that their method could obtain better routability and wirelength than other methods such as RePlAce [28] and NTUplace4h [26].

4.1.2. Learning-Based

Learning-based algorithms have been a hot topic in academic research in recent years. We categorize related studies into classical algorithm augmentation and automatic design, depending on the degree of ML embedding. The related work is summarized in Table 3.

Table 3. Recent studies on learning-based placement optimization algorithms.

Type	Description	Algorithm
Classical Algorithm Augmentation	Guide standard cells for partitioning [71,72]	K-Means GNN
	Improve heuristic placement quality [73,74]	RL
	Explore better heuristic rules in analysis [75]	RL
	Automatic parameter tuning [76–82]	RL GNN Bayesian Opt
Automatic Design	Automatic macro placement [12,51,83–86]	RL GCN CNN
	Automatic heuristic design [87]	RL

Classical algorithm augmentation refers to improving classical placement algorithms by applying learning techniques, where most work is decision guidance, such as parameter recommendation and heuristic rule exploration. We further group the classical algorithm augmentation into four categories: learning-based partitioning [71,72], learning-based heuristics [73,74], learning-based analytical methods [75], and automatic parameter tuning [76–82]. The automatic design approach has deeper ML embedding than the classical algorithm augmentation. It mainly uses reinforcement learning (RL) to train an agent end-to-end for automatic placement, which can significantly improve placement efficiency [12,51]. However, there is less related work available.

One of the typical works for learning-based partitioning is [71,72]. The authors focused on optimizing standard cell placement and proposed a graph-learning-based framework called PL-GNN, which generates logical affinity-based cell clustering for commercial placers. Graph neural networks (GNNs) and k-means clustering are used to get standard cell clusters. First, the netlist graph and initial node features are input into the GNN for node representation learning. K-means clustering is performed with the learned node embedding information to generate standard cell clusters. Finally, the clustering results are used as guidance to complete optimization via a commercial placer. Through comparative experiments on two commercial multi-core CPU designs in the TSMC 28 nm technology node, PL-GNN outperformed the modularity-based clustering [29] and can yield better wirelength (−3.9%), power (−2.8%), and performance (+85.7%) than the default placement flow of Synopsys IC Compiler II. Since PL-GNN learns the node representation for every design instance by optimizing an unsupervised loss function, it can be generalized to any design.

Inspired by previous work [73], a combination of RL and heuristics [74] was proposed for the first time for completing macro placement. The RL agent learns to generate a sequence pair that serves as a good initialization for SA to explore the solution space further and develop an optimal solution. They showed that RL can provide better initialization for SA, and the macro placement is subsequently completed with better quality. The RL-driven heuristic optimization framework is shown in Figure 5. However, they did not conduct experiments with advanced technology node benchmarks, and the framework ignores generalization considerations.

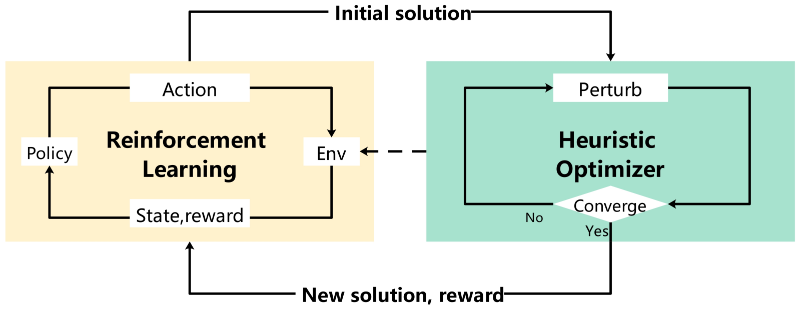


Figure 5. The reinforcement learning (RL)-driven heuristic optimization framework. This framework consists of the RL agent and the heuristic optimizer. The RL agent generates the initial solution for the heuristic optimizer. Then, the heuristic optimizer completes the execution until convergence conditions are met and returns a new solution and reward to the RL agent for retraining.

In the nonlinear placement optimization model, a density cost weight must be chosen to trade off the wirelength and density costs. RePlAce [28] and ePlace [65] improved the placement solution by developing different heuristic rules to select the appropriate density cost weight. Kirby et al. [75] proposed using RL to search for better heuristic rules that further reduce the total wirelength. Their experimental result showed an average improvement of 1% in final DP HPWL across academic benchmarks and more than 1% in GP HPWL on real industry designs. Their work was the first attempt to integrate RL into force-based global placement algorithms. Limited by open-source datasets used for training, the RL agent cannot be generalized well.

Classical placement algorithms take much time to adjust the parameters to get a satisfactory solution. Automatic parameter tuning by ML can improve placement efficiency. A RL framework for optimizing the placement parameters of commercial EDA tools is presented in [76,77]. The framework is shown in Figure 6. The state contains the netlist information and the current parameter set. The agent uses the actor–critic framework to get a new parameter set. The reward is the negative value of the HPWL output from the commercial EDA placement tool. The method uses a GNN to generate node embeddings, allowing the framework to generalize well to unseen netlists. Their experimental results under the TSMC 28nm technology node included up to 11% and 2.5% wirelength improvements on unseen netlists compared with a human engineer and a state-of-the-art tool auto-tuner, in just one placement iteration (20× and 50× less iterations). This work demonstrates the potential to accelerate VLSI PD through automatic parameter tuning. In addition, there are other efforts [78–82] for automatic-design-flow parameter tuning.

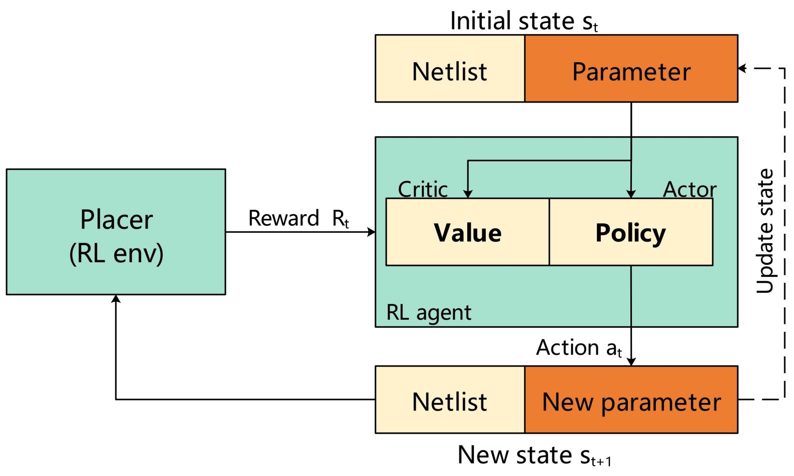


Figure 6. An automatic parameter optimization framework based on RL. The state contains the netlist information and the current parameter set. The agent uses the actor-critic framework to get a new parameter set. The reward is the negative value of the HPWL output from the placement engine environment.

Automatic design solves smaller-scale placement problems, especially macro placement. Based on the graph convolutional network (GCN) and RL, Google's method [12,51,83] trains an agent to automatically place macros and then place the standard cells using the force-directed method. When all the cells are placed, the reward can be computed and the RL policy is updated. They claim their approach can leverage experience to learn rich and transferable representations, making it applicable to placement optimization for unseen netlists. Their experimental results show that their well-trained agent can generate manufacturable chip floorplans in under six hours, which requires months of effort by human experts. The code is publicly available at: https://github.com/google-research/circuit_training (4 January 2023). However, their work cannot be done without the support of enormous computing power. Based on Google's work, DeepPlace and DeepPR were developed in [84] to optimize macro placement and to route with two RL agents jointly. Jiang et al. [85] replaced the force-directed method used by Google [12,51,83] with DREAMPlace [88] to place soft macros, which can quickly yield better placement results. Lai et al. [86] observed that the hypergraph constructed by previous work [12,51,83–85] is not scalable to comprehensively encode information of a netlist. They developed a novel RL method, named MaskPlace, which casts placement as a problem of pixel-level visual representation learning for circuit modules using convolutional neural networks (CNN). Experiments on public benchmarks show that MaskPlace outperforms existing RL approaches [12,51,83–85] in all key metrics, including wirelengh, congestion, and density. Another work [87] explored new heuristic algorithms for macro placement using RL. Specifically, a deep Q-Network agent was trained to walk through the search space by selecting a candidate neighbor solution at each step, where perturbing sequence pairs generate the neighbor solution.

