Multi-Electrostatics Based Placement for Non-Integer Multiple-Height Cells

Yu Zhang
The Chinese University of Hong Kong, Hong Kong SAR
Peking University, Beijing, China

Yuan Pu
The Chinese University of Hong Kong
Hong Kong SAR

Fangzhou Liu
The Chinese University of Hong Kong
Hong Kong SAR

Peiyu Liao
The Chinese University of Hong Kong
Hong Kong SAR

Kai-Yuan Chao
Hong Kong Research Center, Huawei Technology Investment Co. Ltd.
Hong Kong SAR

Keren Zhu
The Chinese University of Hong Kong
Hong Kong SAR

Yibo Lin
Peking University, Beijing, China
Institute of EDA, Peking University, Wuxi, China

Bei Yu
The Chinese University of Hong Kong
Hong Kong SAR

ABSTRACT

A circuit design incorporating non-integer multi-height (NIMH) cells, such as a combination of 8-track and 12-track cells, offers increased flexibility in optimizing area, timing, and power simultaneously. The conventional approach for placing NIMH cells involves using commercial tools to generate an initial global placement, followed by a legalization process that divides the block area into row regions with specific heights and relocates cells to rows of matching height. However, such placement flow often causes significant disruptions in the initial placement results, resulting in inferior wirelength. To address this issue, we propose a novel multi-electrostatics-based global placement algorithm that utilizes the NIMH-aware clustering method to dynamically generate rows. This algorithm directly tackles the global placement problem with NIMH cells. Specifically, we utilize an augmented Lagrangian formulation along with a preconditioning technique to achieve high-quality solutions with fast and robust numerical convergence. Experimental results on the OpenCores benchmarks demonstrate that our algorithm achieves about 12% improvements on HPWL with 23.5× speed up on average, outperforming state-of-the-art approaches. Furthermore, our placement solutions demonstrate a substantial improvement in WNS and TNS by 22% and 49% respectively. These results affirm the efficiency and effectiveness of our proposed algorithm in solving row-based placement problems for NIMH cells.

CCS CONCEPTS

• **Hardware → Placement**.

KEYWORDS

Placement, non-integer multi-height cell

ACM Reference Format:
Yu Zhang, Yuan Pu, Fangzhou Liu, Peiyu Liao, Kai-Yuan Chao, Keren Zhu, Yibo Lin, and Bei Yu. 2024. Multi-Electrostatics Based Placement for Non-Integer Multiple-Height Cells . In *Proceedings of the 2024 International Symposium on Physical Design (ISPD '24), March 12–15, 2024, Taipei, Taiwan.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3626184.3633320

1 INTRODUCTION

Typically, standard cells are designed with heights that are integer multiples of a minimum wiring pitch, i.e., a single-row height. However, as digital circuit design requirements continue to evolve, adhering strictly to this standard cell height may result in suboptimal area utilization to meet power and timing requirements. Additionally, reducing the transistor size, such as FinFET spacing [1], may not align optimally with the minimum wiring pitch. Consequently, maintaining a standard cell height as an integer multiple of a single-row height can lead to increased power consumption and unnecessary area costs [2].

Standard-cell libraries can be designed with varying cell heights, such as FinFlex with different fin configurations [3]. This offers vital design flexibility, leading to improved performance, area, and power efficiency. This approach also broadens the solution space and enhances design quality. Specifically, larger height cells have increased drive strengths and pin accessibility, and reduced delay time. However, these advantages come at the cost of increased area, power consumption, and pin capacitance. Conversely, cells with smaller heights result in reduced areas, pin capacitance, and power consumption. However, these cells have weaker drive strengths and are more susceptible to routing congestion and pin accessibility issues. Notably, in industry, TSMC, the newest 3nm manufacturer, has announced its usage of NIMH cells in its FinFlex cell technology to achieve better PPA through the co-optimization of place-and-route [3]. In academia, the application of NIMH cells, as outlined in Dobre's work [4], in a 28nm low-power foundry technology results

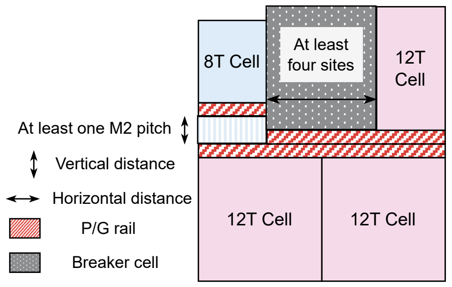


Figure 1: Horizontal and vertical spacings between two cell regions of different heights, with breaker cells [4].

in a 25% reduction in area compared to a 12T-only implementation, while maintaining the same performance. Additionally, it provides a 20% performance improvement compared to an 8T-only implementation, with a similar total cell area.

Compared to traditional cell placement, the introduction of non-integer multiple-height cells brings about several additional constraints. As demonstrated in Figure 1, due to design and spacing limitations, it becomes necessary to insert breaker cells to meet the minimum horizontal spacing distances - a minimum of four placement sites - between two regions with differing cell heights. Furthermore, a vertical gap, such as at least one M2 pitch, is required to circumvent power/ground (P/G) track alignment complications associated with mixed-cell heights. These unique constraints for NIMH cells add complexity to the circuit design problem, rendering it beyond the capabilities of existing commercial tools.

Several studies have explored circuit designs incorporating NIMH cells. Dobre *et al.* [4] pioneered the first study addressing the placement issue of NIMH cells within a block. They proposed an optimized physical design flow to implement design blocks with NIMH cells in a fine-grained manner. However, the efficacy of their results is heavily dependent on the grid size and cut size during the partitioning phase. For larger designs, both the grid size and cut size must be significantly increased for optimal results, leading to a considerable increase in runtime. Furthermore, the granular approach may result in substantial cell displacement, subsequently escalating wirelength and power consumption. Chen *et al.* [5] introduced an analytical placer designed to directly accommodate a design incorporating NIMH cells. They transformed the NIMH placement issue into an unconstrained optimization problem and suggested an Exact Penalty Iterative Shrinkage and Thresholding (EPIST) algorithm to optimize the global placement problem. In contrast to the island-based partitioning utilized in prior studies, Lin *et al.* [6] suggest partitioning the block area into rows of specific cell heights, based on the initial placement outcome. They then apply k-means clustering to allocate a specific height to each row. When compared to the island-based method, the row-based approach reduces both the routed wirelength and power consumption. However, notable differences can be seen between the initial placement results and the final legal solutions. This implies that the wirelength, meticulously optimized in global placement, can be substantially disrupted during the final legalization.

