OpenPARF 3.0: Robust Multi-Electrostatics Based FPGA Macro Placement Considering Cascaded Macros Groups and Fence Regions

Jing Mai[1,2], Jiarui Wang[1,2], Yifan Chen[2], Zizheng Guo[2], Xun Jiang[2], Yun Liang[2], Yibo Lin[2*]
[1]School of Computer Science, Peking University [2]School of Integrated Circuits, Peking University
{jingmai,jiaruiwang,chenyifan2019,gzz}@pku.edu.cn, xunjiang@stu.pku.edu.cn,
{ericlyun,yibolin}@pku.edu.cn

***Abstract*—FPGA macro placement exerts a significant influence on routability and timing closure in FPGA physical design. Macros could subject to cascaded macro constraints and necessitate placement in contiguous sites. Meanwhile, instances could also subject to fence region constraints, permitting placement within designated areas. Such kind of heterogeneity exacerbates the solution space discontinuity and leads to divergence and local optima entrapment.**

In this work, we propose a robust multi-electrostatics-based FPGA macro placer OpenPARF 3.0 that can handle the aforementioned constraints efficiently. We adopt a novel multi-electrostatics region model to handle the fence region discontinuity and propose a divergence-aware density weight scheduling scheme that can address the robustness issues effectively. Experimental results demonstrate that our proposed framework can address robustness issues effectively and outperform state-of-the-art placers in both quality and efficiency.

***Index Terms*—FPGA, Placement, Macro Placement, Cascaded Macro, Fence Region**

I. Introduction

FPGA macro placement significantly influences the ultimate performance of FPGA designs. Macros on FPGAs, such as DSP and BRAM, offer efficient functionalities such as digital signal processing and on-chip storage, surpassing the mere utilization of LUTs and FFs. Hence, macros are widely employed in high-performance FPGA designs such as IP blocks, and play a crucial role in the overall performance and routability.

FPGA macro placement has two major challenges that arise from macro heterogeneity and fence regions. The heterogeneity stems from larger size variations, sparser feasible sites, higher routing resources demands, and the tailored cascaded shapes to achieve enhanced performance. Furthermore, clock routing architectures or user-defined fence region constraints disrupt the solution space continuity for FPGA placement, thereby weakening the effectiveness of analytical placement algorithms that are primarily based on continuous optimization.

In recent years, the emergence of large-scale academic FPGA placement benchmarks [1]–[3] has led to the development of numerous placement algorithms [4]–[19]. These algorithms, successfully applied to the aforementioned large-scale benchmarks, predominantly employ analytical placement techniques. Among them, state-of-the-art performance is achieved by nonlinear placement algorithms, including the esteemed ePlace-series [12]–[14] and NTUPlace-series [16] algorithms. However, it is worth noting that these benchmarks oversimplify the consideration of macros, disregarding the intricate nature of cascaded macros in real-world scenarios. Furthermore, the region constraints specified prior to placement are conspicuously absent in these benchmarks. Unlike fence regions generated as intermediates during the placement process, as exemplified in [5], [8], the region constraints specified before placement pose greater challenges, as they are prone to routing congestion and divergence, thus elevating the difficulty level.

In this work, we tackle the FPGA macro placement considering considering both fence region constraints and cascaded macro constraints based on a multi-electrostatics-based placement framework.

*Corresponding author

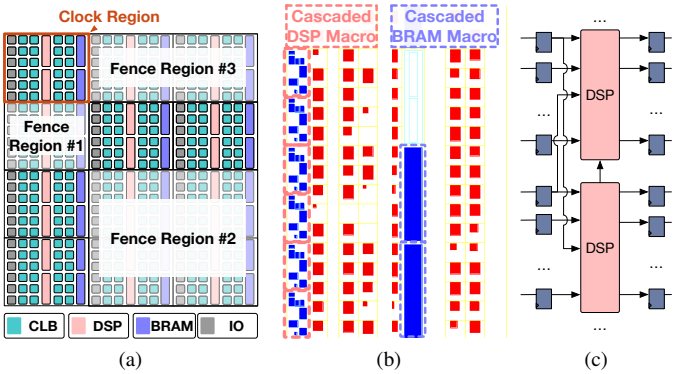

Fig. 1: (a) A simplified columnar FPGA architecture depiction for Xilinx Ultrascale+ xcvu3p series. (b) An example of cascaded DSP macros and cascaded BRAM macros from a Vivado GUI screenshot (color inverted). (c) An example of cascaded macro group, including cascaded DSP macros and the closely connected FFs.

We propose a novel footprint compression technique under multi-electrostatics region models and are capable of addressing the robustness issues effectively. Our major contributions are summarized as follows.

- We present a novel model called the multi-electrostatics region model, accompanied by a footprint compression technique, to effectively handle the discontinuity in the fence region.
- We propose a divergence-aware density weight scheduling scheme, which effectively addresses robustness issues.
- To address the imbalance in the size of the cascaded macros, we introduce a cascaded macro shredding technique.
- Experiments on the MLCAD 2023 FPGA macro placement benchmark demonstrate an improvement of 27.8%, 6.9%, 13.3%, and 49.1% compared to the winners of the Vivado and MLCAD 2023 contest, respectively. Our placer also supports GPU acceleration and can achieve 1.81-3.18× speedup over the baselines.

The rest of the paper is organized as follows. Section II introduces the preliminary knowledge of the FPGA architecture and multi-electrostatics-based FPGA placement. Section III describes the core placement algorithms. Section IV shows the experimental results, followed by the conclusion in Section V.

II. Preliminaries

In this section, we will introduce the targeted FPGA architecture, especially focusing on the fence region constraint and the cascaded macro constraint. We will also introduce the concept of multi-electrostatics-based FPGA placement.