It is easy to find that learning-based placement optimization algorithms mainly use RL to explore heuristic rules, obtain better initial solutions, make placement decisions, etc. GNNs are mainly used to generate node embeddings for netlists to allow RL agents to perform placement on unseen netlists and ensure generalizability. The combination of RL and GNN is expected to further promote RL as a generalized placement algorithm, like heuristic and analytic algorithms, and to be applied to accelerate VLSI PD.

4.2. Computational Acceleration

Placement problems are enormous and are often accompanied by heavy numerical computations. According to the new PD flow proposed in Figure 2, it is also necessary to speed up the acquisition of the placement results needed for the predictors, thereby reducing the time overhead. It makes sense to use modern accelerated computing hardware, such as multi-threaded CPUs or GPUs, to speed up these placement solvers.

There are architectural differences between CPUs and GPUs, as shown in Figure 7. They have the same components: control units, compute units (ALU), caches, and memory. However, their sizes and performances are different. GPUs have more control units, computing units, and caches, yet they are simpler and smaller. This implies that a GPU prefers parallel execution of small tasks with simple control flows [89]. Separate compute units on GPUs may not be as powerful as those on CPUs, but because of massive parallelism, GPUs may be faster. The related research on placement acceleration is listed in Table 4.

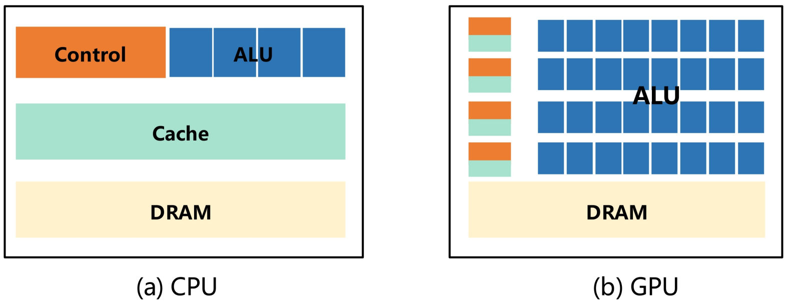


Figure 7. Comparison of CPU and GPU architectures. The orange, blue, green, and milk-white rectangles represent the control unit, computing unit, cache, and memory.

Table 4. Related research on multi-threaded CPUs and GPUs accelerating VLSI placement.

Type	Description
Multi-threaded CPUs	POLAR 3.0, a quadratic placer using a multi-core system [24]
	RePlace, a nonlinear placer with multi-threading support [28]
GPUs Parallelism	GPU-accelerated implementation for the TimberWolf [90]
	Exploit GPU parallelism to speed up nonlinear placement [91]
	DREAMPlace, a GPU-accelerated placement framework [64,68,88,92]
	An overview of GPU accelerated physical design [93]
	Accelerate density accumulation on GPU [94]

Lin et al. [24] proposed a quadratic global placer called POLAR 3.0, which can fully use a multi-core system. In POLAR 3.0, the GP iterations are divided into a series of frames, in which partitioning is applied based on cells' locations and the placement of each partition is performed simultaneously. Finally, POLAR 3.0 can achieve around 5× speedup over state-of-the-art academic placers with a typical quality degradation of 2–6%. RePlAce [28] is also implemented with multi-threading support, and the authors describe that this nonlinear global placer can achieve 2.4% reduction of routed wirelength on average and consume less than 2× runtime compared to a commercial placement tool.

GPUs are getting more attention than multi-threaded CPUs because they have better parallel potential and involved in the boom of AI applications. Related work has some on heuristics [90], while most related work focuses on nonlinear placement [64,68,88,91–94]. In [90], the authors present a parallel CUDA-implementation of the TimberWolf [22] placement application, achieving great speedups compared to the sequential algorithm. Lin et al. [91] exploited GPU parallelism to speed up nonlinear placement, considering the computation of wirelength and density, but missing the validation from real nonlinear placement flows. Another recent work [94] was devoted to accelerating the density accumulation of nonlinear placement on CPU/GPU, for which they have developed some new kernels.

Particularly of note is the DL engine DREAMPlace, a GPU-accelerated nonlinear placer developed by Lin et al. [64,68,88,92]. It leverages the GPU and the software framework

Pytorch, transforming the nonlinear placement problem into a neural network training problem. By decoupling high-level algorithm design from low-level acceleration efforts, DREAMPlace significantly reduces development and maintenance overheads. Compared to state-of-the-art multi-threaded CPUs placer RePlAce [28], it receives over 30× speedup without quality degradation. A design with 10 million standard cells can be completed in 5 min with DREAMPlace, compared to 3 h with RePlAce [95].

In advanced technology nodes, algorithm improvements and computational acceleration form a benign complementary relationship. Algorithm improvements allow us to get better placement results, and computational acceleration decreases the time overhead of the algorithm solution. Both of them jointly promote accelerating VLSI PD.

5. Learning-Based Predictor

Figure 2 reveals the roles of learning-based predictors. In the traditional VLSI PD flow, it is necessary to wait until the final routing to know whether the placement quality is good or bad, which inevitably leads to multiple iterations and lengthy design cycles. Suppose the placement quality can be predicted accurately in the early design stages. In that case, engineers can choose the best candidate placement result before proceeding to the downstream PD flow, which will significantly accelerate VLSI PD. Learning-based predictors use ML techniques to predict downstream outcomes in advance, allowing for design-adaptive flows with substantially fewer iterations (even "one-pass"), albeit with potential quality of result (QoR) degradation [96].

Learning-based predictors have been a hot topic in academic research. Depending on the input stage of the predictors, there are after-macro placement [97–101] and after-standard-cell placement [67,102–115]. According to the evaluation metrics, there are routing congestion [67,99–101,103,115], short violations [109–111], pin accessibility violations [102], wirelength [99], crosstalk [113], etc. By the algorithms used, there are mainly traditional supervised learning methods such as the support vector machine (SVM) [98,104,108], regression [99,100], and RUSBoost methods [105], but also DL methods such as fully convolutional network (FCNs) [67,103], CNNs [97,106], GCN [109], and generative adversarial networks (GANs) [114]. Compared with traditional ML methods, DL-based methods are superior at capturing the global information from a larger region of the circuit layout [116]. Compared with the analytical methods in Section 3.3, which directly optimize the total wirelength and density, the cost functions of the learning-based predictors are closely related to the adopted learning paradigm. The usual form of the cost function is to minimize the mean-square error of the labeled and predicted values when completing a regression task and to minimize the cross-entropy loss function when completing a classification task. The related work is organized in Table 5.

Table 5. Recent studies on learning-based predictors.

Input Stage	**Evaluation Metrics**	**Algorithms**
After Macro Placement	Timing failures [98]	SVM
	HPWL and routing congestion [99]	Regression
	Routing congestion [100]	K-Means, Regression
	Routing congestion and WNS/TNS [101]	Ensemble learning
	The number of DRVs [97]	CNN
After Standard Cell Placement	Routing congestion [67,103,115]	FCN, LHNN
	The number of DRVs [104–106]	SVM, RUSBoost, CNN
	The locations of DRVs [102,106–111]	SVM, GCN, CNN
	Clock tree synthesis outcomes [112]	GAN, RL
	Crosstalk [113]	XGBoost, GNN
	Coupling effect [114]	GAN

Making predictions immediately after macro placement saves the time consumed by standard cell placement but is limited by fewer available features, resulting in lower prediction accuracy. The methods used include both traditional supervised learning methods such as SVM [98] and regression [99,100]; traditional unsupervised learning methods such as k-means clustering [100]; and also some DL methods such as the CNN [97].