Our work falls within the realm of row structure-based placement algorithms. However, different from the traditional flow that assigns row regions after global placement, we choose to adaptively generate regions for each cell type during global placement to identify more desired solutions. More importantly, to accommodate various cell types in NIMH cell placement, we adopt the idea of casting placement with heterogeneous cells to multi-electrostatics systems. Overall, we design a unified "Place-Cluster-Place-Legalize" flow to directly solve the row-based global placement problem associated with NIMH cells. The main contributions of this paper are summarized as follows:

- We explore the analogy between the NIMH cell placement problem and the multi-electrostatics system, and introduce a multi-electrostatics-based global placement algorithm to directly solve the global placement problem with NIMH cells and additional constraints.
- We suggest an augmented Lagrangian method with a preconditioning technique to improve the numerical convergence and ensure high-quality solutions.
- We propose a surrogate subgradient method to update density penalty multipliers, which manage the dispersion of different cell types in a self-adaptive way.
- We present a modified *BestChoice* clustering method to dynamically generate row regions for different cell heights during the global placement process.
- Experimental results on the OpenCores [7] benchmark suites demonstrate that we can achieve about 12% improvements on HPWL with 23.5× speed up on average, compared with the state-of-the-art placement algorithm [6] for NIMH cells. Concurrently, our method improves WNS and TNS by 22%, and 49%, respectively.

2 PRELIMINARIES

2.1 Layout Constraints

Beyond the conventional placement constraints, there are further restrictions for non-integer multiple-height cell placement. As in [4], the intricate constraints of NIMH cells are as follows:

Constraint $C1$: To ensure manufacturability, there must be at least two cell rows in each region of a particular cell height.

Constraint $C2$: Due to N-well rules, there must be an even number of cell rows in each fixed region of a particular cell height.

Constraint $C3$: Each cell must be located at a placement site on a row, and its corresponding region with a particular cell height must align with the overall metal and track definitions of the chip.

Constraint $C4$: For any regions with different cell heights, the horizontal spacing must be at least four placement sites.

Constraint $C5$: There must be a vertical gap to avoid the P/G track alignment issue with two partitions of different cell heights; the minimum vertical distance between the two partitions must ensure that the P/G rail of one region does not encroach on that of another region.

Constraint $C6$: Breaker cells must be inserted to meet the minimum horizontal spacing between two regions with different cell heights.

Constraints $C4$, $C5$ and $C6$ are illustrated in Figure 1.

2.2 Multi-Electrostatics Based Placement

The electrostatics-based placement concept [8], initially introduced for ASIC placement, leverages the fundamental principle that a balanced charge distribution in an electrostatic system results in low

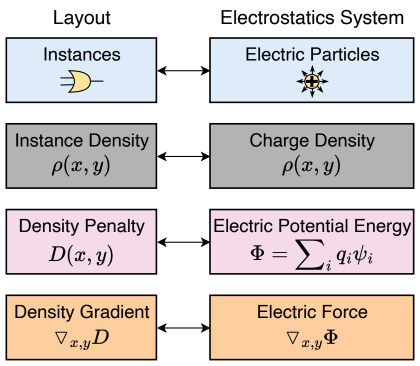

Layout
Electrostatics System
Instances
Electric Particles
Instance Density $\rho(x,y)$
Charge Density $\rho(x,y)$
Density Penalty $D(x,y)$
Electric Potential Energy $\Phi = \sum_i q_i\psi_i$
Density Gradient $\nabla_{x,y}D$
Electric Force $\nabla_{x,y}\Phi$

Figure 2: Analogy between ASIC placement and an electrostatic system [8].

potential energy. By minimizing this potential energy, density overflow issues can be mitigated, and instances can be uniformly distributed within the layout. This approach equates each instance to an electric particle within an electrostatic system, drawing an analogy between the electrostatic system and the ASIC placement problems. As illustrated in Figure 2, cells are represented as positive charges, cell density as potential energy, density penalty as electric potential energy, and the gradient corresponds to the electric force guiding the cell movement. This electrostatic analogy offers two key advantages: 1) a smooth density penalty function, and 2) a comprehensive view of the entire placement region, even with extremely fine bin dimensions.

This methodology is subsequently extended to multi-electrostatic fields for FPGA placement [9], thereby accommodating various resource types in FPGA designs such as LUT, FF, DSP, and BRAM. Particularly, the multi-electrostatic-based placement models the density constraints for each resource type as a distinct electrostatic system, aiming to minimize the total potential energy across multiple fields. Hence, the problem can be formulated as:

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

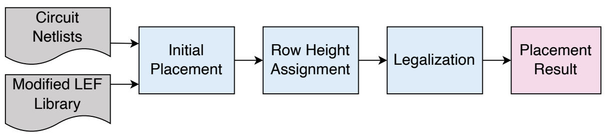
where $\boldsymbol{x},\boldsymbol{y}$ represent instance locations, $\tilde{W}(\cdot)$ signifies the wirelength objective, $\mathcal{S}$ is the set of resource types, and $\Phi_s(\cdot)$ designates the electric potential energy of the field for resource type $s \in \mathcal{S}$. The target energy $\Phi_s(\boldsymbol{x},\boldsymbol{y})$ is strictly constrained to 0, since low energy equates to a balanced distribution of instances. Density constraints can be incorporated into the objective and resolved using the gradient descent method. In practical terms, optimization is halted when the energy reaches a sufficiently low level, or equivalently, when the density overflow is minimal enough.

It's worth noting that while our proposed placement algorithm also employs the multi-electrostatic system to model the placement problem, our row regions are dynamically generated during global placement. This is in contrast to the regions of FPGA placement, which are predefined in the floorplan stage [10].