A. FPGA Architecture

In this work, we target the Xilinx Ultrascale+ xcvu3p series as a representative architecture, as illustrated in Figure 1(a) [1]. A simplified

[1]Figure 1(a) only contains part of the whole 6×5 clock regions, but is sufficient for illustration.

version of this architecture is also used in the MLCAD 2023 FPGA Macro placement contest with a subset of instance types, i.e., {LUT, FF, DSP, BRAM, IO} [20].

Due to the significantly larger sizes of DSPs and BRAMs compared to LUTs and FFs, DSPs and BRAMs are considered as *macros*. On the other hand, LUTs and FFs are considered as *standard cells*. FPGA macros significantly impact the overall routability of the design [20]. Furthermore, FPGA placement necessitates adherence to fence region constraints and cascaded shape constraints, which will be elaborated upon in the subsequent subsections.

1) Fence Region (FR) Constraint: Fence region constraints may stem from clock regions or can be user-defined. The fence region constraint can be abstracted as follows, and as illustrated in Figure 1(a).

Definition 1 (Fence Region Constraint)**.** Given a set of instances and a feasible region on the FPGA layout (usually a rectangle), in the final placement result, the instances within the set must be placed inside the feasible region.

Note that multiple fence region constraints can exist in a single FPGA placement task, and instances not assigned to any fence region constraint can be placed on any feasible site on the FPGA layout.

2) Cascaded Macro (CM) Constraint: For the sake of enhanceing performance and reducing routing resource overload, it is necessary for FPGA placement to meticulously arrange a number of macro instances in close proximity, thereby forming a *cascaded macro* as illustrated in Figure 1(b). A cascaded macro constraint can be defined as follows.

Definition 2 (Cascaded Macro Constraint)**.** Given a set of macro instances (typically of the same type, such as DSPs or BRAMs), in the final placement result, the instances within the set must be placed in continuous columnated sites in a prescribed sequence.

3) Cascaded Macro Group: Cascaded macros mainly include DSP macros and BRAM macros. In truth, due to the characteristics of DSP and the features of synthesis and mapping tools, the input and output signals of DSP instances in cascaded DSP macros are often connected to FFs as illustrated in Figure 1(c). This is a prevalent occurrence in the context of FPGA placement. It's noteworthy that, given the multitude of input and output signals from DSPs, the quantity of these FFs can reach hundreds or even thousands. These FFs are also closely connected to the DSP macros in a certain pattern. Thus, from a broad perspective, we can categorize the standard cells closely connected to cascaded macros [2] and the cascaded macros itself as a *cascaded macro group*. It is advantageous for standard cells to be proximate to the location of the macros in a cascaded macro group, such as the adjacent columns where the macros are situated, albeit this is not an inflexible requirement.

B. Multi-Electrostatics-based FPGA Placement

Originally conceived for ASIC placement [21], the electrostatic-based placement model conceptualizes each instance as an electric particle within an electrostatics system. This methodology draws on the fundamental physics principle that a balanced distribution of charge in an electrostatics system results in diminished potential energy. Consequently, minimizing the potential energy can alleviate density overflow and facilitate the equitable dispersion of instances within in layout.

This paradigm is subsequently broadened to encompass multiple electrostatic fields, thereby accommodating the diversity of resource types $\mathcal{T}$={LUT, FF, DSP, BRAM, IO} in FPGA placement [11].

The multi-electrostatics-based FPGA placement model can be written as

$$\min_{\boldsymbol{x},\boldsymbol{y}} \widetilde{W}(\boldsymbol{x},\boldsymbol{y}) \quad s.t.\ \Phi_s(\boldsymbol{x},\boldsymbol{y};\boldsymbol{\mathcal{A}}^s)=0, \quad \forall s \in S, \tag{1}$$

[2]We can infer it from the instance name on MLCAD 2023 FPGA macro placement benchmark [20].

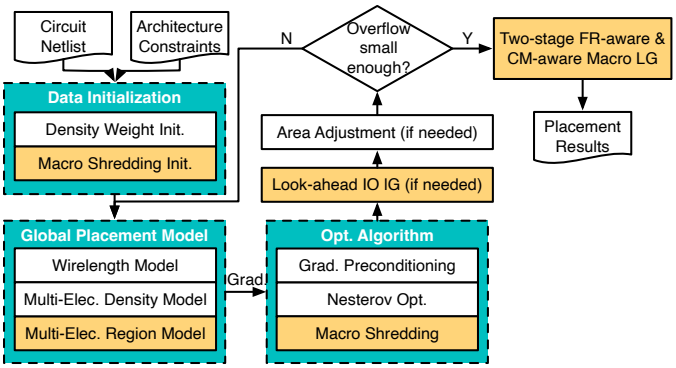

Fig. 2: Overview of the proposed framework.

where $\widetilde{W}(\cdot)$ is the wirelength objective [22]–[24], $\boldsymbol{x},\boldsymbol{y}$ are instance locations, $S = \{S^D_{LUT}, S^D_{FF}, S^D_{DSP}, S^D_{BRAM}, S^D_{IO}\}$ denotes the electrostatics system set, $\Phi_s(\cdot)$ is the electric potential energy for electrostatics system $s \in S$, and $\boldsymbol{\mathcal{A}}^s$ denotes the placement instance areas (including movable instances, fixed instances, and fillers) for electrostatics system s. By mitigating the influence elicited by selecting a zero potential point [17], we can formally target the density $\Phi_s(\cdot)$ as zero. In practical terms, we halt the optimization process once the energy reaches a sufficiently low threshold or when the density overflow is adequately reduced [3].