Chan et al. [98] used an SVM to predict post-P&R slack of SRAMs, given only a post-synthesis netlist, timing constraints, and a floorplan. Using their method, SoC designers may avoid floorplans and constraints leading to timing failures at sign-off. Other works focus on performing routability prediction [97,99–101]. Cheng et al. [100] constructed a dual-model architecture to predict routing congestion, which uses a combination of supervised and unsupervised learning to improve prediction accuracy. In addition, five ML regression algorithms were compared for their accuracy in predicting HPWL and routing congestion after macro placement [99]. Gao et al. [101] presented the first feasibility study of cross-design prediction on macro placement and developed a discerning ensemble approach to provide predictions on post physical synthesis QoR metrics, including routing congestion and WNS/TNS. Huang et al. [97] presented the first work on routability-driven macro placement using DL. They constructed a CNN-based routability predictor to predict design rule violations (DRVs) without standard cell placement and routing information. In addition, they embedded the predictor into a macro placer and used the SA algorithm to finish macro placement.

Making predictions after standard cell placement involves more research than macro placement predictions. This phenomenon may be due to the fact that more features can be extracted after standard cell placement, resulting in higher prediction accuracy. Such predictors mainly use methods such as traditional supervised learning [104,105,107,108,110,111,113], RL [112], and DL [67,102,103,106,109,111,112]. The predictors mainly perform routability prediction, and the metrics are mainly global routing congestion [67,103,115] and detailed routing violations [102,104–111].

Global routing congestion is one of the main causes of DRVs in detailed routing solutions, as most DRVs are due to overcrowded wires and vias [117]. PROS [103] is a predictor based on FCN that uses the data from standard cell-placement results to predict global routing congestion. Another work [67] also used an FCN to predict global routing congestion hotspots and embedded the predictor into DREAMPlace [88] to get a more route-friendly result. Wang et al. [115] proposed a novel graph formulation for circuits named lattice hypergraph (LH-graph) and further developed a heterogeneous graph neural network architecture, LHNN, to predict routing congestion. Their work aimed to reserve full netlist information during the whole learning process, fully utilize the supervision of the routing demand map and congestion map, and facilitate more appropriate receptive field expansion. However, there is a mismatch between the global routing congestion map and the final DRV map due to increasingly complex design rules in advanced technology nodes [107]. This miscorrelation is shown in Figure 8. Thus, predicting a congestion map is neither good for evaluating design routability nor identifying potential DRVs before detailed routing.

Most recent work has been devoted to directly predicting design rule check (DRC) violations, since DRC violation is the most accurate, ultimate measurement of routability. The prediction can be divided into two categories: the location (fine-grained) [102,106–111] and number (coarse-grained) [104–106] of DRVs. Chan et al. [107] modeled the prediction problem as a supervised classification problem to predict the locations of DRC hotspots. Chen et al. [108] built a learning-based framework and used an SVM to train a model for predicting violated regions and violation density. Several research efforts focused on predicting the locations of detailed routing short violations [109–111]. In [109], a GCN was used to establish a mapping relationship between placement features and detailed routing short violations. Tabrizi et al. [111] built a DL framework that predicts short violations without using a global router and treats the prediction problem as a binary classification problem with imbalanced data. Aside from pin short, pin accessibility is also a cause of detailed routing violations. Liang et al. [102] proposed a customized convolutional network architecture for general DRC hotspot prediction that emphasizes pin accessibility and does not need global routing information. The technology has been evaluated on several industrial designs at 7 nm technology nodes and was proven effective. For the evaluation of routability, Chan et al. [104] use the SVM algorithm to make predictions of the number of DRVs. Their approach does not need global routing information. In [105], the authors chose the RUSBoost algorithm instead of the SVM for predicting the number of short violations, demonstrating the superiority of the RUSBoost algorithm. RouteNet [106] is the first work to use CNN for routability prediction, and it can either predict the number of DRVs without global routing or detect the locations of DRC hotspots.

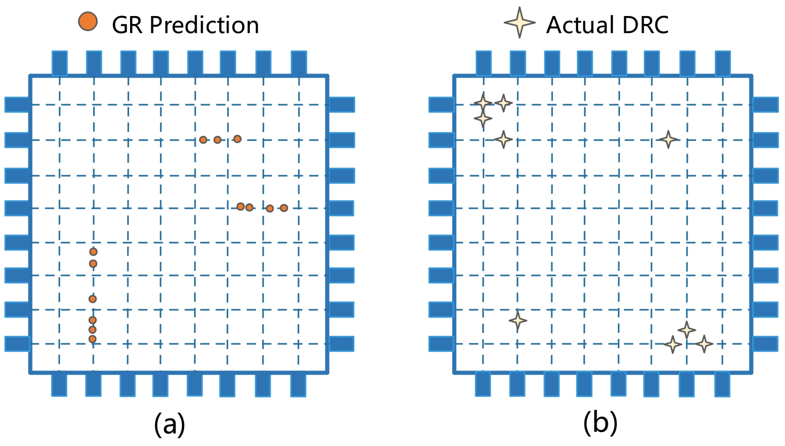


Figure 8. The mismatch between global routing congestion map and final design rule violations (DRVs) in advanced technology nodes. (**a**) Routing hotspots are predicted by global routing. (**b**) Actual routing hotspots.

Aside from routability, other aspects can also be predicted. Liang et al. [113] developed a crosstalk prediction model without routing by employing several ML algorithms, such as XGBoost, GraphSage, and graph attention networks. The framework allows designers to modify placement to minimize crosstalk in advance. Lu et al. [112] used conditional GAN and RL algorithms to build a framework that can predict and optimize clock tree synthesis (CTS) results. They used transfer learning to extract design features from placement images to ensure that the framework can be applied to different netlists. The method in [114] predicts the coupling effect based on a GAN model at global placement and integrates the model into the objective of DREAMPlace [88] to guide placement.

6. Open Challenges

As the technology nodes advance, many new challenges have emerged for placement optimization. We provide an overview of placement challenges from the perspective of accelerating VLSI PD. Specifically, we discuss the common challenge encountered by placement optimizers and learning-based predictors and then present the respective challenges.

The common challenge is the application of learning algorithms. These challenges include a lack of datasets, difficulty integrating the placer, poor generalization of trained models, and expensive training. As data-driven algorithms, learning algorithms most critically require a large amount of data, which is usually lacking in VLSI designs. Some work on data generation [12,51,67,118] was mainly based on existing several datasets and then modified the parameter settings with the help of EDA tools to get more datasets with different qualities. However, this is insufficient to achieve true design-level or netlist-level generalization. These two works [119,120] are worth mentioning for better generalization. The first study [119] proposes an artificial netlist generator which does not rely on real netlist data and can generate numerous netlists for ML training, promising models with netlist-level generalization capability. The second study [120] proposes federated learning to address data silos existing in academic research and industry research, promising design-level generalization while preserving commercial privacy. With the model size of recent learning algorithms continuously increasing, the efficient integration of the learning model into existing VLSI PD flow becomes difficult. Existing integration work has made such efforts: [67,97,103,107]. In the future, more new benchmarks, models, and metrics still be needed to drive placement research.

In addition to the common challenge mentioned above, placement optimizers also require attention on the following:

1. Classical placement algorithms do not require large amounts of data for training compared to learning algorithms. However, the solution time may be too long to apply to increasingly complex designs. Keeping the classical placement algorithms feasible is a crucial challenge. It may be necessary to revisit the process of classical placement algorithms, rebalance solution time and algorithm quality, use learning algorithms to circumvent drawbacks, and so on.
2. Advanced challenges of GPU acceleration. These challenges include a lack of parallelism and irregular computation patterns, high expectations for quality and inevitable quality degradation, a lack of available baseline implementations, and high development cost. Future efforts on GPU acceleration include algorithmic innovation, pushing the speed limit on very hard kernels, and generating universal frameworks or programming models [93].

Similarly, learning-based predictors require additional attention on the following:

1. What information needs to be predicted in different technology nodes. Each processing technology has its own rules. As a result, prediction for different technology nodes may require different methods and ML models. Taking routability prediction as an example, there is a large discrepancy between routing congestion and final detailed routing violations in advanced technology nodes, as shown in Figure 8. If routing congestion is predicted and placement is done under its guidance, it may lead to more modification work and a lengthy VLSI PD cycle.
2. Prediction needs to be more profound while maintaining accuracy. Prediction should increase its span across multiple design steps [121]. The authors of [122] also depicted that predicting post-route design quality during placement is preferred to predicting wirelength and congestion only. More profound predictions can help designers see further and avoid unnecessary iterations, thereby improving VLSI PD efficiency. However, the earlier the design, the fewer the features that can be extracted, which inevitably leads to a decline in model accuracy.