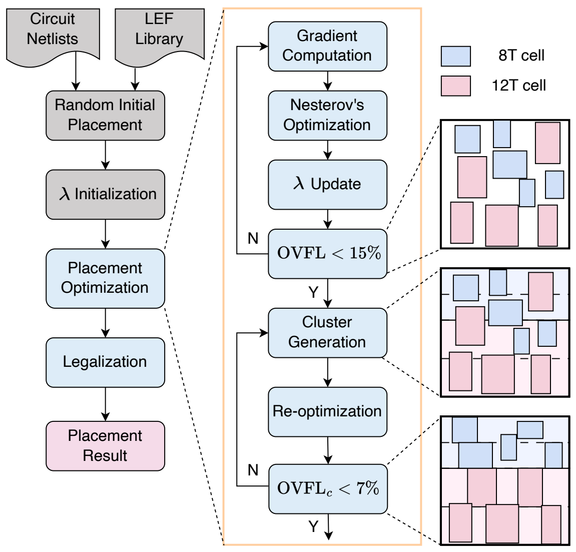
2.3 Problem Formulation

We formally define the NIMH cell placement problem below:

Problem 1 (Non-integer Multiple-height Cell Placement)**.** Given m nets and n cells in a standard cell library with non-integer multiple heights, determine the position of each cell such that the total wirelength is minimized, and all NIMH layout constraints are satisfied.



(a)



(b)

Figure 3: (a) Traditional placement flow for NIMH cells; (b) The proposed placement flow for NIMH cells.

3 MULTI-ELECTROSTATICS BASED PLACEMENT

In this section, we present an effective placement flow that accommodates the NIMH layout constraints and optimizes wirelength. In previous work [6], the initial placement is generated by commercial tools after modifying LEF files so that all cells and rows have the same height. Then, regions are determined by row partitioning based on the initial global placement result, followed by an NIMH-aware legalization process to move cells into the row region with the same height (see Figure 3(a)). This method could lead to sub-optimal wirelength due to substantial disruptions to the initial placement. As a result, it would be more effective to generate row regions during the cell placement process.

Figure 3(b) illustrates the overall flow of our proposed placement algorithm. Starting from a random initial placement, the density multiplier vector $\boldsymbol{\lambda}$ is initialized and then enters the core placement optimization phase. Each placement iteration involves the calculation of a wirelength-density co-optimization problem's gradient, which is then input into a Nesterov's optimizer [8] for a gradient descent step. After that, $\boldsymbol{\lambda}$ is updated to balance the spreading efforts on different cell types and universally emphasize slightly more density penalties. When the global overflow is reduced to 15%, the NIMH-aware clustering method is performed to generate row regions for different cell types. After the clustering, the nearly-equilibrated states may

Table 1: Notations used in our formulation

Notation	Description
$\mathcal{C}$	The cell type set, e.g., {8T, 12T}
$\mathcal{V}, \mathcal{V}_c$	The instance set and its subset with cell type c
$\boldsymbol{\lambda}, \lambda_c$	The density multiplier vector and the density multiplier for cell type c
Φ, Φ_c	The potential energy function and the potential energy for cell type c
θ_λ	The density multiplier preconditioner
q_i	The electric charge quantity of instance i
a_i	The area of instance i
B, B_c	The overall bin grid and the bin grid for cell type c
ρ^b, ρ_c^b	The charge density of grid b and The charge density of grid b with cell type c

be disrupted as cells are relocated to their corresponding regions. Therefore, we reduce the density multipliers $\boldsymbol{\lambda}$ and perform the optimization step again to recover the quality. When the overlaps for all cell types are minimal, specifically when the overflow of each cell type is below 7%, we legalize the global placement results using a row-based legalization approach similar to [11]. Finally, we incorporate vertical distance and update the floorplan following the method in [6], thereby achieving the final legal placement solutions for NIMH cells. Table 1 lists the notations used in our problem/solution formulation. The subsequent subsections provide an in-depth explanation of our algorithm.

3.1 The Augmented Lagrangian Method for Multi-Electrostatics Based Placement

The NIMH cell placement method entails the placement of cells with varying heights. We denote the position of all instances as $(\boldsymbol{x}, \boldsymbol{y})$, and use $\mathcal{C}$ to represent all valid cell types. Considering the density constraint for each cell type c as a distinct electrostatic system, we frame the placement problem with NIMH cells as follows:

$$\min_{\boldsymbol{x},\boldsymbol{y}} \tilde{W}(\boldsymbol{x}, \boldsymbol{y}) \quad \text{s.t. } \Phi_c(\boldsymbol{x}, \boldsymbol{y}) = 0, \ \forall c \in \mathcal{C}, \tag{2}$$

where Φ_c represents the electric potential energy of cell type c. We formally constrain the Φ_c to 0, as the energy is usually nonnegative. Here $\tilde{W}(\boldsymbol{x}, \boldsymbol{y})$ approximates the non-differentiable half-perimeter wirelength (HPWL) with the weighted-average (WA) model [12],

$$\begin{aligned} \tilde{W}_x(\boldsymbol{x}, \boldsymbol{y}) &= \frac{\sum_{i\in e} x_i \exp(x_i/\gamma)}{\sum_{i\in e} \exp(x_i/\gamma)} - \frac{\sum_{i\in e} x_i \exp(-x_i/\gamma)}{\sum_{i\in e} \exp(-x_i/\gamma)}, \\ \tilde{W}(\boldsymbol{x}, \boldsymbol{y}) &= \tilde{W}_x(\boldsymbol{x}, \boldsymbol{y}) + \tilde{W}_y(\boldsymbol{x}, \boldsymbol{y}), \end{aligned} \tag{3}$$

where γ regulates the smoothness and accuracy of the approximation to HPWL. A smaller value of γ suggests a more accurate, albeit less smooth, HPWL approximation. To ensure stability and convergence of our multi-electrostatic system across diverse benchmarks, we enhance the first-order density penalty and employ a modified version of the augmented Lagrangian method (ALM) [9]:

$$\min_{\boldsymbol{x},\boldsymbol{y}} f(\boldsymbol{x}, \boldsymbol{y}) = \tilde{W}(\boldsymbol{x}, \boldsymbol{y}) + \sum_{c\in\mathcal{C}} \lambda_c (\Phi_c(\boldsymbol{x}, \boldsymbol{y}) + \frac{1}{2}\mu\theta_\lambda \Phi_c(\boldsymbol{x}, \boldsymbol{y})^2). \tag{4}$$

Here λ_c represents the density multiplier for each cell type $c \in \mathcal{C}$. The quadratic penalty is also regulated by the density weight $\boldsymbol{\lambda}$ to prevent excessive spreading of cells. The weighting coefficient μ strikes a balance between first-order and second-order density penalty terms. Meanwhile, θ_λ serves as a density weight preconditioner that is based on the initial density, i.e., $\theta_\lambda = 1/\Phi^0$. It is worth noting that slightly different from the canonical ALM formulation where $\Phi(\boldsymbol{x}, \boldsymbol{y})^2$ is typically independent to $\boldsymbol{\lambda}$, the magnitude of second-order penalty in Equation (4) is also influenced by $\boldsymbol{\lambda}$.