1) Augmented Lagrangian Formulation: A modified *augmented Lagrangian formulation* is used to ease the constraints into the objective and to formulate a better unconstrained problem [11], [25],

$$\min_{\boldsymbol{x},\boldsymbol{y}} \quad \mathcal{L}(\boldsymbol{x},\boldsymbol{y}) = \widetilde{W}(\boldsymbol{x},\boldsymbol{y}) + \langle \boldsymbol{\lambda}, \boldsymbol{\Phi} + \frac{\mu}{2}\boldsymbol{\mathcal{P}}^{\Phi} \odot \boldsymbol{\Phi}^2 \rangle \tag{2}$$

where $\boldsymbol{\mathcal{P}}^{\Phi}$ is a vectorized density weight preconditioner based on the initial density $\boldsymbol{\mathcal{P}}^{\Phi} = 1/\boldsymbol{\Phi}^{(0)}$, and μ is the quadratic penalty coefficient that can balance the magnitudes between the first-order and second-order terms [11] [4]. By using the gradient descent method, we can solve the above optimization problem.

2) Problem Formulation: In this work, we focus on the FPGA placement problem considering fence region constraints and cascaded macro constraints. We define the FPGA placement problem as follows [20].

Problem 1 (FPGA Placement)**.** *Given a netlist consisting of instances in {LUT, FF, DSP, BRAM, IO} and the FPGA target architecture, produce a feasible FPGA placement solution with optimized routability, satisfying the fence region constraints $\mathcal{F}$ and cascaded macro constraints $\mathcal{C}$.*

III. Algorithms

A. Overview of the Proposed Algorithm

Figure 2 illustrates the overall framework. Our proposed framework takes a circuit netlist and architecture constraints as input and generates legal macro placement results.

The global placement commences with the density weight initialization (Section III-C) and macro shredding initialization (Section III-D). Subsequently, we employ the multi-electrostatic placement model, and obtain the objective function value along with its gradient. We then subject the gradient to preconditioning treatment [15] and utilize the Nesterov algorithm to optimize the positions of placement instances. After each Nesterov iteration, we apply the macro shredding technique to handle cascaded macro constraints (Section III-D) and propose a novel divergence-aware density weight scheduling method (Section III-C). Next, we sequentially evaluate the need for IO legalization (Section III-E) and area adjustment [11].

[3]For brevity, we simplify $\Phi_s(\boldsymbol{x},\boldsymbol{y};\boldsymbol{\mathcal{A}}^s)$ to Φ_s for all $s \in S$, and the potential energy vector whose elements are $\Phi_s(\forall s \in S)$ is denoted by $\boldsymbol{\Phi}$ in later discussions.

[4]For brevity, we denote $\mathcal{D}$ as $\langle \boldsymbol{\lambda}, \boldsymbol{\Phi} + \frac{\mu}{2}\boldsymbol{\mathcal{P}}^{\Phi} \odot \boldsymbol{\Phi}^2 \rangle$ in later discussions.

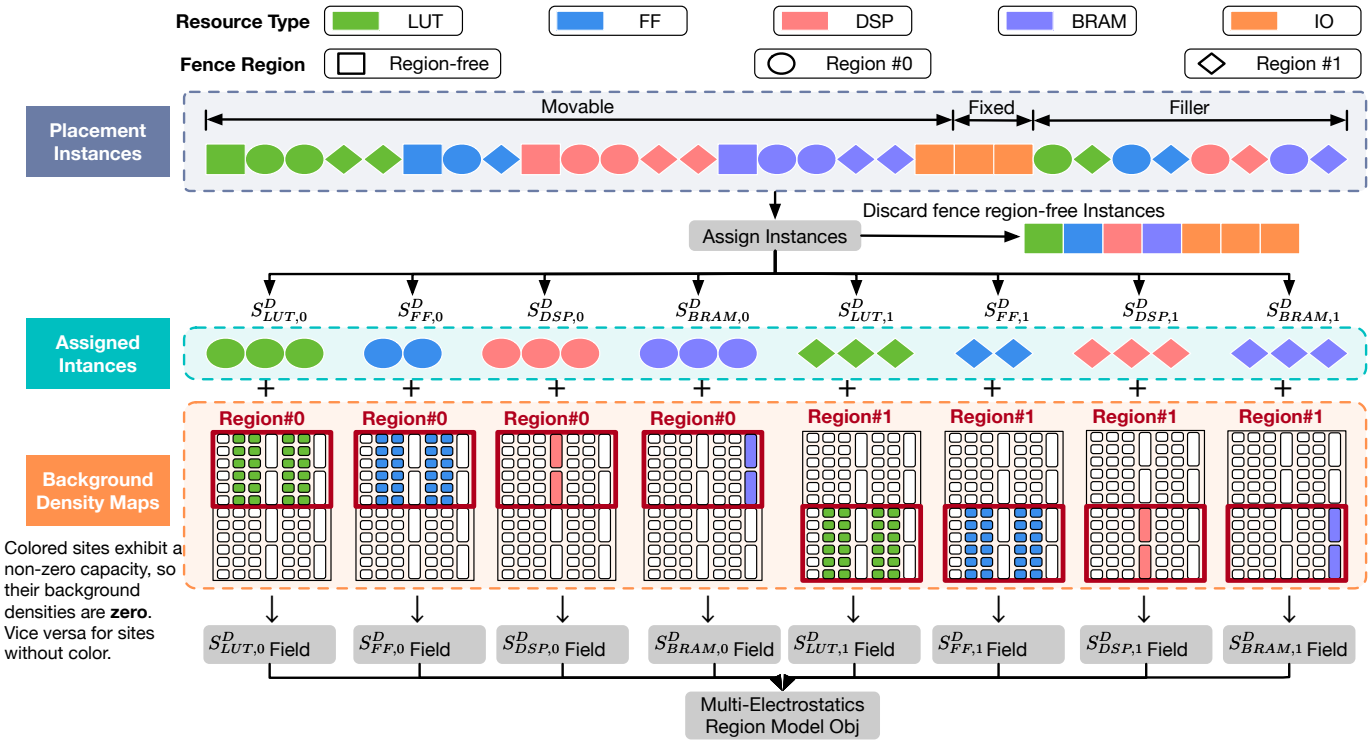

Fig. 3: An example illustration for the multiple electrostatics region model. The colors of the instances represent their respective resource types, while the shapes indicate whether they subject to region constraints and the corresponding numbers of region constraints. In this example, there are eight electrostatics region models, individually crafted for the various resource types inherent in each fence region constraint.