Placement optimization must be high quality and fast for accelerating VLSI PD. Pursuing high quality, with a focus on closing the sub-optimal gap, necessitates revisiting the classical placement optimization flow, and can use these approaches such as constructing artificial netlists, using accelerated computing platforms, improving the research infrastructure, and exploring learning algorithms. Achieving speed relates to placement-predictability research, which improves efficiency by predicting the output of current and downstream design optimization steps. As expected, the exploration of modern (multi-core, GPU) computing substrates and the application of learning algorithms are indispensable to achieving high-quality and fast placement.

7. Conclusions

In this paper, we comprehensively reviewed the progress of placement optimization from the perspective of accelerating VLSI physical design. After introducing the background of placement optimization research, we highlighted emerging trends in modern placement-centric VLSI physical design flows, including placement optimizers and learning-based predictors. We first defined the placement problem and reviewed the classical placement algorithms, then discussed the application bottleneck of the classical placement algorithms in advanced technology nodes and gave corresponding solutions.

After that, we introduced the development of placement optimizers from algorithm improvements and computational acceleration, pointing out that these two aspects will jointly promote accelerating VLSI physical design. We also presented research on learning-based predictors from various angles. Finally, we discussed the common and individual challenges encountered by placement optimizers and learning-based predictors.

Effective and fast placement remains a challenging problem. Although learning algorithms are becoming mainstream research, we should continue to maintain research on classical placement algorithms. There is still significant research space for learning-based placement algorithms. The construction of infrastructure, especially datasets, will become increasingly crucial for advancing learning-assisted placement research. RL and GNN deserve further research, which promises to deepen the automation of solving placement problems and accelerate VLSI physical design.

Author Contributions: Conceptualization, Y.Q. and Y.X.; investigation, Y.Q.; writing—original draft preparation, Y.Q.; writing—review and editing, Y.X., X.Z., and P.G.; writing—revision manuscript, Y.Q. and Y.X.; visualization, Y.Q. and X.Z.; supervision, X.X. and Y.X.; funding acquisition, Y.X and S.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the Industry-University Research Collaboration Project Funded by Zhuhai City (No. ZH22017001200095PWC), the Ministry of Education's Cooperative Education Project (No. 202102172025), and the Guangdong-Hong Kong-Macao Joint Innovation Field Project (No. 2021A0505080006).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Waldrop, M.M. The chips are down for Moore's law. *Nature* **2016**, *530*, 144–147. [CrossRef] [PubMed]
2. Garey, M.R.; Johnson, D.S.; Stockmeyer, L. Some simplified NP-complete problems. In Proceedings of the STOC '74, Sixth Annual ACM Symposium on Theory of Computing, Seattle, WA, USA, 30 April–2 May 1974; pp. 47–63. [CrossRef]
3. Nam, G.J.; Alpert, C.J.; Villarrubia, P.; Winter, B.; Yildiz, M. The ISPD2005 placement contest and benchmark suite. In Proceedings of the ISPD '05, 2005 International Symposium on Physical Design, San Francisco, CA, USA, 3–6 April 2005; pp. 216–220. [CrossRef]
4. Nam, G.J. ISPD 2006 Placement Contest: Benchmark Suite and Results. In Proceedings of the ISPD '06, 2006 International Symposium on Physical Design, San Jose, CA, USA, 9–12 April 2006; p. 167. [CrossRef]
5. Viswanathan, N.; Alpert, C.J.; Sze, C.; Li, Z.; Nam, G.J.; Roy, J.A. The ISPD-2011 routability-driven placement contest and benchmark suite. In Proceedings of the ISPD '11, 2011 International Symposium on Physical Design, Santa Barbara, CA, USA, 27–30 March 2011; pp. 141–146. [CrossRef]
6. Yutsis, V.; Bustany, I.S.; Chinnery, D.; Shinnerl, J.R.; Liu, W.H. ISPD 2014 benchmarks with sub-45nm technology rules for detailed-routing-driven placement. In Proceedings of the 2014 on International Symposium on Physical Design-ISPD '14, Petaluma, CA, USA, 30 March–2 April 2014; pp. 161–168. [CrossRef]
7. Bustany, I.S.; Chinnery, D.; Shinnerl, J.R.; Yutsis, V. ISPD 2015 Benchmarks with Fence Regions and Routing Blockages for Detailed-Routing-Driven Placement. In Proceedings of the 2015 Symposium on International Symposium on Physical Design, Monterey, CA, USA, 29 March–1 April 2015; pp. 157–164. [CrossRef]
8. Viswanathan, N.; Alpert, C.; Sze, C.; Li, Z.; Wei, Y. The DAC 2012 routability-driven placement contest and benchmark suite. In Proceedings of the 49th Annual Design Automation Conference on—DAC '12, San Francisco, CA, USA, 3–7 June 2012; p. 774. [CrossRef]
9. Viswanathan, N.; Alpert, C.; Sze, C.; Li, Z.; Wei, Y. ICCAD-2012 CAD contest in design hierarchy aware routability-driven placement and benchmark suite. In Proceedings of the International Conference on Computer-Aided Design-ICCAD '12, San Jose, CA, USA, 5–8 November 2012; p. 345. [CrossRef]
10. Kim, M.C.; Huj, J.; Viswanathan, N. ICCAD-2014 CAD contest in incremental timing-driven placement and benchmark suite: Special session paper: CAD contest. In Proceedings of the 2014 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), San Jose, CA, USA, 3–6 November 2014; pp. 361–366, ISSN 1558-2434. [CrossRef]
11. Kim, M.C.; Hu, J.; Li, J.; Viswanathan, N. ICCAD-2015 CAD contest in incremental timing-driven placement and benchmark suite. In Proceedings of the 2015 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), Austin, TX, USA, 2–6 November 2015; pp. 921–926. [CrossRef]
12. Mirhoseini, A.; Goldie, A.; Yazgan, M.; Jiang, J.W.; Songhori, E.; Wang, S.; Lee, Y.J.; Johnson, E.; Pathak, O.; Nazi, A.; et al. A graph placement methodology for fast chip design. *Nature* **2021**, *594*, 207–212. [CrossRef] [PubMed]
13. Huang, G.; Hu, J.; He, Y.; Liu, J.; Ma, M.; Shen, Z.; Wu, J.; Xu, Y.; Zhang, H.; Zhong, K.; et al. Machine Learning for Electronic Design Automation: A Survey. *ACM Trans. Des. Autom. Electron. Syst.* **2021**, *26*, 1–46. [CrossRef]

