PPA-driven Placement via Adaptive Cluster Constraints Optimization

Ziyan Liu[1], Siyuan Xu[2], Jie Wang[1†], Zijie Geng[1], Yeqiu Chen[1], Mingxuan Yuan[2], Jianye Hao[2,3], Feng Wu[1]
[1]MoE Key Laboratory of Brain-inspired Intelligent Perception and Cognition, University of Science and Technology of China
[2]Huawei Noah's Ark Lab, [3]Tianjin University
[†]Corresponding Author. E-mail: jiewangx@ustc.edu.cn

Abstract*—The clustering-based placement framework has demonstrated promising potential in improving the efficiency and quality of very-large-scale integration (VLSI) placement. However, existing methods typically impose unified and rule-based constraints on different clusters, overlooking the unique intra- and inter-cluster connection properties that vary across clusters, which leads to suboptimal results. To address this challenge and promote effective PPA optimization, we introduce an innovative PPA-driven placement paradigm with mixed-grained Adaptive Cluster Constraints Optimization (ACCO), which applies constraints with customized *constraint tightness* to different clusters, balancing local and global interactions for improved placement performance. Specifically, we propose a novel *eBound* model with quantified *constraint tightness*, combined with a Bayesian optimizer to dynamically adjust the constraints for each cluster based on PPA outcomes, which are ultimately passed on to the final flat placement. Experimental results on benchmarks across various domains show that our methods can achieve up to 62%, 97% and 25% improvements in post-route WNS, TNS and power compared to existing methods.

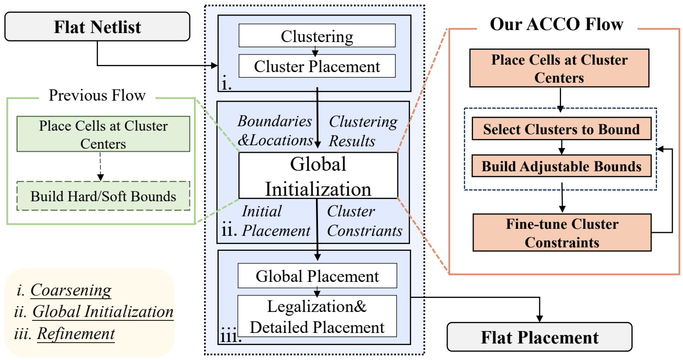

Fig. 1: We illustrate the typical clustering-based placement framework with three stages: (i) coarsening, (ii) global initialization and (iii) refinement. The coarsening stage clusters the netlist and obtains a coarse placement. Then, cells are initialized at their respective cluster centers. Finally, refinement is performed by analytical placer or local refinement heuristics. We also present our ACCO flow.

I. Introduction

Placement, which aims to find feasible locations for various instances under numerous given constraints, is a critical stage in VLSI physical design and is central to optimization of chip performance, power consumption, and area (PPA). Poor placement quality can significantly hinder design closure, ultimately increasing time-to-market. As modern designs grow in complexity and scale, following the trends predicted by Moore's Law, achieving PPA optimization has become increasingly challenging. Clustering-based methods, which partition large-scale designs into small clusters, have recently shown promising potential to enhance placement performance [1]–[4]. While clustering was initially introduced to reduce problem size and enhance placement efficiency by minimizing cuts [5]–[9], some recent approaches [1]–[4], [10], [11] have further incorporated design information into clustering to improve placement quality. For example, TritonPart [2] applies a timing-aware cost and slack propagation methodology to optimize cuts for both critical and non-critical timing paths. Other works, such as [4], [10], [11], consider design hierarchy, while [3] further integrates timing and power information. Additionally, [1] introduces a machine learning driven clustering method to improve placement performance. These methods can provide cluster constraints as placement guidance to improve the performance of analytical placer, such as considering design hierarchy to improve routability [10], [11] and bounding cells in a timing-critical path close to enhance timing performance [1], [3]. We illustrate a typical clustering placement framework in Fig. 1.

However, existing clustering methods often apply unified, rule-based constraints (hard or soft bounds) to all clusters, disregarding the intra- and inter-cluster connection properties that vary across different clusters, which leads to results far from optimal. For instance, when timing-critical paths are grouped, simply applying uniform hard bounds (e.g., a fence region that prevents cells from exceeding defined boundaries) to all clusters, as shown in Fig. 2 (a), can lead to suboptimal results: timing-critical paths are bounded close, while some non-critical paths turn critical due to excessive cuts and overly tight constraints. Conversely, applying soft bounds or leaving clusters entirely unconstrained fails to provide the necessary local refinement for extremely timing-critical paths, also leading to degradation in timing performance. Similar challenges arise when bounding nets with high switching activity.

To address this challenge and promote effective PPA optimization, we introduce an innovative PPA-driven placement paradigm through **A**daptive **C**luster **C**onstraints **O**ptimization (ACCO), a mixed-grained approach that enhances placement performance by applying constraints with customized *constraint tightness* to different clusters. Specifically, ACCO begins by identifying the clustering level for a given design. Then, with the generated clusters, mixed-grained optimization proceeds as follows (see Fig. 1): ACCO (i) selects clusters to constrain, (ii) builds bounds with tunable *constraint tightness* for the

978-3-9826741-1-7/DATE26/© 2026 EDAA

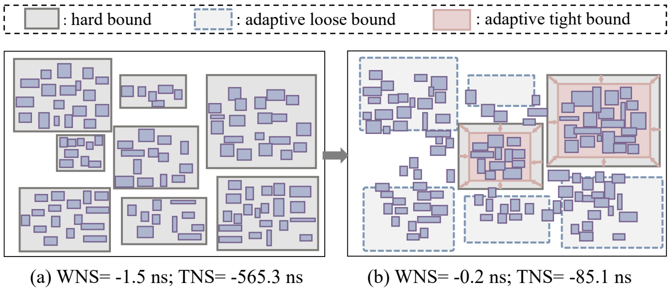

Fig. 2: **Motivation example of bounding timing groups.** (a) Uniform hard bounds are applied to all clusters. (b) Bounds are adaptively adjusted (tightened, loosened, or removed) based on each cluster's connection properties, resulting in improved timing performance.

selected clusters, and (iii) employs a Bayesian optimizer to iteratively refine the constraints by *re-selecting constrained clusters* (coarse-grained) and *adjusting the tightness of each bound* (fine-grained), based on PPA outcomes. The optimized constraints strike a better balance between intra- and inter-cluster connections for each individual cluster, ultimately guiding the analytical placer toward improved flat placement quality. As illustrated in Fig. 2 (b), clusters with strong global interconnections are adaptively relaxed or freed, while extremely timing-critical groups are more tightly bounded, resulting in improved timing performance. As a core component of this approach, we propose *eBound*, a novel placement bound model with quantified *constraint tightness* to enable ACCO. Specifically, *eBound* formulates placement constraint as an *electrostatic attractive force* and controls the *constraint tightness* by adjusting electric quantity.