TABLE I: Main Annotations.

$\mathcal{T}$	The set of resource types. $\mathcal{T}$={LUT, FF, DSP, BRAM, IO}.
$\mathcal{F}$	The set of fence region constraints (Section II-A1).
$\mathcal{N}$	The set of movable instances and fixed instances in the netlist (Section III-B).
$\mathcal{O}$	The set of artificially created fillers (Section III-B).
S^R	The set of electrostatics systems for multi-electrostatics region model (Section III-B).

Finally, we determine if the current overflow satisfies the conditions for macro legalization (Section III-F). If so, we proceed with the two-stage macro FR-aware and CM-aware legalization to obtain the macro placement results. Otherwise, we continue the iterative optimization process.

The multi-electrostatics placement model comprises three major optimization models: the WAWL wirelength model [23], the multi-electrostatics density model for electrostatics system set S_D akin to [11], and the multi-electrostatics region model for electrostatics system set S_R (see Section III-B). We adopt the formulation described in Section II-B and the electrostatics system is extended with the multi-electrostatics region model, denoted as $S = \{S^D, S^R\}$. A detailed explanation of the multi-electrostatics region model will be provided in Section III-B. The update strategy for density weight also plays a crucial role in the quality of the optimization results and whether they will diverge. Further details regarding the density weight update strategy will be discussed in Section III-C.

In the following sections, we will elaborate on the details of each component of the proposed framework. We also summarize the commonly used annotations in Table I to facilitate efficient retrieval.

B. Multi-Electrostatics Region Model

Figure 3 illustrates our proposed multi-electrostatics region model. For each resource type $t \in \mathcal{T}$ within each fence region $f \in \mathcal{F}$, we construct a novel electrostatics system $S^R_{t,f}$ (which indicates totally $|\mathcal{T}| \times |\mathcal{F}|$ electrostatics systems, i.e., $|S^R| = |\mathcal{T}||\mathcal{F}|$). For each electrostatics system $S^R_{t,f}$, we derive its potential energy $\Phi^R_{t,f}$ as the density objective as follows.

- All **placement instances** are firstly categorized as movable instances, fixed instances, and fillers, with movable instances and fixed instances representing instances in the netlist (denoted as $\mathcal{N}$). Fillers are artificially created for the purpose of target density control for each electrostatics system, which will be discussed later (denoted as $\mathcal{O}$). The placement instances are then assigned to electrostatics systems based on their resource type and the constraints imposed by the fence region they subject to.
- The **assigned instances** for electrostatics system $S^R_{t,f}$ include (1) the movable instances and fixed instances that own resource type t within the fence region f, and (2) the fillers that are assigned to $S^R_{t,f}$. The movable and fixed instances of this electrostatics system is denoted as $\mathcal{N}^R_{t,f}$. The fillers are assembled as per the method delineated in [26], with a target density empirically set to 1.
- The **background density map** for electrostatics system $S^R_{t,f}$ indicates the potential site locations for instances $N_{t,f}$. We first extract the capacity maps for sites that process resource type t within the fence region f , and then derive the background density maps for $S^R_{t,f}$. For sites with capacity, the background density is set to zero, indicating the permissibility of placing instances on those sites. Vice versa for sites without capacity.

Combining the assigned instances and background density maps, we can calculate the potential energy of the electrostatics system as the density objective [11]. Movable instances and fillers are driven towards their target sites by the repulsive force exerted by the background density map. Fillers crafted from individual target density lead to compact placement in each fence region. When the overflow of this system is sufficiently small, it can be considered equivalent to all instances within $\mathcal{N}^R_{t,f}$ being in the feasible sites within the fence region.

1) Footprint Compression Technique under Multi-electrostatics Region Models: During the implementation of the electrostatics system, memory allocation is required to store the sizes of placement instances under different electrostatics systems. This is the most memory-consuming part. In [11], an array A of size $(|\mathcal{N}| + |\mathcal{O}|) \times |S^R| \times 2$ is created, where $A_{i,j,\{0,1\}}$ represents the length and width of placement instance i in electrostatics system j, as illustrated in

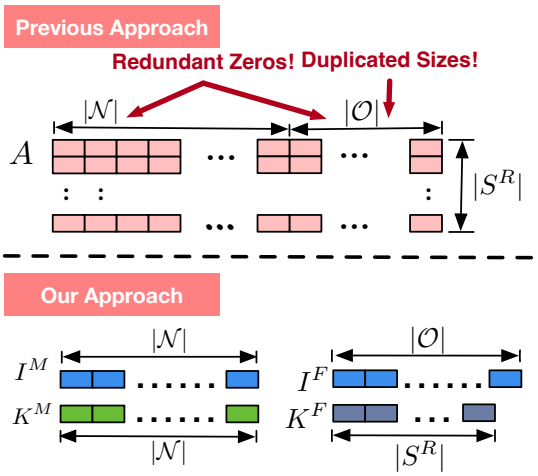

Fig. 4: The memory footprint comparison between [11] and our proposed method. For the sake of conciseness, we omit the last dimension of arrays A, K^M, and K^F in this illustration.

Figure 4. However, this implementation results in a space complexity of $O((|\mathcal{N}|+|\mathcal{O}|)|S^R|)$, which is unacceptable.

To address this challenge, we propose a footprint compression technique under multi-electrostatics region models, which can compress the space complexity to $O(|\mathcal{N}|+|\mathcal{O}|+|S^R|)$. Our technique is based on three key observations:

- Each movable and fixed instance subject to a maximum of one fence region constraint, and thus can be assigned to at most one electrostatics system.
- Each filler is exclusively associated with one electrostatics system.
- Filler sizes within the same electrostatics system are consistent.