14. Hao, R.; Cai, Y.; Zhou, Q. Intelligent and kernelized placement: A survey. *Integration* **2022**, *86*, 44–50. [CrossRef]
15. Alpert, C.J.; Mehta, D.P.; Sapatnekar, S.S. (Eds.) *Handbook of Algorithms for Physical Design Automation*; OCLC: Ocn214935396; CRC Press: Boca Raton, FL, USA, 2009;
16. Cheon, Y.; Ho, P.H.; Kahng, A.B.; Reda, S.; Wang, Q. Power-aware placement. In Proceedings of the DAC '05, 42nd Annual Design Automation Conference, Anaheim, CA, USA, 13–17 June 2005; pp. 795–800. [CrossRef]
17. Kahng, A.B.; Kang, S.M.; Li, W.; Liu, B. Analytical thermal placement for VLSI lifetime improvement and minimum performance variation. In Proceedings of the 2007 25th International Conference on Computer Design, Lake Tahoe, CA, USA, 7–10 October 2007; pp. 71–77, ISSN 1063-6404. [CrossRef]
18. Huang, Y.C.; Chang, Y.W. Fogging Effect Aware Placement in Electron Beam Lithography. In Proceedings of the 54th Annual Design Automation Conference 2017, Austin, TX, USA, 18–22 June 2017; pp. 1–6. [CrossRef]
19. Roy, J.A.; Papa, D.A.; Adya, S.N.; Chan, H.H.; Ng, A.N.; Lu, J.F.; Markov, I.L. Capo: Robust and scalable open-source min-cut floorplacer. In Proceedings of the ISPD '05, 2005 International Symposium on Physical Design, San Francisco, CA, USA, 3–6 April 2005; pp. 224–226. [CrossRef]
20. Can Yildiz, M.; Madden, P.H. Improved cut sequences for partitioning based placement. In Proceedings of the 38th Conference on Design Automation—DAC '01, Las Vegas, NV, USA, 18–22 June 2001; pp. 776–779. [CrossRef]
21. Wang, M.; Yang, X.; Sarrafzadeh, M. Dragon2000: Standard-cell placement tool for large industry circuits. In Proceedings of the IEEE/ACM International Conference on Computer Aided Design, San Jose, CA, USA, 5–9 November 2000; pp. 260–263, ISSN 1092-3152. [CrossRef]
22. Sechen, C.; Sangiovanni-Vincentelli, A. The TimberWolf placement and routing package. *IEEE J. Solid-State Circuits* **1985**, *20*, 510–522. [CrossRef]
23. Viswanathan, N.; Chu, C.N. FastPlace: Efficient analytical placement using cell shifting, iterative local refinement,and a hybrid net model. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2005**, *24*, 722–733. [CrossRef]
24. Lin, T.; Chu, C.; Wu, G. POLAR 3.0: An ultrafast global placement engine. In Proceedings of the 2015 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), Austin, TX, USA, 2–6 November 2015; pp. 520–527. [CrossRef]
25. Kim, M.C.; Lee, D.J.; Markov, I.L. SimPL: An effective placement algorithm. In Proceedings of the 2010 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), San Jose, CA, USA, 7–11 November 2010; pp. 649–656. [CrossRef]
26. Hsu, M.K.; Chen, Y.F.; Huang, C.C.; Chou, S.; Lin, T.H.; Chen, T.C.; Chang, Y.W. NTUplace4h: A Novel Routability-Driven Placement Algorithm for Hierarchical Mixed-Size Circuit Designs. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2014**, *33*, 1914–1927. [CrossRef]
27. Lu, J.; Zhuang, H.; Chen, P.; Chang, H.; Chang, C.C.; Wong, Y.C.; Sha, L.; Huang, D.; Luo, Y.; Teng, C.C.; et al. ePlace-MS: Electrostatics-Based Placement for Mixed-Size Circuits. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2015**, *34*, 685–698. [CrossRef]
28. Cheng, C.K.; Kahng, A.B.; Kang, I.; Wang, L. RePlAce: Advancing Solution Quality and Routability Validation in Global Placement. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2019**, *38*, 1717–1730. [CrossRef]
29. Fogaça, M.; Kahng, A.B.; Reis, R.; Wang, L. Finding placement-relevant clusters with fast modularity-based clustering. In Proceedings of the ASPDAC '19, 24th Asia and South Pacific Design Automation Conference, Tokyo, Japan, 21–24 January 2019; pp. 569–576. [CrossRef]
30. Kernighan, B.W.; Lin, S. An Efficient Heuristic Procedure for Partitioning Graphs. *Bell Syst. Tech. J.* **1970**, *49*, 291–307. [CrossRef]
31. Fiduccia, C.; Mattheyses, R. A Linear-Time Heuristic for Improving Network Partitions. In Proceedings of the 19th Design Automation Conference, Las Vegas, NV, USA, 14–16 June 1982; pp. 175–181, ISSN 0146-7123. [CrossRef]
32. Ou, S.; Pedram, M. Timing-driven bipartitioning with replication using iterative quadratic programming. In Proceedings of the ASP-DAC '99 Asia and South Pacific Design Automation Conference 1999 (Cat. No.99EX198), Hong Kong, China, 21 January 1999; Volume 1, pp. 105–108. [CrossRef]
33. Blutman, K.; Fatemi, H.; Kahng, A.B.; Kapoor, A.; Li, J.; de Gyvez, J.P. Floorplan and placement methodology for improved energy reduction in stacked power-domain design. In Proceedings of the 2017 22nd Asia and South Pacific Design Automation Conference (ASP-DAC), Chiba, Japan, 16–19 January 2017; pp. 444–449. [CrossRef]
34. Sze, C.; Wang, T.-C. Performance-driven multi-level clustering for combinational circuits. In Proceedings of the ASP-DAC Asia and South Pacific Design Automation Conference, Kitakyushu, Japan, 21–24 January 2003; pp. 729–740. [CrossRef]
35. Chen, T.C.; Chang, Y.W. Modern floorplanning based on B/sup */-tree and fast simulated annealing. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2006**, *25*, 637–650.
36. Chen, X.; Lin, G.; Chen, J.; Zhu, W. An Adaptive Hybrid Genetic Algorithm for VLSI Standard Cell Placement Problem. In Proceedings of the 2016 3rd International Conference on Information Science and Control Engineering (ICISCE), Beijing, China, 8–10 July 2016; pp. 163–167. [CrossRef]
37. Kennedy, J.; Eberhart, R. Particle swarm optimization. In Proceedings of the ICNN'95—International Conference on Neural Networks, Perth, WA, Australia, 27 November–1 December 1995; Volume 4, pp. 1942–1948. [CrossRef]