We integrate ACCO with the *seeded placement* framework, which is used in [3], [9]. Leveraging TritonPart [2] for timing-aware clustering, our methods achieve up to 62% (average: 23%) and 97% (43%) improvements in post-route WNS and TNS, compared to the default OpenROAD flow. Additionally, our methods achieve up to 25% (7%) reduction in power. On the other hand, ACCO outperformed existing clustered methods, with gains of 42% (average: 21%), 96% (49%), 5% (2%) and 12% (3%) in WNS, TNS, area and power.

We summarize our major contributions as follows.

- We propose a novel PPA-driven placement paradigm based on mixed-grained ACCO, which tailors *constraint tightness* for each cluster to balance local and global interconnections, ultimately improving chip performance.
- We introduce *eBound*, the first placement bound modeling technique with quantified constraint control, enabling fine-grained adjustments and facilitating advanced placement optimization.
- We design an efficient Bayesian Optimization (BO)-based framework to optimize cluster constraints, directly improving PPA outcomes.
- Experimental results demonstrate that our methods significantly outperform existing methods in optimizing post-route PPA for diverse chip designs.

The remaining sections in this paper are organized as follows. Section II introduces the preliminaries. The eBound technique and ACCO-based placement is illustrated in Section III and Section IV. The experimental results are shown in Section V. Finally, Section VI concludes this paper.

II. Preliminaries

A. Analytical Placement Formulation

Analytical placement involves three main stages: global placement (GP), legalization (LG), and detailed placement (DP). GP is the most critical stage, determining the overall cell distribution by solving the following minimization problem:

$$\min_{\mathbf{x},\mathbf{y}} \sum_{e_j \in E} WL(\mathbf{x},\mathbf{y};e_j) + \lambda D(\mathbf{x},\mathbf{y}), \tag{1}$$

where $\mathbf{x}, \mathbf{y}$ are cell coordinates, WL is the wirelength cost (e.g., using LSE [12] or WA [13] models), and $D(\mathbf{x},\mathbf{y})$ is the density penalty. The penalty factor λ is gradually increased to ensure cells are spread out. This problem is typically solved using Nesterov's method [14], [15].

Electrostatic System Modeling. The ePlace/RePlAce [14], [16], [17] family of placers introduces a state-of-the-art density penalty, *eDensity*, by modeling the placement as a 2D electrostatic system. In this model, each instance is a positively charged particle with charge q_i proportional to its area A_i. The density penalty $D(\mathbf{x},\mathbf{y})$ becomes the system's total electric potential energy, and its gradient is the electric force on each cell. This high-quality formulation is significantly accelerated on GPUs in modern placers such as DREAMPlace [15].

B. Placement Bound

A placement bound is a *region constraint* that co-locates a group of standard cells to optimize PPA [18]. Common applications include managing power domains, minimizing clock skew [19], or improving timing and power for critical nets by reducing parasitic R and C. Bounds are classified by their enforcement strictness (*hard/soft*) for both member and non-member cells. For instance, a *fence* region imposes hard constraints on all cells, whereas a *guide* is entirely soft. Modern placers have been adapted to handle such constraints using various techniques, including look-ahead legalization [20], [21] and fence-aware density models [22]. More recently, the *Multiple Density Map* technique [23] has been shown to produce high-quality placement under region constraints. However, a critical limitation of existing models is their lack of quantifiable and flexible control over constraint strictness, hindering advanced optimization. To bridge this gap, we develop the *eBound* technique (Section III), which enables fine-grained and quantitative adjustment of *constraint tightness*.

III. eBound for Constrained Placement

This section details our *eBound* technique, which enables quantified constraint control for placement. We first formulate the constrained global placement problem, then review the *Multiple Density Map* technique [23] that forms the basis for our work, and finally present our *Multiple Electric Field* approach and the resulting parameterized optimization problem.

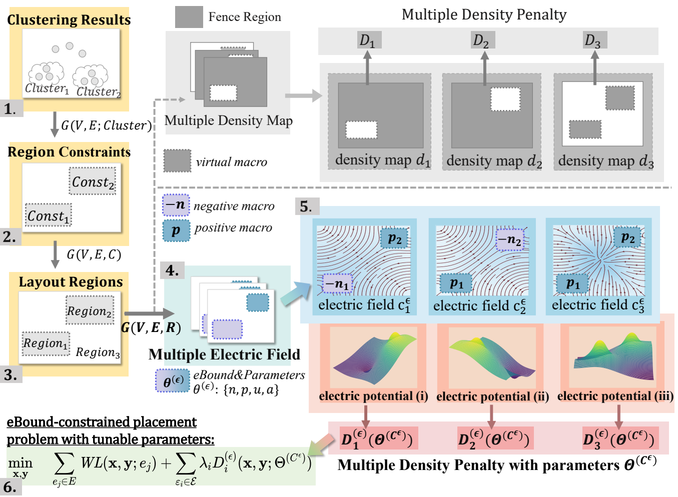

Fig. 3: **Illustration of the multiple density map technique and our multiple electric field technique with eBound.** We demonstrate how to formulate the eBound-constrained placement problem with tunable parameters. Please see Section III-B and III-C for details.