Indeed, the ALM formulation can be interpreted as a combination of the multiplier method and the quadratic penalty method [13]. The rationale behind this formulation is that when cell type c has a high potential energy $\Phi_c(\boldsymbol{x}, \boldsymbol{y})$, the quadratic penalty term $\frac{\lambda_c\mu\theta_\lambda}{2}\Phi_c(\boldsymbol{x}, \boldsymbol{y})^2$ would predominate and expedite the convergence. Conversely, when cell type c converges to a relatively low potential energy $\Phi_c(\boldsymbol{x}, \boldsymbol{y})$, the multiplier term $\lambda_c\Phi_c(\boldsymbol{x}, \boldsymbol{y})$ would dominate, aiding in smoothing the optimization without the risk of ill-conditioning. This is how the weighting coefficient μ achieves a balance between first-order and second-order density penalty terms. In practice, we empirically set the value of μ to 1000 to ensure robust convergence.

3.2 Gradient Computation and Preconditioning

The first-order derivatives can be deduced from Equation (4). For the sake of brevity, we will only discuss the x-directed derivatives in the remainder of this subsection, as similar principles apply to the y-directed derivatives. The x-directed gradient of our ALM objective function can be derived as follows:

$$\frac{\partial f(\boldsymbol{x}, \boldsymbol{y})}{\partial x_i} = \frac{\partial \tilde{W}(\boldsymbol{x}, \boldsymbol{y})}{\partial x_i} + \lambda_c (\frac{\partial \Phi_c(\boldsymbol{x}, \boldsymbol{y})}{\partial x_i} + \mu\theta_\lambda \Phi_c(\boldsymbol{x}, \boldsymbol{y}) \frac{\partial \Phi_c(\boldsymbol{x}, \boldsymbol{y})}{\partial x_i}), \forall i \in \mathcal{V}_c, \tag{5}$$

where $\mathcal{V}_c$ represents the set of physical instances with cell type c. Then, the gradient defined in Equation (5) undergoes preconditioning before being input into the optimizer. Preconditioning aids in rendering the local curvature of the objective function nearly spherical, thereby mitigating the ill-conditioning problem and enhancing numerical convergence and stability [8, 14]. The most frequently utilized preconditioner is the inverse of the Hessian matrix $\mathbf{H}$ of the objective function $f(\boldsymbol{x}, \boldsymbol{y})$. In this case, the preconditioned gradient $\mathbf{H}^{-1}\nabla f$, rather than the original ∇f, is used to determine the descent direction. However, computing the exact Hessian matrix $\mathbf{H}$, let alone its inverse, is impractical due to the scale of our placement problem. Consequently, we resort to the more economical Jacobi preconditioner [15] to approximate the complex Hessian, which essentially is a diagonal matrix with the i-th diagonal entry equivalent to $\mathbf{H}$. Moreover, the closed-form expression of $\frac{\partial^2 \tilde{W}(\boldsymbol{x}, \boldsymbol{y})}{\partial x_i^2}$ is computationally expensive to calculate in practice. Therefore, we choose to approximate it as:

$$\frac{\partial^2 \tilde{W}(\boldsymbol{x}, \boldsymbol{y})}{\partial x_i^2} \sim \sum_{e\in\varepsilon_i} \frac{1}{|e| - 1}, \tag{6}$$

where ε_i represents the set of nets corresponding to instance i and $|e|$ signifies the degree of the net e. Hence, the overall x-directed second-order derivative of the objective is approximated as follows:

$$\frac{\partial^2 f(\boldsymbol{x},\boldsymbol{y})}{\partial x_i^2} \sim \tilde{\mathbf{H}}_{x_i} = \max\Big(\sum_{e\in\mathcal{E}_i} \frac{1}{|e|-1} + \lambda_c q_i, 1\Big),\ \forall i \in \mathcal{V}_c, \quad (7)$$

where q_i is the electric charge quantity of charge (instance) i, and the $\max(\cdot, 1)$ operation is designed to avoid small derivatives of filler instances as fillers do not have incident nets. Denote $\tilde{\mathbf{H}}^{-1}$ as the approximated preconditioner, the preconditioned gradient $\tilde{\mathbf{H}}^{-1}\nabla f$ would then be fed into Nesterov's optimizer [8] to iteratively update the placement solution.

3.3 Density Multipliers Update

The phase prior to the complete spreading of all instances is crucial for wirelength optimization. Consequently, we must meticulously adjust the density weight individually for each electrostatic field. This is accomplished by setting density multipliers λ_c, which regulate the spreading efforts across different cell types. Specifically, the initial density multipliers $\boldsymbol{\lambda}^0$ are defined as follows:

$$\boldsymbol{\lambda}^0 = \zeta \frac{\|\nabla \tilde{W}(\boldsymbol{x}^0,\boldsymbol{y}^0)\|_1}{\sum_{i\in\mathcal{V}} q_i \|\xi_i^0\|_1}(1,1,\ldots,1)^\top. \quad (8)$$

Here $(\boldsymbol{x}^0,\boldsymbol{y}^0)$ denotes the initial placement, ξ_i^0 represents the initial electric field of instance i, ζ is a weighting parameter, and $\|\cdot\|_1$ signifies the $L1$-norm. Based on Equation (8), $\boldsymbol{\lambda}^0$ is essentially an $|\mathcal{C}|$-dimensional vector, where $|\mathcal{C}|$ is the number of cell types. Notably, we commence with identical density multipliers for all cell types.