38. Zaruba, D.; Zaporozhets, D.; Kureichik, V. Artificial Bee Colony Algorithm—A Novel Tool for VLSI Placement. In Proceedings of the First International Scientific Conference "Intelligent Information Technologies for Industry" (IITI'16), Rostov-on-Don–Sochi, Russia, 16–21 May 2016; Abraham, A., Kovalev, S., Tarassov, V., Snášel, V., Eds.; Series Title: Advances in Intelligent Systems and Computing; Springer International Publishing: Cham, Switzerland, 2016; Volume 450, pp. 433–442. [CrossRef]
39. Spindler, P.; Schlichtmann, U.; Johannes, F. Kraftwerk2—A Fast Force-Directed Quadratic Placement Approach Using an Accurate Net Model. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2008**, *27*, 1398–1411. [CrossRef]
40. Fan, S. An Introduction to Krylov Subspace Methods. *arXiv* **2018**, arXiv:1811.09025.
41. Kleinhans, J.; Sigl, G.; Johannes, F.; Antreich, K. GORDIAN: VLSI placement by quadratic programming and slicing optimization. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **1991**, *10*, 356–365. [CrossRef]
42. Brenner, U.; Struzyna, M. Faster and better global placement by a new transportation algorithm. In Proceedings of the 42nd Annual Conference on Design Automation—DAC '05, San Diego, CA, USA, 13–17 June 2005; pp. 591–596. [CrossRef]
43. Xiu, Z.; Rutenbar, R.A. Mixed-size placement with fixed macrocells using grid-warping. In Proceedings of the 2007 International Symposium on Physical Design—ISPD '07, Austin, TX, USA, 18–21 March 2007; pp. 103–110. [CrossRef]
44. Naylor, W.C.; Donelly, R.; Sha, L. Non-Linear Optimization System and Method for Wire Length and Delay Optimization for an Automatic Electric Circuit Placer. U.S. Patent 6301693B1, 9 October 2001.
45. Hsu, M.K.; Chang, Y.W.; Balabanov, V. TSV-aware analytical placement for 3D IC designs. In Proceedings of the DAC '11, 48th Design Automation Conference, San Diego, CA, USA, 5–10 June 2011; pp. 664–669. [CrossRef]
46. Chou, S.; Hsu, M.K.; Chang, Y.W. Structure-aware placement for datapath-intensive circuit designs. In Proceedings of the DAC '12, 49th Annual Design Automation Conference, San Francisco, CA, USA, 3–7 June 2012; pp. 762–767. [CrossRef]
47. Chan, T.F.; Cong, J.; Shinnerl, J.R.; Sze, K.; Xie, M. mPL6: Enhanced multilevel mixed-size placement. In Proceedings of the 2006 International Symposium on Physical Design—ISPD '06, San Jose, CA, USA, 9–12 April 2006; pp. 212–214. [CrossRef]
48. Lu, J.; Chen, P.; Chang, C.C.; Sha, L.; Huang, D.J.H.; Teng, C.C.; Cheng, C.K. ePlace: Electrostatics-Based Placement Using Fast Fourier Transform and Nesterov's Method. *ACM Trans. Des. Autom. Electron. Syst.* **2015**, *20*, 1–34. [CrossRef]
49. Chen, T.C.; Jiang, Z.W.; Hsu, T.C.; Chen, H.C.; Chang, Y.W. NTUplace3: An Analytical Placer for Large-Scale Mixed-Size Designs With Preplaced Blocks and Density Constraints. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2008**, *27*, 1228–1240. [CrossRef]
50. Zhu, Z.; Chen, J.; Peng, Z.; Zhu, W.; Chang, Y.W. Generalized augmented lagrangian and its applications to VLSI global placement. In Proceedings of the 55th Annual Design Automation Conference, San Francisco, CA, USA, 24–29 June 2018; pp. 1–6. [CrossRef]
51. Mirhoseini, A.; Goldie, A.; Yazgan, M.; Jiang, J.; Songhori, E.; Wang, S.; Lee, Y.J.; Johnson, E.; Pathak, O.; Bae, S.; et al. Chip Placement with Deep Reinforcement Learning. *arXiv* **2020**, arXiv:2004.10746.
52. Chang, C.H.; Chang, Y.W.; Chen, T.C. A novel damped-wave framework for macro placement. In Proceedings of the 2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), Irvine, CA, USA, 13–16 November 2017; pp. 504–511. [CrossRef]
53. Vidal-Obiols, A.; Cortadella, J.; Petit, J.; Galceran-Oms, M.; Martorell, F. RTL-Aware Dataflow-Driven Macro Placement. In Proceedings of the 2019 Design, Automation & Test in Europe Conference & Exhibition (DATE), Florence, Italy, 25–29 March 2019; pp. 186–191. [CrossRef]
54. Lin, J.M.; Deng, Y.L.; Yang, Y.C.; Chen, J.J.; Chen, Y.C. A Novel Macro Placement Approach based on Simulated Evolution Algorithm. In Proceedings of the 2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), Westminster, CO, USA, 4–7 November 2019; pp. 1–7. [CrossRef]
55. Lin, J.M.; Deng, Y.L.; Yang, Y.C.; Chen, J.J.; Lu, P.C. Dataflow-Aware Macro Placement Based on Simulated Evolution Algorithm for Mixed-Size Designs. *IEEE Trans. VLSI Syst.* **2021**, *29*, 973–984. [CrossRef]
56. Shunmugathammal, M.; Columbus, C.C.; Anand, S. A nature inspired optimization algorithm for VLSI fixed-outline floorplanning. *Analog. Integr. Circuits Signal Process.* **2020**, *103*, 173–186. [CrossRef]
57. Ye, Y.; Yin, X.; Chen, Z.; Hong, Z.; Fan, X.; Dong, C. A novel method on discrete particle swarm optimization for fixed-outline floorplanning. In Proceedings of the 2020 IEEE International Conference on Artificial Intelligence and Information Systems (ICAIIS), Dalian, China, 20–22 March 2020; pp. 591–595. [CrossRef]
58. Zaporozhets, D.; Zaruba, D.; Kulieva, N. Hybrid Heuristic Algorithm for VLSI Placement. In Proceedings of the 2019 International Russian Automation Conference (RusAutoCon), Sochi, Russia, 8–14 September 2019; pp. 1–5. [CrossRef]
59. Sun, F.K.; Chang, Y.W. BiG: A Bivariate Gradient-Based Wirelength Model for Analytical Circuit Placement. In Proceedings of the 56th Annual Design Automation Conference 2019, Las Vegas, NV, USA, 2–6 June 2019; pp. 1–6. [CrossRef]
60. Zhu, W.; Huang, Z.; Chen, J.; Chang, Y.W. Analytical solution of Poisson's equation and its application to VLSI global placement. In Proceedings of the International Conference on Computer-Aided Design, San Diego, CA, USA, 5–8 November 2018; pp. 1–8. [CrossRef]
61. Huang, C.C.; Lee, H.Y.; Lin, B.Q.; Yang, S.W.; Chang, C.H.; Chen, S.T.; Chang, Y.W.; Chen, T.C.; Bustany, I. NTUplace4dr: A Detailed-Routing-Driven Placer for Mixed-Size Circuit Designs with Technology and Region Constraints. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2018**, *37*, 669–681.
62. Mangiras, D.; Stefanidis, A.; Seitanidis, I.; Nicopoulos, C.; Dimitrakopoulos, G. Timing-Driven Placement Optimization Facilitated by Timing-Compatibility Flip-Flop Clustering. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2020**, *39*, 2835–2848. [CrossRef]