A. Problem Statement

A circuit of the VLSI placement problem with region constraints can be formulated as a hypergraph $G(V, E, C)$, where circuit instances are denoted by vertices V, nets are denoted by hyperedges E and region constraints are denoted by constraints C. Each region constraint c_i is defined by a box region r_i of $[r_{i_l}, r_{i_r}] \times [r_{i_b}, r_{i_t}]$ and a constraint type t_i (e.g., fence or guide as introduced in Section II-B). Formally, the global placement problem with region constraints can be viewed as a constrained optimization problem:

$$\begin{aligned} &\min_{\mathbf{x},\mathbf{y}} \sum_{e_j \in E} WL(\mathbf{x}, \mathbf{y}; e_j) + \lambda D(\mathbf{x}, \mathbf{y}), \\ &s.t. \quad \mathbf{x}, \mathbf{y} \in Const(r_i, t_i), \quad i = 1, 2, ..., |C|, \end{aligned} \tag{2}$$

where $Const(r_i, t_i)$ denotes the feasible solution space for c_i.

B. Multiple Density Map Technique

The *Multiple Density Map* technique employs multiple density penalties as soft constraints to address the constrained optimization problem 2, as shown in Fig. 3. Specifically, the entire placement area is divided into $|C|$+1 regions, which include all fence regions and a global region (the placement area excluding all fence regions). We denote those regions as R. Each region r_i corresponds to a unique density map, where infeasible areas are covered by full-density *virtual macros*. Each cell is added to its respective density map, and when a cell moves to an infeasible location, the overflow caused by *virtual macros* leads to a high density penalty D_i. This reformulates the constrained optimization problem 2 as:

$$\min_{\mathbf{x},\mathbf{y}} \sum_{e_j \in E} WL(\mathbf{x}, \mathbf{y}; e_j) + \sum_{r_i \in R} \lambda_i D_i(\mathbf{x}, \mathbf{y}), \tag{3}$$

where $D_i(\mathbf{x}, \mathbf{y})$ represents the density penalty for region r_i. After GP, a fence-aware legalization step is applied to finalize the placement. Despite achieving high-quality placement, this technique cannot achieve flexible constraint control. We develop our Bound technique based on this backbone.

C. eBound-Constrained Placement Formulation

In this subsection, we present our *eBound* modeling and *Multiple Electric Field* technique. For clarity, we denote eBound region constraints as C^ϵ. Inspired by the electrostatic system modeling, we define an eBound c_i^ϵ as a **negatively charged macro** with a surface charge density of -n_i to attract bounded cells and a **positively charged macro** with a surface charge density of p_i to repel other cells. The *constraint tightness* of c_i^ϵ is then quantified by parameters $\theta_i^{(\epsilon)}$ = $\{n_i, p_i, u_i, a_i\}$, where u_i (utilization) and a_i (aspect ratio) determine the shape of the eBound. Evidently, an eBound with a larger n imposes a tighter constraint on the bounded cells. As illustrated in Fig. 3, we establish $|C^\epsilon|$+1 electric fields, collectively denoted as $\mathcal{E}$. Each bound c_i^ϵ is negatively charged in its corresponding electric field ε_i and positively charged in all other fields. Cells are assigned to their respective electric fields, while cells from other fields are treated as *virtual cells*. These *virtual cells* are included in potential calculations but excluded from gradient backpropagation to avoid overlaps. This modeling allows us to formulate the *eBound-constrained placement problem* as follows:

$$\min_{\mathbf{x},\mathbf{y}} \sum_{e_j \in E} WL(\mathbf{x}, \mathbf{y}; e_j) + \sum_{\varepsilon_i \in \mathcal{E}} \lambda_i D_i^{(\epsilon)}(\mathbf{x}, \mathbf{y}; \Theta^{(C^\epsilon)}), \tag{4}$$

where $D_i^{(\varepsilon)}(\mathbf{x}, \mathbf{y}; \Theta^{(C^\epsilon)})$ represents the potential energy penalty of electric field ε_i, which is parameterized by constraint parameters $\Theta^{(C^\epsilon)}$ = $\{\theta_i^{(\epsilon)}\}_{i=1}^{|C^\epsilon|}$. We physically evaluate the accuracy of our eBound model for achieving quantified control of *constraint tightness* in Section V-D.

IV. PPA-driven Placement with ACCO

This section introduces our PPA-driven placement method through **A**daptive **C**luster **C**onstraints **O**ptimization (ACCO), which leverages eBound technique to obtain an optimized placement solution S_{opt} for post-route PPA. We provide an overview of our framework in Section IV-A and introduce our BO-based ACCO algorithm in Section IV-B.

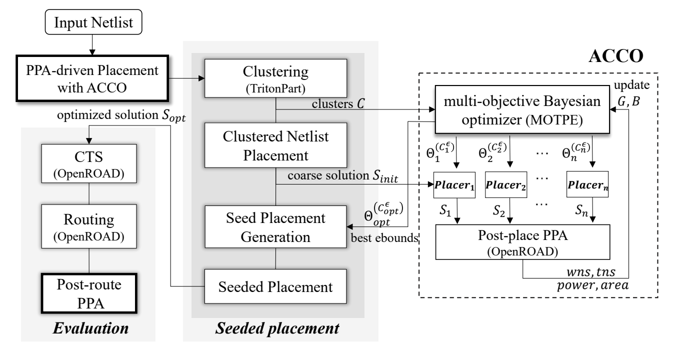

Fig. 4: **Our overall PPA-driven placement framework.** Leveraging eBound technique and ACCO algorithm, our approach can effectively optimize post-route PPA based on post-place PPA feedback.