So the basic idea is, within the array A, there exist a considerable number of redundant zeros as well as duplicated (filler) sizes. Therefore, we can establish index arrays for instances in $\mathcal{N}$ and $\mathcal{O}$, respectively, indicating the electrostatics system to which each instance belongs. Furthermore, we can also compress the filler sizes within the same electrostatics system.

As shown in Figure 4, for movable and fixed instances, we construct an index array $I^M \in \mathbb{Z}^{|\mathcal{N}|}$ and a key array $K^M \in \mathbb{R}^{|\mathcal{N}|\times 2}$. I_i^M represents the electrostatics system to which instance i is assigned and K_i^M represents the size of the instance. For fillers, we construct an index array $I^F \in \mathbb{Z}^{|\mathcal{O}|}$ and $K^F \in \mathbb{R}^{|S^R|\times 2}$. Here, I_i^F represents the electrostatics system to which instance i is assigned, and K_j^F represents the size of the filler within electrostatics system j.

With the aforementioned technique, we can compress the storage of the array that originally had the highest memory consumption to a space complexity of $O(|\mathcal{N}|+|\mathcal{O}|+|S^R|)$. In practice, we can reduce the GPU memory usage of the framework from 21.4GB to 3.9GB even on the design with the maximum number of regions (i.e., `Design_142` with 22 fence regions [20]), suggesting that the constants beyond complexity are indeed tolerable.

C. Divergence-aware Density Weight Scheduling

Simultaneously optimizing multiple heterogeneous electrostatic field systems can easily lead to divergence issues. Therefore, we propose a modified divergence-aware subgradient-based method [26] to update the density weight. For brevity, we follow the annotation in [26], and denote two additional auxiliary variables as follows:

$$w_s^{(t+1)} = \sum_{i\in\mathcal{N}_s} |\partial\widetilde{W}^{(t)}/\partial x_i|_1, \quad \forall s \in S \tag{3a}$$

$$d_s^{(t+1)} = |\nabla\mathcal{D}_s^{(t)}|_1, \quad \forall s \in S \tag{3b}$$

where $\mathcal{N}_s$ represents all the nodes within the electrostatics system denoted by s, and $\boldsymbol{w}^{(\cdot)}$ and $\boldsymbol{d}^{(\cdot)}$ respectively represent the L1-norm magnitudes of the wirelength gradient and density gradient within each electrostatics system. The key difference of our weight scheduling from [26] lies in the divergence-aware weight $\boldsymbol{\theta}^{(\cdot)}$ when updating $\boldsymbol{\lambda}^{(\cdot)}$ as follows:

$$\boldsymbol{\theta}^{(t+1)} = \max\left(1, \boldsymbol{d}^{(t+1)}/\boldsymbol{w}^{(t+1)}\right)/\boldsymbol{\lambda}^{(t)} \tag{4a}$$

$$\boldsymbol{\lambda}^{(t+1)} = \min(\boldsymbol{u}^{(t+1)} \odot \boldsymbol{v}^{(t+1)}, \gamma/\boldsymbol{\theta}^{(t+1)}) \tag{4b}$$

where γ is emperically set to 2. The intuition behind $\boldsymbol{\theta}^{(\cdot)}$ is that we observe when the optimization comes to the final stage, the large density gradients ($|\boldsymbol{d}^{(\cdot)}| \gg |\boldsymbol{w}^{(\cdot)}|$), whose underlying reason is the excessively rapid growth of $\boldsymbol{\lambda}$, is likely to cause the optimization to diverge. Thus, we introduce $\boldsymbol{\theta}^{(\cdot)}$ to balance the density and wirelength gradients. During the initial stages of optimization, the wirelength gradient reigns supreme, thus it is imperative to regulate the growth rate of $\boldsymbol{\lambda}$ to not exceed γ times. However, as we progress to the later stages, when the density gradient becomes dominant, $\boldsymbol{\theta}^{(\cdot)}$ can effectively govern the growth rate of λ_s not to surpass $\frac{\gamma}{\boldsymbol{d}^{(t+1)}/\boldsymbol{w}^{(t+1)}}$ times. The larger the density gradient, the more pronounced the decelerating effect of growth becomes. By employing this particular strategy, we can effectively control divergence, and we will further demonstrate its effectiveness in Section IV-C.

D. Cascaded Macro Shredding Technique

We overcome the cascaded macro-constraints through the macro shredding technique [27]. The basic idea is to substitute large cascaded structures with multiple individual macros during the optimization process. During the initialization phase, we first identify the cascaded macro groups and enlarge the weight of nets within the cascaded macro groups by a factor of 2. Then, we shred the cascaded macros into individual macros and update the placement. At the end of each iteration, we gather the shredded macros and arrange them in a columnar shape based on the horizontal coordinates of their center of gravity. What sets it apart is that the resulting instance sizes achieved through macro shredding can be directly processed by any existing placement engine, but can also achieve similar results when compared to the results when treating the cascaded shape as a single one.

E. Look-ahead IO Legalization

During the initial stages, movable IOs undergo optimization within a dedicated electrostatics system. Due to the relatively smaller number of IO instances and IO sites, the occurrence of IO overflow quickly diminishes and remains stable. We observe that if the optimization of IO positions continues unabated during this phase, the optimizer may persistently pursue IO position optimization, thereby impeding the optimization of other electrostatics systems. To address this issue, when the following two conditions are met:

- The number of iterations exceeds 20.
- The numerical values of the IO overflow in the 20 most recent iterations satisfy: $(max - min)/mean < 0.1$.

we employ a bipartite graph matching-based legalization algorithm to proactively legalize the IOs.