Given that the dual function, $Z(\boldsymbol{\lambda}) = \max f(\boldsymbol{x},\boldsymbol{y})|_{\boldsymbol{\lambda}}$, associated with Equation (5) is not smooth but piecewise linear [16], we utilize the subgradient method to update $\boldsymbol{\lambda}$. Specifically, a subgradient is a simple extension of the gradient for a nonsmooth objective function [17]. For a convex function $f:\mathbf{R}^n \to \mathbf{R}$, a subgradient at $\boldsymbol{x}$ is a vector $\boldsymbol{g}_{\text{sub}}(\boldsymbol{x}) \in \mathbf{R}^n$ such that $f(\boldsymbol{y}) \geq f(\boldsymbol{x}) + \boldsymbol{g}_{\text{sub}}(\boldsymbol{x})^\top(\boldsymbol{y}-\boldsymbol{x})$ for all $\boldsymbol{x} \in \mathbf{R}^n$. In accordance with Equation (5), the subgradient of $\boldsymbol{\lambda}$ is defined as follows:

$$\boldsymbol{g}_{\text{sub}}(\boldsymbol{\lambda}) = \Big(\ldots, \Phi_c(\boldsymbol{x},\boldsymbol{y}) + \frac{1}{2}\mu\theta_\lambda \Phi_c(\boldsymbol{x},\boldsymbol{y})^2, \ldots\Big),\ c \in \mathcal{C}. \quad (9)$$

Given the $\boldsymbol{\lambda}^k$ and the step size α^k at iteration k, the subgradient method updates $\boldsymbol{\lambda}^{k+1}$ as,

$$\boldsymbol{\lambda}^{k+1} \leftarrow \min(\boldsymbol{\lambda}_{\max}, \max(0, \boldsymbol{\lambda}^k + \alpha^k \boldsymbol{g}_{\text{sub}}(\boldsymbol{\lambda}))), \quad (10)$$

where α^k represents the step size at k-th iteration and $\boldsymbol{\lambda}_{\max}$ denotes the maximum density weight. During each iteration, a step is taken from the current point $\boldsymbol{\lambda}^k$ in the direction of the subgradient, aiming to approach the optimal point $\boldsymbol{\lambda}^\star$. The convergence of the subgradient algorithm can be guaranteed with a proper selection of step size [17]:

$$0 < \alpha^k < \frac{2(Z^\star - Z(\boldsymbol{\lambda}^k))}{\|\boldsymbol{g}_{\text{sub}}(\boldsymbol{f}^k)\|_1^2}, \quad (11)$$

where $Z^\star$ represents the optimal Lagrangian dual value. However, in practice, the traditional subgradient model may diverge due to the unavailability of $Z^\star$. To address this issue, we employ the surrogate subgradient method [18], which ensures convergence to $\boldsymbol{\lambda}^\star$ and consequently to $Z^\star$. The surrogate subgradient method is based on

Algorithm 1 Density weight update for multi-electrostatics system

Input: Electric potential energy function Φ, wirelength function $\tilde{W}$, quadratic penalty weighting coefficient μ, density weight step size lower bound α_l,and maximum iteration T;

```
1: Initialize $\boldsymbol{\lambda}^0 = \zeta \frac{\|\nabla \tilde{W}(\boldsymbol{x}^0,\boldsymbol{y}^0)\|_1}{\sum_{i\in\mathcal{V}} q_i\|\xi_i^0\|_1}(1,1,\ldots,1)^\top$;        ▷ Eq. (8)
2: Set $\alpha^0 = (\alpha_l - 1)\|\Phi^0 + \frac{1}{2}\mu\theta_\lambda(\Phi^0)^2\|_2$ ;
3: for $t \leftarrow 1,\ldots,T$ do
4:     $g^t \leftarrow \Phi^t + \frac{1}{2}\mu\theta_\lambda(\Phi^t)^2$;        ▷ Eq. (9)
5:     $\boldsymbol{\lambda}^{t+1} \leftarrow \min(\boldsymbol{\lambda}_{\max}, \max\{0, \boldsymbol{\lambda}^k + \alpha^k \boldsymbol{g}_{\text{sub}}(\boldsymbol{\lambda})\})$;        ▷ Eq. (10)
6:     $\alpha^{k+1} \leftarrow \eta^k \frac{\alpha^k \|\boldsymbol{g}_{\text{sub}}(\boldsymbol{f}^k)\|_1}{\|\boldsymbol{g}_{\text{sub}}(\boldsymbol{f}^{k+1})\|_1}$ ;        ▷ Eq. (14)
7: end for
```

adjusting stepsizes to progressively reduce the distances between density multipliers $\boldsymbol{\lambda}^k$ at consecutive iterations, i.e.,

$$\|\boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^k\|_1 = \eta^k \|\boldsymbol{\lambda}^k - \boldsymbol{\lambda}^{k-1}\|_1, \quad (12)$$

where $0 < \eta^k < 1$, then the stepsize formula can be derived using Equation (10). Indeed, Equation (10) and Equation (12) imply

$$\|\alpha^k \boldsymbol{g}_{\text{sub}}(\boldsymbol{f}^k)\|_1 = \eta^k \|\alpha^{k-1}\boldsymbol{g}_{\text{sub}}(\boldsymbol{f}^{k-1})\|_1. \quad (13)$$

With $\|\boldsymbol{g}_{\text{sub}}(\boldsymbol{f}^k)\|_1 > 0$, we have

$$\alpha^k = \eta^k \frac{\alpha^{k-1}\|\boldsymbol{g}_{\text{sub}}(\boldsymbol{f}^{k-1})\|_1}{\|\boldsymbol{g}_{\text{sub}}(\boldsymbol{f}^k)\|_1}. \quad (14)$$

The distance between density multipliers at consecutive iterations consistently decreases regarding Equation (12), leading to the convergence of $\boldsymbol{\lambda}^k$ and the reduction of step size α^k towards zero. To prevent premature termination of algorithms, we ensure that η^k remains sufficiently close to 1, avoiding excessively fast reduction of step sizes. The surrogate subgradient algorithm to update the density weight $\boldsymbol{\lambda}$ is shown in Algorithm 1.