Algorithm 1 Our overall approach

Input: $Netlist$ (.v, .lib, .lef, .def, .sdc), $|P|$ (number of paths), $Placer$
Output: WNS, TNS, NVP, $Power$, $Area$

/* Coarsening stage */
$P \leftarrow$ Extract top $|P|$ timing paths using OpenSTA
$C_t \leftarrow$ Run timing-aware clustering using TritonPart [24]
$Netlist_{clust} \leftarrow$ Generate clustered netlist from C_t
$S_{cluster} \leftarrow$ Run placement on $Netlist_{clust}$
/* Seed placement generation with ACCO */
$S_{init} \leftarrow$ Place instances in $Netlist$ at their respective cluster centers
C^{ϵ}_{opt},$\Theta^{(C^{\epsilon}_{opt})}_{opt} \leftarrow$ Generate eBound constraints using ACCO algorithm 2
$S_{seed} \leftarrow$ Build C^{ϵ}_{opt} with parameters $\Theta^{(C^{\epsilon}_{opt})}_{opt}$ on S_{init}
/* eBound-constrained seeded placement */
$S_{seeded} \leftarrow$ Run eBound-constrained placement on S_{seed} using $Placer$
Remove constraints
$S_{opt} \leftarrow$ Run detailed placement using OpenROAD
/* Placement evaluation */
Run CTS and Routing on S_{opt}
WNS, TNS, NVP, $Power$, $Area \leftarrow$ Record post-route worst negative slack, total negative slack, count of negative violating paths, power and cell area from OpenROAD
return WNS, TNS, NVP, $Power$, $Area$

Algorithm 2 ACCO algorithm

Input: S_{init}, $Placer$, $\mathcal{C}$ (search space for $\Theta^{(C^{\epsilon})}$, constructed by clustered results $Netlist_{cluster}$), N_i (number of iterations), N_{min} (minimal number to build a model), ρ (fraction of random runs to encourage exploration)
Output: best $\Theta^{(C^{\epsilon}_{opt})}_{opt}$

/* Optimization setup */
$D \leftarrow$ Initialize observations as $\emptyset$
for each i from 1 to N_i **do**
  /* MOTPE-based sampling */
  if $|D| < N_{min}$ **then**
    $\Theta^{(C^{\epsilon}_i)}_i \leftarrow$ Random sampling candidate from $\mathcal{C}$
  else if $rand() < \rho$ **then**
    $\Theta^{(C^{\epsilon}_i)}_i \leftarrow$ Random sampling candidate from $\mathcal{C}$
  else
    $\Theta^{(C^{\epsilon}_i)}_i \leftarrow$ Sampling from $\mathcal{C}$ based on D using MOTPE
  end if
  /* Evaluation of samples */
  $S_i \leftarrow$ Build eBound constraints C^{ϵ}_i with $\Theta^{(C^{\epsilon}_i)}_i$ on S_{init}
  $S(\Theta^{(C^{\epsilon}_i)}_i) \leftarrow$ Run eBound-constrained placement on S_i using $Placer$
  $r^{(i)} \leftarrow$ Record post-place PPA $wns, tns, area, power$ from OpenROAD
  $D \leftarrow D \cup (\Theta^{(C^{\epsilon}_i)}_i, r^{(i)})$
end for
/* Determine optimal constraints */
$\mathcal{F} \leftarrow$ Pareto front of D
$\Theta^{(C^{\epsilon}_{opt})}_{opt} \leftarrow$ Best average score in $\mathcal{F}$
return Optimized eBound constraints C^{ϵ}_{opt} with $\Theta^{(C^{\epsilon}_{opt})}_{opt}$

A. Overview of Our Approach

Fig. 4 depicts the overall flow of our framework. Our placement method is based on the *seeded placement* framework [3], [9], where a *seed placement* of clusters is used to induce *seed locations* of instances. The complete flow is formally presented in Algorithm 1. Our placement approach identifies clustering level for a given design, performs mixed-grained cluster-constraints level optimization (ACCO) and provides PPA-aware guidance towards final flat placement.

B. ACCO Framework for PPA Optimization

We first present the general formulation of PPA optimization problem and reformulate it as a black-box optimization problem using eBound technique. Then, we introduce the ACCO algorithm, based on Bayesian Optimization (BO), to efficiently and effectively perform mixed-grained PPA optimization.

1) PPA Optimization Problem Formulation: Given the EDA tools, the PPA metrics are determined by the netlist $G(V, E)$ and the placement solution S, which can be modeled as a function $\mathcal{P} : \mathcal{D} \rightarrow \mathbb{R}^n$ of $(S, G) \in \mathcal{D}$, where $\mathcal{D}$ represents the design space. Without loss of generality, we assume that, for all metrics, a lower value indicates better performance. The PPA optimization problem can then be expressed as:

$$\min_{S,G \in \mathcal{D}} \quad \mathcal{P}\left(S, G\right) \tag{5}$$

The objective of PPA-driven placement is to produce the optimal placement solution S_{opt} to minimize $\mathcal{P}$ for a given netlist G. However, since $\mathcal{P}$ can be conceptualized as an underlying "black box" to designers that can only be accurately evaluated by EDA tools, analytical placement algorithms are unable to directly optimize Equation 5, often leading to suboptimal performance. To achieve effective optimization, our ACCO framework leverages eBound constraints to guide the analytical placer and optimize the constraint parameters directly based on the PPA metrics reported by EDA tools. Specifically, given clustering results C, ACCO selects clusters in C to construct eBound constraints C^{ϵ} with parameters $\Theta^{(C^{\epsilon})}$. To achieve flexible cluster selection, we introduce a binary variable (1 or 0) into each parameter $\theta^{(\epsilon)}_i$, which indicates whether a cluster is bounded. An analytical placer then performs the eBound-constrained placement (Equation 4) to obtain a solution $S = S(\Theta^{(C^{\epsilon})})$. Thus, the optimization objective of ACCO takes the form of:

$$\underset{\Theta^{(C^{\epsilon})} \in \mathcal{C}}{\operatorname{argmin}} \quad \mathcal{P}(S(\Theta^{(C^{\epsilon})}), G), \tag{6}$$

where $\mathcal{C}$ denotes the feature domain of cluster constraints.

Remark 1: Constraints as a Search Space. Notably, *the eBound technique essentially introduces a cluster constraints search space $\mathcal{C}$ for PPA optimization*, while PPA-aware clustering and PPA-benefits of placement bound ensure that exploration within this space can provide high-quality placement guidance for analytical placer, which is also documented in Section V-B and Section V-E below. In contrast, previous approaches that rely on unified, rule-based cluster constraints can be viewed as manual selections within $\mathcal{C}$, often leading to suboptimal results (see Section V-B).