F. Two-stage FR-aware and CM-aware Macro Legalization

When the overflow of LUT and FF is less than 0.15, and the overflow of BRAM and DSP is less than 0.25, it can be considered that the macro is located near the target sites. Due to the varying sizes and shapes of macros, as well as their significant impact on routability, we sequentially apply the Tetris-based Legalization algorithm and the bipartite matching-based legalization algorithm to legalize cascaded macros and individual macros, while satisfying the fence region constraints.

TABLE II: MLCAD 2023 Public Benchmark Suite Statistics [20].

Designs	Statistics
Number	140
#Instances	558K-711K
#DSP+#BRAM	2.4K-2.7K
LUT (%)	70%-84%
FF (%)	38%-47%
BRAM (%)	80%-90%
DSP (%)	80%-90%
Rent[5]	0.65-0.72
#Regions	0-22
#Instances within Regions	0-285K
Instances within Regions (%)	0%-44.29%
Cascaded DSP Macros Size	{2, 5, 7, 10, 60} ×
Cascaded BRAM Macros Size	{2, 5, 7, 10, 30} ×

IV. Experimental Results

We implement our proposed algorithms in C++/Python and embrace `Pytorch` for agile GPU acceleration. We perform the experiments on a Linux machine running with Intel(R) Xeon(R) Silver 4210R CPU (2.40 GHz, 10 cores), 320 GB RAM, and one NVIDIA GeForce RTX 2080 GPU. The public benchmark suite released by AMD Xilinx for the MLCAD 2023 FPGA macro placement contest [20] are used to validate the effectiveness of the proposed approaches. Section IV-A illuminates the benchmark and the evaluation flow. In Section IV-B, we undertake a comprehensive comparison between our proposed approach and other state-of-the-art FPGA macro placers. In Section IV-C, we meticulously evaluate the robustness of our proposed methodology.

A. Experimental Setup

1) Benchmark Suite: The target FPGA for the benchmark is a 16nm single-die *Ultrascale+ xcvu3p ffvc1517-1-i* device. The statistics of the benchmark suite (140 cases) are summarized in Table II. The high Rent values, the elevated resource utilization levels, the large cascaded macros with a maximum length reaching half the height of the layout, and the significant proportion of instances constrained by fence regions make the placement problem highly challenging.

2) Evaluation Metrics: We employ the same formulas and notations, i.e., $\boldsymbol{score}$ and $\boldsymbol{routability}$, as [20] to define the evaluation metrics. We also report the raw macro placement runtime in minutes as $\boldsymbol{RT_{mpl}}$ for reference [6] . Additionally, we observe that the intermediate metric $routing\_score_{final}$ in [20] mostly yields a value of 1 in the results, lacking discrimination. Therefore, we include the total ripup-reroute iterations [7] as an additional metric $\boldsymbol{\#ripup}$ to assess the complexity of detailed routing. It is worth noting that the evaluation metrics, namely $score$ and RT_{mpl}, are related to the runtime and are machine-dependent. On the other hand, $routability$ and $\#ripup$ are machine-independent.

3) Evaluation Flow: We employ the evaluation flow identical to the MLCAD 2023 contest [20]. We use Vivado 2021.1 for standard cell placement and routing in this flow. We compare our proposed framework with the top three FPGA macro placers in the contest announcement [20], i.e., `UTDA`, `SEU`, and `MPKU`. The executables are obtained from their authors and executed on our machine. `UTDA`, `SEU`, and `MPKU` run on CPU with 16 threads, while our framework is executed on GPU. It is noteworthy that the execution time of all the placers is within 10 minutes, and it does not incur any penalty in the metrics formula provided in [20]. Our placer supports running on CPU as well. Its runtime is comparable to the other placers and does not exceed 10 minutes. Due to the page limit, we omit to report our placement runtime on CPU, since it does not trigger any penalty.

[5]*Rent* serves as a metric that reflects the complexity of interconnect nodes in terms of their routing demand in the Xilinx Vivado report. It is generally considered that Rent exceeding 0.65 is deemed as high.

[6]Please distinguish between the macro runtime score $time_{mpl}$. Their relationship can be expressed as $time_{mpl} = 1 + \max(0, RT_{mpl} - 10)$ [20].

[7]This metric is delineated as the total count of lines that encompass the string "Number of Nodes with overlaps" within Phase 4 of the Vivado log file.

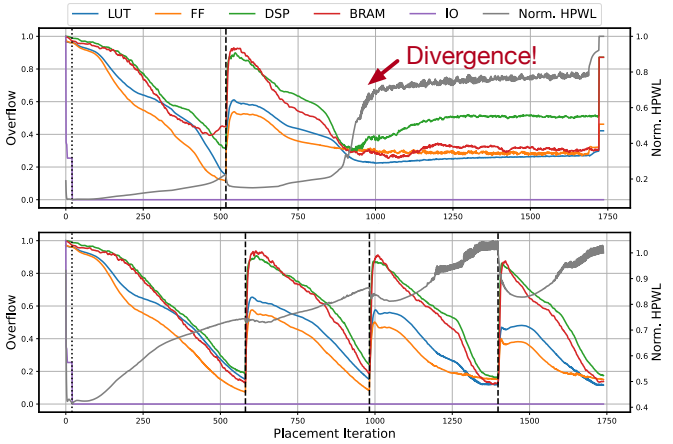

Fig. 5: The overflows and HPWL during the placement iteration on the `Design_156` using 1) the density weight updating method in [26] (depicted in the upper graph), and 2) our proposed method (depicted in the lower graph), respectively. The short dashed lines represent look-ahead IO legalization (Section III-E), and the long dashed lines represent area adjustment (Section III-A).