63. Arora, H.; Banerjee, A. A quadratic approach for routability driven placement design: Initial insight. In Proceedings of the 2015 International Conference on VLSI Systems, Architecture, Technology and Applications (VLSI-SATA), Bengaluru, India, 8–10 January 2015; pp. 1–6. [CrossRef]
64. Gu, J.; Jiang, Z.; Lin, Y.; Pan, D.Z. DREAMPlace 3.0: Multi-electrostatics based robust VLSI placement with region constraints. In Proceedings of the 39th International Conference on Computer-Aided Design, Virtual Event, 2–5 November 2020; pp. 1–9. [CrossRef]
65. Lu, J.; Zhuang, H.; Kang, I.; Chen, P.; Cheng, C.K. ePlace-3D: Electrostatics based Placement for 3D-ICs. In Proceedings of the ISPD '16, 2016 on International Symposium on Physical Design, Santa Rosa, CA, USA, 3–6 April 2016; pp. 11–18. [CrossRef]
66. Lin, J.M.; Chen, T.T.; Chang, Y.F.; Chang, W.Y.; Shyu, Y.T.; Chang, Y.J.; Lu, J.M. A fast thermal-aware fixed-outline floorplanning methodology based on analytical models. In Proceedings of the 2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), San Diego, CA, USA, 5–8 November 2018; pp. 1–8, ISSN 1558-2434.
67. Liu, S.; Sun, Q.; Liao, P.; Lin, Y.; Yu, B. Global Placement with Deep Learning-Enabled Explicit Routability Optimization. In Proceedings of the 2021 Design, Automation & Test in Europe Conference & Exhibition (DATE), Grenoble, France, 1–5 February 2021; pp. 1821–1824. [CrossRef]
68. Liao, P.; Liu, S.; Chen, Z.; Lv, W.; Lin, Y.; Yu, B. DREAMPlace 4.0: Timing-driven Global Placement with Momentum-based Net Weighting. In Proceedings of the 2022 Design, Automation & Test in Europe Conference & Exhibition (DATE), Antwerp, Belgium, 14–23 March 2022; pp. 939–944. [CrossRef]
69. Lin, J.M.; Huang, C.W.; Zane, L.C.; Tsai, M.C.; Lin, C.L.; Tsai, C.F. Routability-driven Global Placer Target on Removing Global and Local Congestion for VLSI Designs. In Proceedings of the 2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD), Munich, Germany, 1–4 November 2021; pp. 1–8. [CrossRef]
70. Karypis, G.; Kumar, V. *A Hypergraph Partitioning Package*; Army HPC Research Center, Department of Computer Science & Engineering, University of Minnesota: Minneapolis, MN, USA, 1998.
71. Lu, Y.C.; Pentapati, S.; Lim, S.K. The Law of Attraction: Affinity-Aware Placement Optimization using Graph Neural Networks. In Proceedings of the ISPD '21, 2021 International Symposium on Physical Design, Virtual Event, 22–24 March 2021; pp. 7–14. [CrossRef]
72. Lu, Y.C.; Lim, S.K. VLSI Placement Optimization using Graph Neural Networks. In Proceedings of the 34th Advances in Neural Information Processing Systems (NeurIPS) Workshop on ML for Systems, Virtual, 6–12 December 2020.
73. Cai, Q.; Hang, W.; Mirhoseini, A.; Tucker, G.; Wang, J.; Wei, W. Reinforcement Learning Driven Heuristic Optimization. *arXiv* **2019**, arXiv:1906.06639.
74. Vashisht, D.; Rampal, H.; Liao, H.; Lu, Y.; Shanbhag, D.; Fallon, E.; Kara, L.B. Placement in Integrated Circuits using Cyclic Reinforcement Learning and Simulated Annealing. *arXiv* **2020**, arXiv:2011.07577.
75. Kirby, R.; Nottingham, K.; Roy, R.; Godil, S.; Catanzaro, B. Guiding Global Placement With Reinforcement Learning. *CoRR* **2021**, abs/2109.02631,
76. Agnesina, A.; Pentapati, S.; Lim, S.K. A General Framework For VLSI Tool Parameter Optimization with Deep Reinforcement Learning. In Proceedings of the NeurIPS 2020 Workshop on Machine Learning for Systems, Virtual, 6–12 December 2020.
77. Agnesina, A.; Chang, K.; Lim, S.K. VLSI placement parameter optimization using deep reinforcement learning. In Proceedings of the 39th International Conference on Computer-Aided Design, Virtual Event, 2–5 November 2020; pp. 1–9. [CrossRef]
78. Kwon, J.; Ziegler, M.M.; Carloni, L.P. A Learning-Based Recommender System for Autotuning Design Flows of Industrial High-Performance Processors. In Proceedings of the 56th Annual Design Automation Conference 2019, Las Vegas, NV, USA, 2–6 June 2019; pp. 1–6. [CrossRef]
79. Xie, Z.; Fang, G.Q.; Huang, Y.H.; Ren, H.; Zhang, Y.; Khailany, B.; Fang, S.Y.; Hu, J.; Chen, Y.; Barboza, E.C. FIST: A Feature-Importance Sampling and Tree-Based Method for Automatic Design Flow Parameter Tuning. *arXiv* **2020**, arXiv:2011.13493.
80. Geng, H.; Xu, Q.; Ho, T.Y.; Yu, B. PPATuner: Pareto-driven tool parameter auto-tuning in physical design via gaussian process transfer learning. In Proceedings of the 59th ACM/IEEE Design Automation Conference, San Francisco, CA, USA, 10–14 July 2022; pp. 1237–1242. [CrossRef]
81. Geng, H.; Chen, T.; Ma, Y.; Zhu, B.; Yu, B. PTPT: Physical Design Tool Parameter Tuning via Multi-Objective Bayesian Optimization. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2022**, *42*, 178–189.
82. Geng, H.; Chen, T.; Sun, Q.; Yu, B. Techniques for CAD Tool Parameter Auto-tuning in Physical Synthesis: A Survey (Invited Paper). In Proceedings of the 2022 27th Asia and South Pacific Design Automation Conference (ASP-DAC), Taipei, Taiwan, 17–20 January 2022; pp. 635–640. [CrossRef]
83. Goldie, A.; Mirhoseini, A. Placement Optimization with Deep Reinforcement Learning. In Proceedings of the 2020 International Symposium on Physical Design, Taipei, Taiwan, 20–23 September 2020; pp. 3–7. [CrossRef]
84. Cheng, R.; Yan, J. On Joint Learning for Solving Placement and Routing in Chip Design. In Proceedings of the Advances in Neural Information Processing Systems 34, Virtual, 6–12 December 2020; pp. 16508–16519.
85. Jiang, Z.; Songhori, E.; Wang, S.; Goldie, A.; Mirhoseini, A.; Jiang, J.; Lee, Y.J.; Pan, D.Z. Delving into Macro Placement with Reinforcement Learning. *arXiv* **2021**, arXiv:2109.02587.
86. Lai, Y.; Mu, Y.; Luo, P. MaskPlace: Fast Chip Placement via Reinforced Visual Representation Learning. *arXiv* **2022**, arXiv:2211.13382.

87. He, Z.; Ma, Y.; Zhang, L.; Liao, P.; Wong, N.; Yu, B.; Wong, M.D. Learn to Floorplan through Acquisition of Effective Local Search Heuristics. In Proceedings of the 2020 IEEE 38th International Conference on Computer Design (ICCD), Hartford, CT, USA, 18–21 October 2020; pp. 324–331. [CrossRef]
88. Lin, Y.; Dhar, S.; Li, W.; Ren, H.; Khailany, B.; Pan, D.Z. DREAMPlace: Deep Learning Toolkit-Enabled GPU Acceleration for Modern VLSI Placement. In Proceedings of the 56th Annual Design Automation Conference 2019, Las Vegas, NV, USA, 2–6 June 2019; pp. 1–6. [CrossRef]
89. Lin, Y.; Li, W.; Gu, J.; Ren, H.; Khailany, B.; Pan, D.Z. ABCDPlace: Accelerated Batch-Based Concurrent Detailed Placement on Multithreaded CPUs and GPUs. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2020**, *39*, 5083–5096. [CrossRef]
90. Al-Kawam, A.; Harmanani, H.M. A Parallel GPU Implementation of the Timber Wolf Placement Algorithm. In Proceedings of the 2015 12th International Conference on Information Technology—New Generations, Las Vegas, NV, USA, 13–15 April 2015; pp. 792–795. [CrossRef]
91. Lin, C.X.; Wong, M.D.F. Accelerate analytical placement with GPU: A generic approach. In Proceedings of the 2018 Design, Automation Test in Europe Conference Exhibition (DATE), Dresden, Germany, 19–23 March 2018; pp. 1345–1350, ISSN 1558-1101. [CrossRef]
92. Lin, Y.; Pan, D.Z.; Ren, H.; Khailany, B. DREAMPlace 2.0: Open-Source GPU-Accelerated Global and Detailed Placement for Large-Scale VLSI Designs. In Proceedings of the 2020 China Semiconductor Technology International Conference (CSTIC), Shanghai, China, 26 June 2020–17 July 2020; pp. 1–4. [CrossRef]
93. Lin, Y. GPU acceleration in VLSI back-end design: Overview and case studies. In Proceedings of the 39th International Conference on Computer-Aided Design, Virtual, 2–5 November 2020; pp. 1–4. [CrossRef]
94. Guo, Z.; Mai, J.; Lin, Y. Ultrafast CPU/GPU Kernels for Density Accumulation in Placement. In Proceedings of the 2021 58th ACM/IEEE Design Automation Conference (DAC), San Francisco, CA, USA, 5–9 December 2021; pp. 1123–1128. [CrossRef]
95. Lee, C.K. Deep Learning Creativity in EDA. In Proceedings of the 2020 International Symposium on VLSI Design, Automation and Test (VLSI-DAT), Hsinchu, Taiwan, 10–13 August 2020; p. 1, ISSN 2472-9124. [CrossRef]
96. Kahng, A.B. New directions for learning-based IC design tools and methodologies. In Proceedings of the 2018 23rd Asia and South Pacific Design Automation Conference (ASP-DAC), Jeju, Republic of Korea, 22–25 January 2018; pp. 405–410. [CrossRef]
97. Huang, Y.H.; Xie, Z.; Fang, G.Q.; Yu, T.C.; Ren, H.; Fang, S.Y.; Chen, Y.; Hu, J. Routability-Driven Macro Placement with Embedded CNN-Based Prediction Model. In Proceedings of the 2019 Design, Automation & Test in Europe Conference & Exhibition (DATE), Florence, Italy, 25–29 March 2019; pp. 180–185. [CrossRef]
98. Chan, W.T.J.; Chung, K.Y.; Kahng, A.B.; MacDonald, N.D.; Nath, S. Learning-based prediction of embedded memory timing failures during initial floorplan design. In Proceedings of the 2016 21st Asia and South Pacific Design Automation Conference (ASP-DAC), Macao, China, 25–28 January 2016; pp. 178–185. [CrossRef]
99. Cheng, W.K.; Guo, Y.Y.; Wu, C.S. Evaluation of routability-driven macro placement with machine-learning technique. In Proceedings of the 2018 7th International Symposium on Next Generation Electronics (ISNE), Taipei, Taiwan, 7–9 May 2018; pp. 1–3. [CrossRef]
100. Cheng, W.K.; Wu, C.S. Machine Learning Techniques for Building and Evaluation of Routability-driven Macro Placement. In Proceedings of the 2019 IEEE International Conference on Consumer Electronics—Taiwan (ICCE-TW), Yilan, Taiwan, 20–22 May 2019; pp. 1–2. [CrossRef]
101. Gao, X.; Jiang, Y.M.; Shao, L.; Raspopovic, P.; Verbeek, M.E.; Sharma, M.; Rashingkar, V.; Jalota, A. Congestion and Timing Aware Macro Placement Using Machine Learning Predictions from Different Data Sources: Cross-design Model Applicability and the Discerning Ensemble. In Proceedings of the ISPD '22, 2022 International Symposium on Physical Design, Virtual, 27–30 March 2022; pp. 195–202. [CrossRef]
102. Liang, R.; Xiang, H.; Pandey, D.; Reddy, L.; Ramji, S.; Nam, G.J.; Hu, J. DRC Hotspot Prediction at Sub-10 nm Process Nodes Using Customized Convolutional Network. In Proceedings of the ISPD '20, 2020 International Symposium on Physical Design, Taipei, Taiwan, 20–23 September 2020; pp. 135–142. [CrossRef]
103. Chen, J.; Kuang, J.; Zhao, G.; Huang, D.J.H.; Young, E.F.Y. PROS: A plug-in for routability optimization applied in the state-of-the-art commercial EDA tool using deep learning. In Proceedings of the 39th International Conference on Computer-Aided Design, Virtual, 2–5 November 2020; pp. 1–8. [CrossRef]
104. Chan, W.T.J.; Du, Y.; Kahng, A.B.; Nath, S.; Samadi, K. BEOL stack-aware routability prediction from placement using data mining techniques. In Proceedings of the 2016 IEEE 34th International Conference on Computer Design (ICCD), Scottsdale, AZ, USA, 2–5 October 2016; pp. 41–48. [CrossRef]
105. Tabrizi, A.F.; Darav, N.K.; Rakai, L.; Kennings, A.; Behjat, L. Detailed routing violation prediction during placement using machine learning. In Proceedings of the 2017 International Symposium on VLSI Design, Automation and Test (VLSI-DAT), Hsinchu, Taiwan, 24–27 April 2017; pp. 1–4. [CrossRef]
106. Xie, Z.; Huang, Y.H.; Fang, G.Q.; Ren, H.; Fang, S.Y.; Chen, Y.; Corporation, N. RouteNet: Routability prediction for mixed-size designs using convolutional neural network. In Proceedings of the International Conference on Computer-Aided Design, San Diego, CA, USA, 5–8 November 2018; pp. 1–8. [CrossRef]
107. Chan, W.T.J.; Ho, P.H.; Kahng, A.B.; Saxena, P. Routability Optimization for Industrial Designs at Sub-14nm Process Nodes Using Machine Learning. In Proceedings of the 2017 ACM on International Symposium on Physical Design, Portland, OR, USA, 19–22 March 2017; pp. 15–21. [CrossRef]