2) BO-based ACCO algorithm: Effective exploration of the mixed continuous-discrete parameter space $\mathcal{C}$ is essential for solving the black-box optimization problem 6. To address this, we adopt Multi-objective Tree-structured Parzen Estimator (MOTPE) [25], an advanced Bayesian optimizer for multi-objective optimization, which naturally supports mixed continuous and discrete parameter spaces. MOTPE utilizes a kernel density estimator to construct distributions of the "good" (G) and "poor" samples (B) based on previous evaluations, and selects the most promising candidate with the highest expected PPA improvement by maximizing $\frac{G(\Theta)}{P(\Theta)}$. Specifically, in each search iteration i, the optimizer samples $\Theta^{(C^{\epsilon}_i)}_i$ from the search space $\mathcal{C}$ and evaluates it using post-place PPA metrics of $S(\Theta^{(C^{\epsilon}_i)}_i)$. This feedback is then applied to update the models G and B, refining the search process for subsequent iterations. Algorithm 2 formally describes our ACCO framework.

Remark 2: Flow Efficiency and Design Closure. ACCO embodies the "shift-left" design philosophy [26]–[30] by ad-

TABLE I

COMPARISON OF POST-ROUTE PPA METRICS—WNS(ns), TNS(ns), NVP, POWER(W) AND AREA(mm^2)—FOR GLOBAL PLACEMENT DERIVED BY DIFFERENT APPROACHES. A LOWER VALUE FOR NVP, POWER AND AREA IS BETTER, WHILE A HIGHER VALUE FOR WNS, TNS IS BETTER. WE MARK THE BEST RESULTS IN **BOLD**, AND WE MARK THE SECOND BEST RESULTS IN BROWN.

Method	Metric	Benchmark						
		aes	jpeg	bp_be	ariane	BlackParrot	toygpu	mor1kx
Comparison with Analytical Approaches								
DREAMPlace	WNS (ns)	-0.128	-0.598	-0.771	-0.478	-3.289	-1.784	-0.581
	TNS (ns)	-9.39	-628.33	-688.41	-617.98	-83.03	-159.06	-820.45
	# NVP	166	2405	1296	1302	702	101	2010
	Power (W)	0.268	0.831	0.308	0.289	0.138	0.521	1.184
	Area (mm^2)	0.0298	0.103	0.112	0.363	0.487	0.988	1.298
DREAMPlace4.0	WNS (ns)	-0.131	-0.571	-0.671	-0.491	-2.171	-1.417	-0.493
	TNS (ns)	-8.97	-601.33	-481.92	-605.78	-65.09	-127.35	-690.11
	# NVP	163	2313	1192	2386	655	114	1606
	Power (W)	0.265	0.844	0.308	0.289	**0.132**	0.522	1.098
	Area (mm^2)	0.0295	0.106	0.113	0.369	0.451	0.993	1.301
OpenROAD	WNS (ns)	-0.126	-0.553	-0.572	-0.492	-2.166	-1.625	-0.565
	TNS (ns)	-10.36	-597.16	-502.31	-609.21	-92.18	-144.91	-783.12
	# NVP	**162**	2546	1287	1871	729	101	2010
	Power (W)	**0.262**	1.059	0.307	0.358	0.137	0.516	1.021
	Area (mm^2)	**0.0293**	0.111	0.112	0.354	0.485	0.985	1.413
Comparison with Clustered Approaches								
[9]	WNS (ns)	-0.118	-0.597	-0.716	-0.504	-2.017	-1.008	-0.489
	TNS (ns)	-8.67	-574.61	-671.91	-604.66	-215.80	-73.31	-768.34
	# NVP	**162**	1202	1350	2276	1217	100	1871
	Power (W)	0.268	0.820	0.314	0.275	0.143	0.519	1.131
	Area (mm^2)	0.0305	0.103	0.111	0.365	0.490	0.979	1.271
[9] with fence	WNS (ns)	-0.217	-0.597	-1.001	-0.828	-2.445	-1.689	-0.891
	TNS (ns)	-14.73	-612.71	-1042.99	-1058.90	-2.758	-814.68	-1131.31
	# NVP	166	2272	2142	3014	**11**	193	2312
	Power (W)	0.272	0.829	0.345	0.360	0.140	0.528	1.003
	Area (mm^2)	0.0299	0.101	0.116	0.365	0.494	1.029	1.313
Our PPA-driven Placement Approach								
[9] with ACCO	WNS (ns)	**-0.106**	**-0.549**	**-0.579**	**-0.291**	**-1.956**	**-0.613**	**-0.323**
	TNS (ns)	**-8.39**	**-481.76**	**-391.98**	**-239.38**	**-2.54**	**-35.86**	**-351.13**
	# NVP	**162**	**1158**	**1089**	**1866**	29	**98**	**1014**
	Power (W)	0.266	**0.805**	**0.305**	**0.269**	0.143	**0.515**	**0.997**
	Area (mm^2)	0.0296	**0.102**	**0.111**	**0.345**	**0.484**	**0.976**	**1.251**

dressing PPA challenges during global placement. Leveraging early, fast PPA feedback, ACCO generates physical-aware layouts, thus reducing the burden on downstream stages like CTS and routing. Our method mitigates the need for costly, time-consuming optimization and repair iterations, thereby improving overall design turn-around time (TAT) and promoting faster design closure, as empirically validated in Section V-C.

V. EXPERIMENTAL RESULTS

This section experimentally validates our PPA-driven placement approach through a series of evaluations. We detail the setup (Section V-A), compare post-route PPA against other methods (Section V-B), analyze runtime (Section V-C), and conduct a physical evaluation of the eBound model (Section V-D). We also present extensive ablation studies in Section V-E to isolate the contributions of each key component.