Meanwhile, we also include the default macro placement of Vivado 2021.1 as a reference, denoted as `Vivado 2021.1`. Since we cannot determine the individual time taken by Vivado for macro placement, we consider the variable $time_{P\&R}$ in [20] as the entire default Vivado runtime and assume no macro placement penalty, i.e., $time_{mpl} = 1$. This is reasonable as the aforementioned FPGA macro placers do not incur any macro placement runtime penalty.

B. Comparison with State-of-the-Art Placers

Table III shows the overall comparison results with `Vivado 2021.1`, `UTDA`, `SEU`, and `MPKU`. The proposed placer demonstrates a significant overall score improvement of 27.8%, 6.9%, 13.3%, and 49.1% when compared to `Vivado 2021.1`, `UTDA`, `SEU`, and `MPKU`, respectively. Table III also shows the detailed comparison results of $routability$ and $\#ripup$, respectively. Our proposed placer also achieves a $routability$ improvement of 22.8%, 8.7%, 12.1%, and 39.1% respectively. In terms of $\#ripup$, our proposed placer also achieves a significant advantage, reducing $\#ripup$ by 10.8%, 2.7%, 4.0%, and 16.8% respectively, compared to `Vivado 2021.1`, `UTDA`, `SEU`, and `MPKU`. These results demonstrate that our proposed placer can achieve superior routability, thereby reducing the runtime of detailed routing and the number of ripup-reroute iterations. In all three metrics, our proposed placer consistently achieves a significant advantage, which fully demonstrates the superiority of our proposed placer.

1) Runtime Evaluation: Table III shows the macro placement runtime RT_{mpl} of each placer. The experimental results demonstrate that our GPU-accelerated multi-electrostatics-based placement achieves 3.180×, 1.808×, and 2.599 × speedup over `UTDA`, `SEU`, and `MPKU`, respectively. These results demonstrate that our proposed placer can achieve a significant speedup over the state-of-the-art FPGA macro placers, thereby demonstrating the superiority of our proposed placer in runtime.

C. Robustness Evaluation

Figure 5 exemplifies the underlying causes of divergence and demonstrates the advantages of our proposed divergence-aware density weight scheduling. In the upper illustration, it is observed that during divergence, the overflow does not decrease; instead, it slightly increases. This indicates that the Nesterov algorithm fails to optimize the energy potential $\boldsymbol{\Phi}$. The primary reason for this lies in the excessive growth of $\boldsymbol{\lambda}$ at this stage, causing instances to wander

TABLE III: Comparison with State-of-the-Art Placers on MLCAD 2023 FPGA Macro Placement Benchmark Suite (140 Cases).

Metrics	Vivado 2021.1		UTDA (1st†)		SEU (1st†)		MPKU (2nd)		Ours	
	Geo. Mean	Ratio	Geo. Mean	Ratio	Geo. Mean	Ratio	Geo. Mean	Ratio	Geo. Mean	Ratio
$score$	4.592	1.278	3.838	1.069	4.069	1.133	5.356	1.491	**3.592**	**1.000**
$routability$	2.802	1.228	2.482	1.087	2.558	1.121	3.174	1.391	**2.282**	**1.000**
$\#ripup$	7.492	1.108	6.940	1.027	7.027	1.040	7.896	1.168	**6.759**	**1.000**
RT_{mpl}	-	-	5.203	3.180	2.958	1.808	4.252	2.599	**1.636**	**1.000**

† UTDA and SEU are jointly ranked first in the official announcement.

near feasible sites (in the vicinity of the potential wells formed by background density maps). Consequently, the search for the optimal position is neglected, and even the range of optimal solutions is surpassed.

On the other hand, in the lower illustration, we observe that our proposed method effectively resolves the issue of divergence. Our approach allows for the rational adjustment of the growth rate of $\boldsymbol{\lambda}$ while simultaneously preserving the reduction of overflow. **In fact, we have successfully applied our proposed method to all designs in [20] without any rollback mechanism or entropy injection strategies [26] , ensuring the absence of divergence and achieving sufficiently low overflow**. This substantiates the effectiveness of our proposed divergence-aware density weight scheduling.

V. Conclusion

In this paper, we propose a robust multi-electrostatics-based FPGA macro placer OpenPARF 3.0 that can handle both fence region constraints and cascaded macro constraints efficiently. Based on the SOTA multi-electrostatics-based FPGA placement model, we propose a novel multi-electrostatics region model to handle the discontinuity of the solution space as well as a macro shredding technique to mitigate the imbalance in the size of the cascaded macros. We also propose a dynamic density weight scheduling scheme to address robustness issues of divergence. Experimental results on the MLCAD 2023 FPGA macro placement benchmark demonstrate that our proposed framework can achieve 27.8%, 6.9%, 13.3%, and 49.1% improvement compared to the recent cutting-edge FPGA macro placers Vivado 2021.1, UTDA, SEU, and MPKU respectively, with 1.81-3.18× speedup leveraging GPU acceleration.

Acknowledgments

This work was supported in part by the National Science Foundation of China (Grant No. T2293700 and T2293701) and the 111 Project (B18001). The authors would like to thank Ismail Bustany *et al*. from AMD Xilinx Inc. for organizing the MLCAD 2023 FPGA Macro Placement Contest, Zhili Xiong from the University of Texas at Austin for preparing the docker image of UTDA, and Hao Gu from Southeast University for preparing the binary of SEU.

References

[1] S. Yang, A. Gayasen, C. Mulpuri, S. Reddy, and R. Aggarwal, "Routability-driven FPGA placement contest," in *Proc. ISPD*, 2016, pp. 139–143.

[2] S. Yang, C. Mulpuri, S. Reddy, M. Kalase, S. Dasasathyan, M. E. Dehkordi, M. Tom, and R. Aggarwal, "Clock-aware FPGA placement contest," in *Proc. ISPD*, 2017, pp. 159–164.

[3] D. Maarouff, A. Shamli, T. Martin, G. Grewal, and S. Areibi, "A deep-learning framework for predicting congestion during fpga placement," in *Proc. FPL*. IEEE, 2020, pp. 138–144.