108. Chen, L.C.; Huang, C.C.; Chang, Y.L.; Chen, H.M. A learning-based methodology for routability prediction in placement. In Proceedings of the 2018 International Symposium on VLSI Design, Automation and Test (VLSI-DAT), Hsinchu, Taiwan, 16–19 April 2018; pp. 1–4. [CrossRef]
109. Chen, X.; Di, Z.X.; Wu, W.; Feng, Q.Y.; Shi, J.Y. Detailed Routing Short Violations Prediction Method Using Graph Convolutional Network. In Proceedings of the 2020 IEEE 15th International Conference on Solid-State Integrated Circuit Technology (ICSICT), Kunming, China, 3–6 November 2020; pp. 1–3. [CrossRef]
110. Tabrizi, A.F.; Rakai, L.; Darav, N.K.; Bustany, I.; Behjat, L.; Xu, S.; Kennings, A. A Machine Learning Framework to Identify Detailed Routing Short Violations from a Placed Netlist. In Proceedings of the 2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC), San Francisco, CA, USA, 24–28 June 2018; pp. 1–6. [CrossRef]
111. Tabrizi, A.F.; Darav, N.K.; Rakai, L.; Bustany, I.; Kennings, A.; Behjat, L. Eh?Predictor: A Deep Learning Framework to Identify Detailed Routing Short Violations From a Placed Netlist. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2020**, *39*, 1177–1190. [CrossRef]
112. Lu, Y.C.; Lee, J.; Agnesina, A.; Samadi, K.; Lim, S.K. GAN-CTS: A Generative Adversarial Framework for Clock Tree Prediction and Optimization. In Proceedings of the 2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), Westminster, CO, USA, 4–7 November 2019; pp. 1–8. [CrossRef]
113. Liang, R.; Xie, Z.; Jung, J.; Chauha, V.; Chen, Y.; Hu, J.; Xiang, H.; Nam, G.J. Routing-Free Crosstalk Prediction. In Proceedings of the 2020 IEEE/ACM International Conference On Computer Aided Design (ICCAD), Virtual, 2–5 November 2020; pp. 1–9, ISSN 1558-2434.
114. Lee, Y.Y.; Ruan, S.J.; Chen, P.C. Predictable Coupling Effect Model for Global Placement Using Generative Adversarial Networks with an Ordinary Differential Equation Solver. *IEEE Trans. Circuits Syst. II* **2021**, *69*, 2261–2265. [CrossRef]
115. Wang, B.; Shen, G.; Li, D.; Hao, J.; Liu, W.; Huang, Y.; Wu, H.; Lin, Y.; Chen, G.; Heng, P.A. LHNN: Lattice hypergraph neural network for VLSI congestion prediction. In Proceedings of the DAC '22, 59th ACM/IEEE Design Automation Conference, San Francisco, CA, USA, 10–14 July 2022; pp. 1297–1302. [CrossRef]
116. Xie, Z.; Pan, J.; Chang, C.C.; Liang, R.; Barboza, E.C.; Chen, Y. Deep Learning for Routability. In Machine Learning Applications in Electronic Design Automation; Ren, H., Hu, J., Eds.; Springer International Publishing: Cham, Switzerland, 2022; pp. 35–61, ISBN 978-3-031-13074-8.
117. Chen, G.; Pui, C.W.; Li, H.; Chen, J.; Jiang, B.; Young, E.F.Y. Detailed routing by sparse grid graph and minimum-area-captured path search. In Proceedings of the ASPDAC '19, 24th Asia and South Pacific Design Automation Conference, Tokyo, Japan, 21–24 January 2019; pp. 754–760. [CrossRef]
118. Chai, Z.; Zhao, Y.; Lin, Y.; Liu, W.; Wang, R.; Huang, R. CircuitNet: An Open-Source Dataset for Machine Learning Applications in Electronic Design Automation (EDA). *arXiv* **2022**, arXiv:2208.01040.
119. Kim, D.; Kwon, H.; Lee, S.Y.; Kim, S.; Woo, M.; Kang, S. Machine Learning Framework for Early Routability Prediction with Artificial Netlist Generator. In Proceedings of the 2021 Design, Automation & Test in Europe Conference & Exhibition (DATE), Grenoble, France, 1–5 February 2021; pp. 1809–1814. [CrossRef]
120. Pan, J.; Chang, C.C.; Xie, Z.; Li, A.; Tang, M.; Zhang, T.; Hu, J.; Chen, Y. Towards collaborative intelligence: Routability estimation based on decentralized private data. In Proceedings of the DAC '22, 59th ACM/IEEE Design Automation Conference, San Francisco, CA, USA, 10–14 July 2022; pp. 961–966. [CrossRef]
121. Kahng, A.B. INVITED: Reducing Time and Effort in IC Implementation: A Roadmap of Challenges and Solutions. In Proceedings of the Proceedings of the 55th Annual Design Automation Conference, San Francisco, CA, USA, 24–29 June 2018; pp. 1–6. [CrossRef]
122. Khailany, B.; Ren, H.; Dai, S.; Godil, S.; Keller, B.; Kirby, R.; Klinefelter, A.; Venkatesan, R.; Zhang, Y.; Catanzaro, B.; et al. Accelerating Chip Design With Machine Learning. *IEEE Micro* **2020**, *40*, 23–32. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.