A. Experimental Setup

We implement eBound modeling on top of the open-source analytical placer DREAMPlace [31], use TritonPart [24] for clustering, and employ an MOTPE-based [32] optimizer for ACCO flow. All experiments are conducted on a server with a 3.20 GHz Intel Xeon E5-2667 v4 processor and a NVIDIA GeForce RTX 2080 Ti GPU. We evaluate our method on seven publicly available designs from the widely-used OpenROAD-Project [33] and OpenRISC [34] repositories, across diverse domains such as processor cores, graphics, and cryptography, all synthesized using the NanGate45 technology [35]. Benchmark statistics are detailed in Table II. Placement quality is assessed using post-route PPA metrics derived from the open-source OpenROAD flow [33], including worst negative slack (WNS), total negative slack (TNS), count of negative violating paths (NVP), Power, and cell area (Area). All evaluation results are validated under five different random seeds with standard deviations negligible compared to performance gap. We omit the wirelength results, which are primarily optimized by the term $WL(\mathbf{x}, \mathbf{y})$ and present comparable quality for all methods. To be noted, our method achieves PPA improvement without degradation in wirelength.

TABLE II

SPECIFICATIONS OF BENCHMARK.

Design (NG45)	#Insts	#Nets	#Pins	*TCP
aes	15046	15430	59126	0.82
jpeg	55940	71522	183584	1.70
bp_be	50891	58428	182949	5.40
ariane	168683	184856	592261	4.00
BlackParrot	301054	333364	984093	6.84
toygpu	368081	466513	1399167	2.00
mor1kx	2158270	2781678	9421072	10.00

*TCP(ns) denotes the target clock periods used in the OpenROAD flow.

B. Main Results

We compare our method, denoted as "PPA-driven placement", with (i) analytical placers and (ii) clustered placement frameworks. For ACCO process, we set N_i = 600, N_{min} = 100, ρ = 1/6, and utilize 12 parallel processes. The search space for eBound parameters is defined as: $n \in [0.0, 4.0]$, $p \in [0.0, 1.5]$, $u \in [0.4, 0.85]$, $a \in [0.6, 1.5]$, and a binary selection variable $s \in \{0, 1\}$. $|P|$ is set as 1000 in our experiments to demonstrate effectiveness by studying three designs: aes, jpeg and ariane. For a fair comparison, the timing-driven mode of DREAMPlace in our framework is disabled, using vanilla DREAMPlace engine [15]. Table I summarizes the post-route PPA results. Our approach consistently outperforms all baselines, achieving the best WNS and TNS in all cases and superior results for NVP, Power, and Area in most instances.

Comparison with Analytical Placement. We compare against

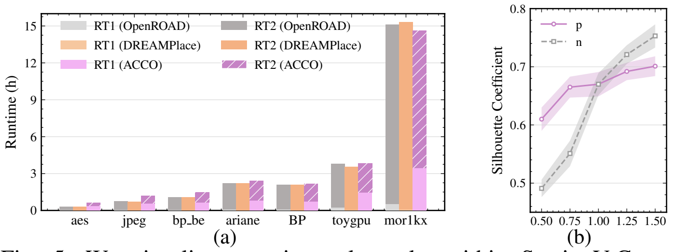

Fig. 5: We visualize experimental results within SectionV-C and Section V-E: **(a) overall runtime analysis**. ACCO reduces subsequent flow runtime (RT2), especially for large designs. **(b) eBound physical validation.** The Silhouette Coefficient shows a strong correlation with eBound parameters, validating its quantified control.

the default OpenROAD flow (using timing-driven RePlace [36]), vanilla DREAMPlace [15], and timing-driven DREAMPlace4.0 [37]. As shown in the results, our method achieves significant PPA improvements over even the timing-driven placers, with maximum (average) gains of 62% (23%) in WNS, 97% (38%) in TNS, and 96% (32%) in NVP relative to the best-performing baseline for each metric. Furthermore, our method improves Power by up to 25% (7%). The advantages are particularly pronounced on large-scale circuits like `ariane`, `BlackParrot`, `toygpu`, and `mor1kx`.

Comparison with Clustered Placement. We also compare against two established clustered placement frameworks: soft-constrained seeded placement from [9] and its variant from [3] which applies uniform fence regions to all clusters. This comparison directly demonstrates the necessity and effectiveness of our adaptive constraint optimization, validating Remark 1. Given identical timing-aware clustering results from TritonPart [24], our ACCO flow unlocks the full potential of the clusters. Specifically, it achieves substantial gains of 42% (avg. 21%) in WNS, 96% (48%) in TNS, and 49% (31%) in NVP. Moreover, the PPA-directed optimization also yields improvements of up to 11% in Power and 5% in Area. These results highlight the efficacy of our approach in adaptively balancing intra- and inter-cluster interactions to enhance overall placement quality.

C. Runtime&Scalability Analysis

We present the overall flow runtime in Fig. 5a, breaking it down into the placement/optimization (**RT1**, which includes multiple trials for ACCO) and the subsequent stages (**RT2**, which includes DP, CTS, and Routing). Our ACCO framework embodies the "shift-left" principle by investing more runtime in the early placement stage to improve quality and accelerate design closure. While the optimization search increases RT1, this proactive optimization yields a significant reduction in the computationally expensive RT2, particularly for large designs under tight timing constraints. For instance, on the large-scale `mor1kx` benchmark, ACCO reduces the post-placement runtime (RT2) by 23%, leading to a 4% reduction in total flow time compared to the default OpenROAD flow. This trend, combined with the scalability of GPU-accelerated DREAMPlace, demonstrates ACCO's potential for efficiently handling large-scale, industrial designs. This result empirically validates Remark 2, confirming that early PPA optimization can enhance overall physical design productivity.

TABLE III

ABLATION STUDY RESULTS OF FOUR REPRESENTATIVE CIRCUITS. WE MARK THE BEST RESULTS IN **BOLD** AND THE SECOND BEST IN BROWN.