[4] W. Li, S. Dhar, and D. Z. Pan, "UTPlaceF: A routability-driven FPGA placer with physical and congestion aware packing," in *Proc. ICCAD*, 2016, pp. 66:1–66:7.

[5] W. Li, Y. Lin, M. Li, S. Dhar, and D. Z. Pan, "Utplacef 2.0: A high-performance clock-aware fpga placement engine," *ACM TODAES*, vol. 23, no. 4, pp. 1–23, 2018.

[6] W. Li, M. Li, J. Wang, and D. Z. Pan, "Utplacef 3.0: A parallelization framework for modern fpga global placement," in *Proc. ICCAD*. IEEE, 2017, pp. 922–928.

[7] W. Li and D. Z. Pan, "A new paradigm for fpga placement without explicit packing," *IEEE TCAD*, vol. 38, no. 11, pp. 2113–2126, 2018.

[8] W. Li, M. E. Dehkordi, S. Yang, and D. Z. Pan, "Simultaneous placement and clock tree construction for modern fpgas," in *Proc. FPGA*, 2019, pp. 132–141.

[9] R. Pattison, Z. Abuowaimer, S. Areibi, G. Grewal, and A. Vannelli, "GPlace: A congestion-aware placement tool for ultrascale FPGAs," in *Proc. ICCAD*, 2016, pp. 68:1–68:7.

[10] Z. Abuowaimer, D. Maarouf, T. Martin, J. Foxcroft, G. Grewal, S. Areibi, and A. Vannelli, "GPlace3. 0: Routability-driven analytic placer for ultrascale fpga architectures," *ACM TODAES*, vol. 23, no. 5, pp. 1–33, 2018.

[11] Y. Meng, W. Li, Y. Lin, and D. Z. Pan, "elfplace: Electrostatics-based placement for large-scale heterogeneous fpgas," *IEEE TCAD*, vol. 41, no. 1, pp. 155–168, 2021.

[12] R. S. Rajarathnam, M. B. Alawieh, Z. Jiang, M. Iyer, and D. Z. Pan, "Dreamplacefpga: An open-source analytical placer for large scale heterogeneous fpgas using deep-learning toolkit," in *Proc. ASPDAC*. IEEE, 2022, pp. 300–306.

[13] R. S. Rajarathnam, Z. Jiang, M. A. Iyer, and D. Z. Pan, "Dreamplacefpga-pl: An open-source gpu-accelerated packer-legalizer for heterogeneous fpgas," in *Proc. ISPD*, 2023, pp. 175–184.

[14] J. Mai, J. Wang, Z. Di, G. Luo, Y. Liang, and Y. Lin, "OpenPARF: an open-source placement and routing framework for large-scale heterogeneous fpgas with deep learning toolkit," in *Proc. ASICON*, 2023, pp. 1–4.

[15] J. Mai, Y. Meng, Z. Di, and Y. Lin, "Multi-electrostatic fpga placement considering slicel-slicem heterogeneity and clock feasibility," in *Proc. DAC*, 2022, pp. 649–654.

[16] J. Chen, Z. Lin, Y.-C. Kuo, C.-C. Huang, Y.-W. Chang, S.-C. Chen, C.-H. Chiang, and S.-Y. Kuo, "Clock-aware placement for large-scale heterogeneous fpgas," *IEEE TCAD*, vol. 39, no. 12, pp. 5042–5055, 2020.

[17] J. Mai, J. Wang, Z. Di, and Y. Lin, "Multi-electrostatic fpga placement considering slicel-slicem heterogeneity, clock feasibility, and timing optimization," *IEEE TCAD*, 2023.

[18] C.-W. Pui, G. Chen, W.-K. Chow, K.-C. Lam, J. Kuang, P. Tu, H. Zhang, E. F. Y. Young, and B. Yu, "RippleFPGA: A routability-driven placement for large-scale heterogeneous FPGAs," in *Proc. ICCAD*, 2016, pp. 67:1–67:8.

[19] C.-W. Pui, G. Chen, Y. Ma, E. F. Young, and B. Yu, "Clock-aware ultrascale fpga placement with machine learning routability prediction," in *Proc. ICCAD*, 2017, pp. 929–936.

[20] I. Bustany, G. Gasparyan, A. Gupta, A. B. Kahng, M. Kalase, W. Li, and B. Pramanik, "The 2023 mlcad fpga macro placement benchmark design suite and contest results," in *Proc. MLCAD*, 2023.

[21] 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 nesterov's method," in *Proc. DAC*, 2014, pp. 1–6.

[22] B. B. Ray, A. R. Tripathy, P. Samal, M. Das, and P. Mallik, "Half-perimeter wirelength model for vlsi analytical placement," in *International Conference on Information Technology*. IEEE, 2014, pp. 287–292.

[23] M.-K. Hsu, Y.-W. Chang, and V. Balabanov, "Tsv-aware analytical placement for 3d ic designs," in *Proc. DAC*, 2011, pp. 664–669.

[24] B. B. Ray and S. Balachandran, "An efficient wirelength model for analytical placement," in *Proc. DATE*. IEEE, 2013, pp. 1711–1714.

[25] Z. Zhu, J. Chen, Z. Peng, W. Zhu, and Y.-W. Chang, "Generalized augmented lagrangian and its applications to vlsi global placement," in *Proc. DAC*, 2018, pp. 1–6.

[26] J. Gu, Z. Jiang, Y. Lin, and D. Z. Pan, "Dreamplace 3.0: Multi-electrostatics based robust vlsi placement with region constraints," in *Proc. ICCAD*, 2020.

[27] P. Ochsendorf, "Timing-driven macro placement," Ph.D. dissertation, Universitäts-und Landesbibliothek Bonn, 2019.