3.4 BestChoice Clustering

In contrast to traditional NIMH cell placement approaches, where regions are defined after placement, our method updates row regions iteratively based on the current cell locations. To achieve this, we propose a modified version of the BestChoice clustering strategy [19], which clusters cells hierarchically while considering NIMH constraints. Specifically, we aim to avoid placing cells with different heights together in a row, as this would result in wasted area due to spacing and breaker cells, as specified by constraints $C4$, $C5$, and $C6$. Therefore, we prioritize clustering cells of the same type together, which helps reduce wasted area for further wirelength optimization. To achieve this, we introduce pseudo-nets for cells with the same height. If cells i and j have the same height and their Manhattan distance is within a user-defined threshold, we add a pseudo-net between them. In this manner, each cluster exclusively comprises cells of the same height, resulting in an implicit placement with closer proximity and subsequently reducing wasted area for further wirelength optimization. The score function $c(i,j)$ in the modified BestChoice clustering algorithm is defined as:

$$c(i,j) = \sum_{e\in E_{i,j}} \frac{\omega_e}{a_i + a_j}, \quad (15)$$

where e is a hyperedge (including real nets and pseudo-net) connecting cells i and j, a_i and a_j denote the areas of cells i and j respectively, and ω_e is a corresponding edge weight defined as:

$$\omega_e = \begin{cases} e^{\frac{1}{|x_i - x_j| + |y_i - y_j|}}, & h_i = h_j \text{ and e is a real net,} \\ 1, & h_i = h_j \text{ and e is a pseudo-net,} \\ 0, & h_i \neq h_j. \end{cases} \tag{16}$$

Here $|x_i - x_j| + |y_i - y_j|$ represents the Manhattan distance between cell instances i and j. The exponential term serves to scale the fractional value, while h_i and h_j denote the heights of cell i and j, respectively. As per Equation (15), the clustering score of two objects is directly proportional to the total sum of the edge weights connecting them, and inversely proportional to the sum of their areas. For a given cell i, we define the closest object, denoted as n_i, to be the neighboring object with the highest clustering score relative to i. Consequently, n_i will be clustered with i, and the circuit netlist will be updated accordingly. The result of this process is that each row cluster will only contain cells of the same height, in compliance with the NIMH constraints, as this score function implicitly positions cells with the same height closer together.

Another critical parameter in the BestChoice clustering is the number of clusters. Given the constraint $C2$, which requires an even number of cell rows in each fixed region of a specific cell height, we limit the area of each region to be a multiple of double rows. Given the netlist information, the aspect ratio R, the utilization rate U, and the total cell area A, we can determine the ideal chip width and height using the following equations:

$$W^\star \times H^\star \times U = A, \tag{17}$$

$$W^\star \times R = H^\star. \tag{18}$$

Given that we know the exact height of each cell type, we can determine the appropriate number of double-row regions, or in other words, the number of clusters, for each cell type as follows:

$$k_c = \left\lceil \frac{A_c}{W^\star \times H_c \times U \times 2} \right\rceil, \quad \forall c \in \mathcal{C}, \tag{19}$$

where k_c denotes the number of clusters for cell type c, A_c represents the total area of cell type c, and H_c is the height of c. Subsequently, we can execute the proposed NIMH-aware clustering methods to dynamically generate row regions for each cell type.

3.5 Stop Criterion

Global placement usually terminates when the overlap is small enough [8], which is measured by density overflow. The global density overflow is given by,

$$\text{OVFL} = \frac{\sum_{b \in B} \max(\rho^b - \hat{\Phi}, 0)}{\sum_{i \in \mathcal{V}} a_i}, \tag{20}$$

where a_i represents the area of movable cells, ρ_b denotes the normalized charge density of grid b caused solely by movable cells, and $\hat{\Phi}$ represents the local target density. The size of B is typically 512×512 or 1024×1024. Traditionally, the global density overflow serves as the stopping criterion for global placement optimization. However, due to the varying convergence speeds of different electrostatic systems, we adopt an individual stop criterion for each electric field based on its specific density overflow as follows,

$$\text{OVFL}_c = \frac{\sum_{b \in B_c} \max(\rho_c^b - \hat{\Phi}_c, 0)}{\sum_{i \in \mathcal{V}_c} a_i}. \tag{21}$$

Here ρ_c^b is the normalized charge density map for bin b with cell type c and $\hat{\Phi}_c$ represents the local target density for cell type c. Once the density overflow of the region runs below the pre-defined stop overflow, e.g., 0.07, we freeze the movable instances as well as the movable filler instances to avoid divergence. By freezing these instances, the gradient from the wirelength and density objectives will only affect the unfrozen instances.

3.6 Row-based Legalization

Following global placement, the final step involves legalizing cells in each row region. The primary objective of this stage is to relocate each cell to a row of the same height, while minimizing the total displacement. In particular, we can employ conventional intra-row legalization methods, such as Abacus [11], given the similarity in row structure. We iteratively carry out row-based legalization for each row region to obtain the final placement solution. Furthermore, the row structure allows for easy insertion of vertical spacing between row regions of different heights during the floorplan update stage, effectively satisfying constraint $C5$.

Figure 4 illustrates the heterogeneous spreading process in our multi-electrostatics-based placement for NIMH cells. As evident in Figure 4(a), our updating scheme for the multi-electrostatics system can greatly preserve those natural physical clusters consisting of heterogeneous cell instances. The placement right after clustering is shown in Figure 4(b), from which we can observe that cells are roughly scattered in corresponding row regions. The near-optimal placement result before legalization shown in Figure 4(c) further underscores our method's capability to achieve nearly overlap-free solutions without explicit legalization. Figure 4(d) visualizes the legal solution after row-based legalization.

4 EXPERIMENTAL RESULTS

To demonstrate the efficacy of our algorithm, which is based on multi-electrostatics for NIMH cells, we have conducted an evaluation of WNS, TNS, HPWL, and CPU runtime using the OpenCores [7] benchmark suite. Our results are then compared with those of another row-based NIMH placement algorithm [6]. It's important to underscore that in this study, the placement engine utilized is the *Cadence Innovus Implementation System*. The runtime indicated in the referenced paper pertains solely to the duration of the k-means-based legalization process. Consequently, a comparison of the legalization runtime with our proposed placement flow, which includes a specific optimization on global placement, would not constitute a fair assessment. Hence, we choose to compare the runtime of the complete placement flow, encompassing both global placement and legalization.