Design	Setting	Post-route PPA metrics				
		WNS	TNS	NVP	Power	Area
`jpeg`	*Random1*	-0.553	-482.41	1301	0.817	0.102
	Random2	-0.623	-483.61	1986	0.837	0.102
	Uniform	-0.617	-509.82	2061	0.831	0.103
	Shape	-0.625	-499.39	1876	0.807	0.102
	Ours	**-0.549**	**-481.76**	**1158**	**0.805**	**0.102**
`bp_be`	*Random1*	-0.588	-401.33	1119	0.311	0.121
	Random2	-0.881	-729.17	2071	0.321	0.119
	Uniform	-0.701	-661.20	1332	0.323	0.123
	Shape	-0.625	-512.19	1199	0.319	-0.117
	Ours	**-0.579**	**-391.98**	**1089**	**0.305**	**0.111**
`BlackParrot`	*Random1*	-2.025	-3.88	**21**	0.141	-0.491
	Random2	-3.231	-211.98	783	0.147	0.517
	Uniform	-2.133	-91.34	551	**0.137**	0.488
	Shape	-2.102	-33.65	113	0.139	0.501
	Ours	**-1.956**	**-2.54**	29	0.143	**0.484**
`mor1kx`	*Random1*	-0.378	-377.13	1123	1.038	1.301
	Random2	-0.691	-1401.04	2013	1.188	1.306
	Uniform	-0.465	-794.28	1486	1.105	1.288
	Shape	-0.401	-422.32	1465	1.012	1.299
	Ours	**-0.323**	**-351.13**	**1014**	**0.997**	**1.251**

D. Physical Evaluation of eBound

To physically validate the quantified control of our force-based eBound model, we conduct experiments on the `aes` design, which is partitioned into two clusters. We fix the parameters of one eBound ($n = 1, p = 1$) and vary either the attractive charge (n) or the repulsive charge (p) of the second eBound, keeping its other parameters constant. We measure the clustering quality using the *silhouette coefficient* [38], a metric that evaluates the balance between intra-cluster cohesion and inter-cluster separation, where higher values indicate better-defined clusters. As depicted in Fig. 5b, the silhouette coefficient shows positive correlation with eBound charge parameters. This result confirms that our model provides accurate, fine-grained control over the physical *constraint tightness*.

E. Ablation Study

We conduct extensive ablation studies to isolate the contributions of our key components. We evaluate several alternative settings: (i) **Random1**, which replaces MOTPE with a random search; (ii) **Random2**, which uses direct random parameter assignments; (iii) **Uniform**, which applies a fixed, uniform eBound to all selected clusters; and (iv) **Shape**, which only optimizes the eBound's shape parameters while keeping charges fixed. We present results of four representative circuits in Table III. The results consistently demonstrate the superiority of our full approach. The performance gap between our method and *Random1* and *Uniform* settings validates Remark 1 by showing the necessity of systematic constraint optimization over naive or fixed strategies. Furthermore, the suboptimal results from the *Random2* and *Shape* settings confirm the efficacy of both the eBound's force-based parameterization and the BO-driven ACCO framework in navigating the complex search space.

VI. CONCLUSION

We have developed a PPA-driven placement paradigm featuring the *eBound* technique and the ACCO framework. By performing MOTPE-based, mixed-grained optimization of cluster constraints, our method improves post-route PPA and demonstrates flow efficiency. The *eBound* model introduces an advanced optimization space. While BO-based search demonstrates effectiveness, we believe future work can develop predictive models, potentially leveraging machine learning, to directly infer optimal eBound constraints from design features, thus creating faster and generalizable placement guidance.

REFERENCES

[1] Y.-C. Lu, T. Yang, S. K. Lim, and H. Ren, "Placement optimization via ppa-directed graph clustering," in *2022 ACM/IEEE 4th Workshop on Machine Learning for CAD (MLCAD)*, 2022, pp. 1–6.

[2] I. Bustany, G. Gasparyan, A. B. Kahng, I. Koutis, B. Pramanik, and Z. Wang, "An open-source constraints-driven general partitioning multi-tool for vlsi physical design," in *2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, 2023, pp. 1–9.

[3] A. Kahng, S. Kang, S. Kundu, K. Min, S. Park, and B. Pramanik, "Ppa-relevant clustering-driven placement for large-scale vlsi designs," in *Proceedings of the 61st ACM/IEEE Design Automation Conference*, 2024, pp. 1–6.

[4] J.-M. Lin, Y.-Y. Chang, and W.-L. Huang, "Timing-driven analytical placement according to expected cell distribution range," in *Proceedings of the 2024 International Symposium on Physical Design*, ser. ISPD '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 177–184.

[5] W.-J. Sun and C. Sechen, "Efficient and effective placement for very large circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 14, no. 3, pp. 349–359, 1995.

[6] B. Hu and M. Marek-Sadowska, "Fine granularity clustering-based placement," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 23, no. 4, pp. 527–536, 2004.

[7] G.-J. Nam, S. Reda, C. Alpert, P. Villarrubia, and A. Kahng, "A fast hierarchical quadratic placement algorithm," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, no. 4, pp. 678–691, 2006.

[8] N. Viswanathan, M. Pan, and C. Chu, "Fastplace 3.0: A fast multilevel quadratic placement algorithm with placement congestion control," in *2007 Asia and South Pacific Design Automation Conference*, 2007, pp. 135–140.

[9] M. Fogaça, A. B. Kahng, E. Monteiro, R. Reis, L. Wang, and M. Woo, "On the superiority of modularity-based clustering for determining placement-relevant clusters," *Integration*, vol. 74, pp. 32–44, 2020.

[10] Y.-L. Chuang, G.-J. Nam, C. J. Alpert, Y.-W. Chang, J. Roy, and N. Viswanathan, "Design-hierarchy aware mixed-size placement for routability optimization," in *2010 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2010, pp. 663–668.

[11] J.-M. Lin, S.-T. Li, and Y.-T. Wang, "Routability-driven mixed-size placement prototyping approach considering design hierarchy and indirect connectivity between macros," in *2019 56th ACM/IEEE Design Automation Conference (DAC)*, 2019, pp. 1–6.

[12] T.-C. Chen, Z.-W. Jiang, T.-C. Hsu, H.-C. Chen, and Y.-W. Chang, "Ntuplace3: An analytical placer for large-scale mixed-size designs with preplaced blocks and density constraints," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, no. 7, pp. 1228–1240, 2008.

[13] M.-K. Hsu, V. Balabanov, and Y.-W. Chang, "Tsv-aware analytical placement for 3-d ic designs based on a novel weighted-average wirelength model," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 32, no. 4, pp. 497–509, 2013.

[14] J. Lu, P. Chen, C.-C. Chang, L. Sha, D. J.-H. Huang, C.-C. Teng, and C.-K. Cheng, "eplace: Electrostatics-based placement using fast fourier transform and nesterov's method," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 20, no. 2, Mar. 2015.

[15] Y. Lin, Z. Jiang, J. Gu, W. Li, S. Dhar, H. Ren, B. Khailany, and D. Z. Pan, "Dreamplace: Deep learning toolkit-enabled gpu acceleration for modern vlsi placement," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 40, no. 4, pp. 748–761, 2021.

[16] J. Lu, H. Zhuang, P. Chen, H. Chang, C.-C. Chang, Y.-C. Wong, L. Sha, D. Huang, Y. Luo, C.-C. Teng, and C.-K. Cheng, "eplace-ms: Electrostatics-based placement for mixed-size circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 5, pp. 685–698, 2015.

[17] C.-K. Cheng, A. B. Kahng, I. Kang, and L. Wang, "Replace: Advancing solution quality and routability validation in global placement," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 38, no. 9, pp. 1717–1730, 2019.

[18] Synopsys, *IC Compiler II Implementation User Guide - Version J-2014.09-SP4*, Synopsys, Inc., March 2015, https://docslib.org/doc/10984866/ic-compiler-ii-implementation-user-guide.

[19] I. S. Bustany, D. Chinnery, J. R. Shinnerl, and V. Yutsis, "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*, ser. ISPD '15. New York, NY, USA: Association for Computing Machinery, 2015, p. 157–164.

[20] N. K. Darav, A. Kennings, A. F. Tabrizi, D. Westwick, and L. Behjat, "Eh? placer: A high-performance modern technology-driven placer," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 21, no. 3, pp. 1–27, 2016.

[21] N. K. Darav, A. Kennings, D. Westwick, and L. Behjat, "High performance global placement and legalization accounting for fence regions," in *2015 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2015, pp. 514–519.

[22] C.-C. Huang, H.-Y. Lee, B.-Q. Lin, S.-W. Yang, C.-H. Chang, S.-T. Chen, Y.-W. Chang, T.-C. Chen, and I. Bustany, "Ntuplace4dr: A detailed-routing-driven placer for mixed-size circuit designs with technology and region constraints," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 3, pp. 669–681, 2018.

[23] Y. Guo, Y. Wu, Z. Huang, and J. Chen, "Complex placement region handling based on electrostatic system modeling," in *2020 IEEE 15th International Conference on Solid-State & Integrated Circuit Technology (ICSICT)*, 2020, pp. 1–3.

[24] Tritonpart: An open-source constraints-driven partitioner. [Online]. Available: https://github.com/The-OpenROAD-Project/OpenROAD/tree/master/src/par

[25] Y. Ozaki, Y. Tanigaki, S. Watanabe, and M. Onishi, "Multiobjective tree-structured parzen estimator for computationally expensive optimization problems," in *Proceedings of the 2020 genetic and evolutionary computation conference*, 2020, pp. 533–541.

[26] L. Chen, Y. Chen, Z. Chu, W. Fang, T.-Y. Ho, Y. Huang, S. Khan, M. Li, X. Li, Y. Liang *et al.*, "The dawn of ai-native eda: Promises and challenges of large circuit models," *arXiv e-prints*, pp. arXiv–2403, 2024.

[27] Z. Geng, J. Wang, Z. Liu, S. Xu, Z. Tang, S. Kai, M. Yuan, H. Jianye, and F. Wu, "Lamplace: Learning to optimize cross-stage metrics in macro placement," in *The Thirteenth International Conference on Learning Representations*.

[28] Z. Wang, Z. Geng, Z. Tu, J. Wang, Y. Qian, Z. Xu, Z. Liu, S. Xu, Z. Tang, S. Kai *et al.*, "Benchmarking end-to-end performance of ai-based chip placement algorithms," *arXiv preprint arXiv:2407.15026*, 2024.

[29] A. Agnesina, P. Rajvanshi, T. Yang, G. Pradipta, A. Jiao, B. Keller, B. Khailany, and H. Ren, "Autodmp: Automated dreamplace-based macro placement," in *Proceedings of the 2023 International Symposium on Physical Design*, 2023, pp. 149–157.

[30] Z. Geng, J. Wang, Z. Liu, S. Xu, Z. Tang, M. Yuan, J. Hao, Y. Zhang, and F. Wu, "Reinforcement learning within tree search for fast macro placement," in *Forty-first International Conference on Machine Learning*.

[31] Open source dreamplace repository. [Online]. Available: https://github.com/limbo018/DREAMPlace.git

[32] J. Guerrero-Viu, S. Hauns, S. Izquierdo, G. Miotto, S. Schrodi, A. Biedenkapp, T. Elsken, D. Deng, M. Lindauer, and F. Hutter, "Bag of baselines for multi-objective joint neural architecture search and hyperparameter optimization," *arXiv preprint arXiv:2105.01015*, 2021.

[33] Openroad-flow-scripts. [Online]. Available: https://github.com/The-OpenROAD-Project/OpenROAD-flow-scripts

[34] Openrisc. mor1kx. [Online]. Available: https://github.com/openrisc/mor1kx.git

[35] Open-source nangate45 pdk enablement (integrated into openroad flow). [Online]. Available: https://github.com/The-OpenROAD-Project/OpenROAD-flow-scripts/tree/master/flow/platforms/nangate45

[36] Openroad global placement module, based on the replace tool. [Online]. Available: https://github.com/The-OpenROAD-Project/OpenROAD/tree/master/src/gpl

[37] P. Liao, S. Liu, Z. Chen, W. Lv, Y. Lin, and B. Yu, "Dreamplace 4.0: Timing-driven global placement with momentum-based net weighting," in *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2022, pp. 939–944.

[38] L. Kaufman and P. J. Rousseeuw, *Finding groups in data: an introduction to cluster analysis*. John Wiley & Sons, 2009.