4.1 Experiment Settings

We modified the 15nm NanGate Open Cell Library [20] to include information on both 8T and 12T cells. The original cells in this library

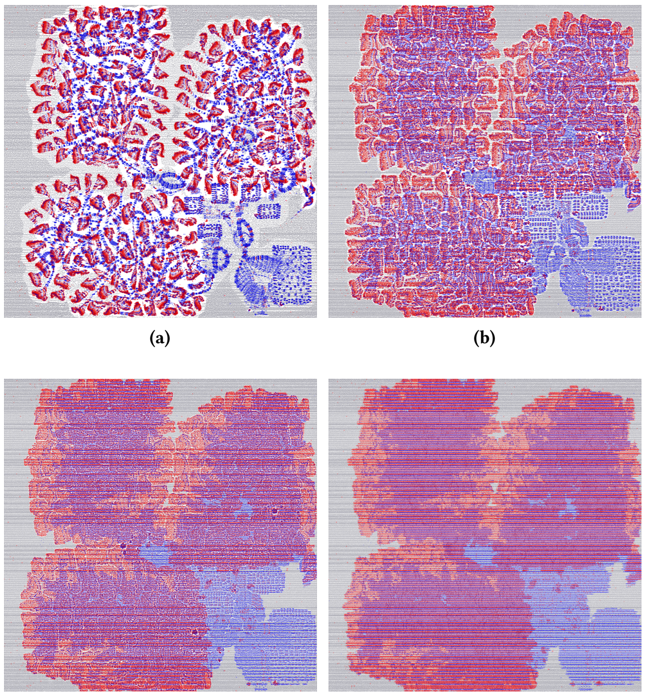


Figure 4: The multi-electrostatics-based placement process for NIMH cells where blue boxes denote 8T cells, red boxes represent 12T cells and grey rectangles denote fillers. (a) An intermediate placement before clustering. (b) An intermediate placement after clustering. (c) The near-optimal placement result before legalization. (d) The placement solution after legalization.

have a height of 0.512 μm, corresponding to 8T cells. We artificially created another version of cells with a height of 12T. Both 8T and 12T cells were used for logic synthesis. The delay-area trade-off between 8T and 12T buffers/inverters in our modified cell libraries is depicted in Figure 5. In the figure, 8T cells are represented in blue, while 12T cells are shown in red. The figure demonstrates that 12T cells generally have a larger cell area but a shorter delay time compared to 8T cells. On the other hand, 8T cells have a smaller area, resulting in lower power consumption, but with a longer cell delay compared to 12T cells.

Our multi-electrostatic system is implemented on the open-source VLSI placer `DREAMPlace` [21], which includes a global placement optimizer and a legalizer. All experiments were conducted on a Linux server equipped with an Intel(R) Xeon(R) Silver 4210R CPU running at 2.40GHz and 128 GB of RAM.

We conducted experiments using eight design blocks (`sha3`, `aes_core`, `des`, `fpu`, `des3`, `mor1kx`, `jpeg`, `aes_128`) obtained from the OpenCores [7] website. These designs range from thousands to hundreds of thousands of nets. The characteristics of these test cases are summarized in Table 2. In the table, "#Cells", "#Nets", "Util", and "Clock" represent the number of cells, number of nets, chip utilization rate, and clock period respectively. The designs were synthesized using *Cadence Genus Synthesis Solution 20.11-s111_1* [22], with both 8T and 12T cell libraries. For each design, we carefully

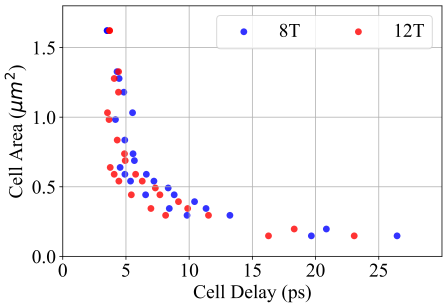


Figure 5: The delay-area trade-off between 8T and 12T buffers/inverters in our modified technology libraries.

Table 2: Statistics of the OpenCores benchmarks

Design	#Cells	#Nets	Util (%)	Clock (ps)
sha3	1337	1397	69.05	100
aes_core	4733	4808	69.84	400
des	18274	18372	67.11	250
fpu	30495	31225	67.65	270
des3	58017	58116	67.05	250
mor1kx	61220	58952	67.32	200
jpeg	210968	233898	68.49	300
aes_128	250672	225888	57.29	300

determined the clock period at which the number of 8T cells and 12T cells were nearly balanced after logic synthesis.

4.2 Comparison with State-of-the-Art Method

To evaluate the effectiveness of our method, we compare it with the state-of-the-art row-based placement algorithm, referred to as "ICCAD'21-imp" [6], for NIMH cells. After obtaining the synthesized netlists, we execute our proposed placement algorithm to generate a global placement solution with optimized wirelength. For "ICCAD'21-imp", we utilize the *Cadence Innovus v20.14-s095_1* [23] placement engine to generate the initial global placement, as mentioned in [6]. After global placement, we use a row-based legalized in `DREAMPlace` [21] to obtain the final placement results without violating the NIMH constraints. We also implement the k-means-based legalization method to obtain the legal placement results for "ICCAD'21-imp". The experimental results are presented in Table 3, where "WNS (ns)" represents the worst negative slack in nanoseconds, "TNS (ns)" indicates the total worst negative slack in nanoseconds, "HPWL (10^5um)" denotes the total wirelength in 10^5 micrometers, and "Runtime (s)" represents the total CPU runtime in seconds. It is important to note that the timing information, including WNS and TNS, is reported by *Cadence Innovus v20.14-s095_1* [23] for both "ICCAD'21-imp" and our method.

The experimental results presented in Table 3 demonstrate the superior performance of our method compared to "ICCAD'21-imp" [6]. On average, our method achieves a 12% reduction in HPWL while exhibiting a remarkable 23.5× faster runtime. Notably, in large cases involving over 200,000 standard cells, our method achieves an impressive speedup of up to 42.85× while delivering better placement

Table 3: WNS (ns), TNS (ns), HPWL (10^5um) and CPU Runtime (s) with State-of-the-art Row-based Placers.

test case	Cells			ICCAD’21-imp [6]				Ours			
	8T	12T	Total	WNS (ns)	TNS (ns)	HPWL (10^5um)	Runtime (s)	WNS (ns)	TNS (ns)	HPWL (10^5um)	Runtime (s)
sha3	662	675	1337	-0.09	-1.89	3.96	45.3	-0.09	-1.85	3.25	23.07
aes_core	2511	2222	4733	-0.15	-22.16	4.38	78.36	-0.14	-19.14	3.76	23.76
des	8853	9421	18274	0.11	0	14.50	218.90	0.11	0	12.21	27.58
fpu	15266	15229	30495	-0.22	-4.37	25.14	315.88	-0.17	-3.28	23.63	31.26
des3	29683	28334	58017	-0.05	-0.20	46.78	564.51	-0.04	-0.11	37.85	33.58
mor1kx	30168	29873	60041	-0.13	-4.10	61.22	808.22	-0.13	-4.49	63.20	32.86
jpeg	107866	103102	210968	-0.48	-128.20	152.66	2528.35	-0.36	-66.10	149.67	59.15
aes_128	123825	126847	250672	-0.32	-61.97	221.90	2569.14	-0.25	-54.33	178.30	72.03
average ratio				1.22	1.49	1.12	23.50	**1.00**	**1.00**	**1.00**	**1.00**

solution quality (see the “jpeg” case in Table 3), thereby demonstrating the scalability and effectiveness of our proposed method.

Furthermore, our approach demonstrates substantial enhancements in TNS and WNS, which validate the high solution quality achieved by our placement flow. Notably, even though timing metrics were not explicitly considered in our placement flow, our method consistently outperforms in terms of timing performance. On average, we observe improvements of 22% and 49% in WNS and TNS, respectively. These enhancements can be attributed to the reduction in wirelength, leading to reduced wire capacitance and resistance, ultimately resulting in improved timing performance. These results solidify the superiority of our placement solution. Overall, the experimental findings conclusively demonstrate the effectiveness of our proposed algorithm in effectively addressing the placement problem with NIMH cells.

5 CONCLUSION

In this paper, we present an analytical placer to consider circuit designs with NIMH cells and additional layout constraints. We propose a multi-electrostatics-based method to handle heterogeneous cells. The proposed ALM along with our preconditioning technique and surrogate subgradient method enables robust convergence for multi-electrostatic systems. A significant contribution of our research lies in the development of an adaptive row region generation approach, i.e. the NIMH-aware clustering technique, for different types of cells. This innovative method seamlessly integrates with the placement optimization flow, resulting in enhanced solution quality when compared to previous approaches. The experimental results demonstrate the effectiveness and efficiency of our proposed algorithm. On the OpenCores benchmark suites, we achieve over 12% improvement in HPWL with more than 23× speedup. Additionally, our method showcases approximately 22% and 49% better performance in terms of WNS and TNS, respectively. These results highlight the effectiveness of our placement method in handling the placement problem with NIMH standard cells.

ACKNOWLEDGEMENT

This work is partially supported by The Research Grants Council of Hong Kong SAR (No. CUHK14210723).

REFERENCES

[1] E. Sicard and L. Trojman, “Introducing 5-nm finfet technology in microwind,” 2021.
[2] M. Hatamian and P. Penzes, “Non-integer height standard cell library,” US Patent 8 788 998, 2014.
[3] S.-Y. Wu, C. Chang, M. Chiang, C. Lin, J. Liaw, J. Cheng, J. Yeh, H. Chen, S. Chang, K. Lai *et al.*, “A 3nm CMOS FinFlex™ Platform Technology with Enhanced Power Efficiency and Performance for Mobile SoC and High Performance Computing Applications,” in *Proc. IEDM*, 2022, pp. 27–5.
[4] S. A. Dobre, A. B. Kahng, and J. Li, “Design implementation with noninteger multiple-height cells for improved design quality in advanced nodes,” *IEEE TCAD*, vol. 37, no. 4, pp. 855–868, 2017.
[5] J. Chen, Z. Huang, Y. Huang, W. Zhu, J. Yu, and Y.-W. Chang, “An efficient epist algorithm for global placement with non-integer multiple-height cells,” in *Proc. DAC*, 2020, pp. 1–6.
[6] Z.-Y. Lin and Y.-W. Chang, “A row-based algorithm for non-integer multiple-cell-height placement,” in *Proc. ICCAD*, 2021, pp. 1–6.
[7] “Opencores,” https://opencores.org/.
[8] J. Lu, P. Chen, C.-C. Chang, L. Sha, D. J.-H. Huang, C.-C. Teng, and C.-K. Cheng, “ePlace: Electrostatics-based placement using fast fourier transform and Nesterov’s method,” *ACM TODAES*, vol. 20, no. 2, pp. 1–34, 2015.
[9] W. Li, Y. Lin, and D. Z. Pan, “elfPlace: Electrostatics-based placement for large-scale heterogeneous FPGAs,” in *Proc. ICCAD*. IEEE, 2019, pp. 1–8.
[10] 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.
[11] P. Spindler, U. Schlichtmann, and F. M. Johannes, “Abacus: Fast legalization of standard cell circuits with minimal movement,” in *Proc. ISPD*, 2008, pp. 47–53.
[12] 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 TCAD*, vol. 32, no. 4, pp. 497–509, 2013.
[13] G. Di Pillo and S. Lucidi, “An augmented lagrangian function with improved exactness properties,” *SIAM Journal on Optimization*, vol. 12, no. 2, pp. 376–406, 2002.
[14] M.-C. Kim and I. L. Markov, “ComPLx: A competitive primal-dual lagrange optimization for global placement,” in *Proc. DAC*, 2012, pp. 747–752.
[15] E. Turkel, “Preconditioning techniques in computational fluid dynamics,” *Annual Review of Fluid Mechanics*, vol. 31, no. 1, pp. 385–416, 1999.
[16] C. Lemaréchal, “Lagrangian relaxation,” *Computational combinatorial optimization: optimal or provably near-optimal solutions*, pp. 112–156, 2001.
[17] D. Bertsekas, *Nonlinear Programming*. Athena Scientific, 2003.
[18] M. A. Bragin, P. B. Luh, J. H. Yan, N. Yu, and G. A. Stern, “Convergence of the surrogate lagrangian relaxation method,” *Journal of Optimization Theory and applications*, vol. 164, pp. 173–201, 2015.
[19] G.-J. Nam, S. Reda, C. J. Alpert, P. G. Villarrubia, and A. B. Kahng, “A fast hierarchical quadratic placement algorithm,” *IEEE TCAD*, vol. 25, no. 4, pp. 678–691, 2006.
[20] M. Martins, J. M. Matos, R. P. Ribas, A. Reis, G. Schlinker, L. Rech, and J. Michelsen, “Open cell library in 15nm freepdk technology,” in *Proc. ISPD*, 2015, pp. 171–178.
[21] Y. Lin, S. Dhar, W. Li, H. Ren, B. Khailany, and D. Z. Pan, “DREAMPlace: Deep learning toolkit-enabled GPU acceleration for modern VLSI placement,” in *Proc. DAC*, 2019, pp. 1–6.
[22] C. D. S. Incorporation., “Genus synthesis solution,” 2020.
[23] C. D. Systems., “Innovus user guide v20.14,” 2020.