Ultrafast Density Gradient Accumulation in 3D Analytical Placement with Divergence Theorem

Peiyu Liao†, Yuxuan Zhao†, Siting Liu, Bei Yu
The Chinese University of Hong Kong

***Abstract*—Density gradient accumulation plays a pivotal role in 3D analytical placement. Analytical placers rely on this fundamental operation during the backward step of each iteration to compute the gradient of the density penalty for every node. This primitive operation thus constitutes a significant runtime bottleneck, especially for mixed-size designs with large macros. Furthermore, this bottleneck becomes increasingly critical as the grid size in 3D placement is considerably larger than that in conventional 2D placement. In this paper, we propose an algorithm inspired by the divergence theorem to reduce the time complexity of density gradient accumulation. We also present our implementations of this algorithm for both CPU and GPU versions. Experimental results demonstrate that our method achieves more than 3× end-to-end runtime speedup on CPU and GPU compared to the SOTA analytical 3D placer.**

I. Introduction

Density accumulation is a frequently employed technique in physical design applications, particularly in the contexts of placement and routing. Generally, the *forward* category accumulates the contributions to the density map given a set of geometric shapes, while the *backward* category accumulates weights from a grid map back into the shapes [1]. In modern analytical placement, the weights to be accumulated in the backward pass are usually gradients of the density objective with respect to certain geometric shapes [2]–[7], and therefore it is often called *density gradient accumulation* in the backward computation of analytical placement.

For example, a nonlinear global placement is typically formulated as an unconstrained optimization problem

$$\min_{\mathbf{v}} f(\mathbf{v}) := W(\mathbf{v}) + \lambda U(\mathbf{v}), \tag{1}$$

where $\mathbf{v}$ is the locations to be optimize, $W(\mathbf{v})$ is any differentiable wirelength function. The density term $U(\mathbf{v})$ in above equation measures the overall overlap of the entire system so that the optimizer could try to even out all nodes to be placed. Modern analytical placers may utilize different density models, and thus the gradient $\nabla U(\mathbf{v})$ may have various forms. However, the forward pass to acquire the density map before computing the gradients and the backward pass to accumulate the gradient map are always essential.

The ePlace family [5], [8], [9] has utilized electrostatics systems to describe the density model for conventional 2D placement. Since the emergence of the ePlace family [5], [8], [9] in the 2010s, modern placers [7], [10] necessitate a substantial number of iterations to achieve convergence. Each iteration involves at least one forward and backward operation, resulting in a significant number of density accumulation calls. Hence, the implementation quality of density accumulation has a substantial impact on the overall efficiency of solving the global placement problem.

Recently, 3D integrated circuits have become a promising solution for maintaining the trajectory of Moore's Law, thereby enhancing the appeal of 3D placement for designers. The ePlace family incorporates the well-known ePlace-3D [6] as a viable solution for 3D analytical global placement. At the time of its proposal, the ePlace-3D [6] framework was at the forefront of innovation, utilizing the capabilities of differentiable optimization to simultaneously determine partitioning and cell locations during the optimization process. The 3D placement formulation generalizes Equation (1) to 3D contexts, wherein both $W(\mathbf{v})$ and $U(\mathbf{v})$ functions are specifically devised for 3D variables. Recent advancements in 3D analytical placement [11]–[14] build upon the methodology of ePlace-3D to model the 3D density objective $U(\mathbf{v})$. The execution of forward and backward operations within the 3D density model [6] necessitates intensive computational effort on 3D grid maps. These grid maps may encompass a substantial number of bins, approaching 100 million, thereby rendering the density accumulation a computational bottleneck. Furthermore, the density gradient accumulation during the backward pass requires the computation of gradients across three dimensions. This requirement compels the `DCT` family to calculate three gradient maps [6] in each iteration. Consequently, the backward operation may assume a more pivotal role in 3D placement.

The challenges of accelerating density gradient accumulation are summarized [1] into the following categories: the large threading overhead for primitive operations of geometric shape updating, and the workload imbalance due to heterogeneous sizes. Some previous works [7], [15], [16] have studied the parallelization schemes for 2D density accumulation on both CPU and GPU. However, these parallelized computation techniques are devised mainly for the forward pass on a cell placement flow. Guo *et al.* [1] propose efficient CPU/GPU kernels for both the forward and backward operations of mixed-sized placement problems with 2D prefix sum. PeF [17] and Pplace-MS [18] developed a potential-driven density force that obviates the need for explicit computation of electric fields for 2D fixed-outline floorplanning and 2D mixed-size global placement, respectively. Consequently, they can bypass two `IDCT/IDXST` operations for computing the electric fields,

†These authors contributed equally.

thereby accelerating the density gradient accumulation at the methodological level.

The workload imbalance challange becomes increasingly critical in 3D mixed-size placement, as 3D shapes that span a large number of grids generally result in the performance degradation of primitive operations. Zhao *et al.* [14] generalize the 2D prefix sum scheme to 3D scenarios with a rigorous and formal theoretical statement, achieving a 2.55× runtime speedup on 3D global placement compared to the vanilla 3D density accumulation [12] which extends the 2D density accumulation scheme of Lin *et al.* [7].

In this paper, we propose an ultrafast algorithm for density gradient accumulation in 3D analytical placement, utilizing the well-known divergence theorem. We theoretically demonstrate that the accumulation of density gradients within a region is equivalent to the accumulation of the potential map on the region's boundary, thereby reducing the dimensionality of the accumulation process. Experiments on the ICCAD 2023 benchmarks [19] for 3D mixed-size placement demonstrate that our method achieves a 3.3× and 4.0× *end-to-end* runtime speedup on CPU and GPU, respectively, compared to the state-of-the-art (SOTA) analytical 3D placer [14].

The rest of this paper is organized as follows. Section II provides an brief introduction to the preliminaries. Section III details the algorithms and theoretical analysis of the proposed method. Section IV presents the experimental results along with related analyses on the adopted benchmarks, followed by the conclusion in Section V.

II. Preliminaries

A. Electrostatics-based 3D Density Model

Given a netlist $\mathcal{N} = (V, E)$ where notations $V = \{v_1, \cdots, v_n\}$ and $E = \{e_1, \cdots, e_m\}$ are the instance set and the net set, respectively, the 3D analytical placement problem aims at solving the non-convex optimization problem [6] specified in Equation (1), where $W(\mathbf{v})$ stands for arbitrary differentiable wirelength function designed for 3D analytical placement [11]–[14]. Each instance $v_i \in V$ is assigned a depth d_i so that it is represented by a cuboid D_{v_i} of size (w_i, h_i, d_i) in the 3D Euclidean space $\mathbb{R}^3$.

The ePlace-3D framework [6] models the total density penalty $U(\mathbf{v})$ of an **electrostatics** system. The electrostatics field is composed of n instances $v_1, \cdots, v_n \in V$, each of which is modeled as a cuboid D_{v_i} with a uniformly distributed charge. Our target is to optimize the total potential energy $U(\mathbf{v})$ so that instances can be evenly dispersed within the 3D region Ω, achieving the dual goal of optimizing both partitioning and instance position coordinates.

Describing the electrostatic field is a subject that has been extensively examined and studied in physics. Neglecting the vacuum permittivity ϵ, the differential form of the famous Gauss's law (one of Maxwell's equations) directly describes this electrostatic field as a Poisson's equation.

$$\begin{aligned} \Delta\phi &= -\rho, \quad \text{in } \Omega \\ \hat{\mathbf{n}} \cdot \nabla\phi &= 0, \quad \text{on } \partial\Omega, \end{aligned} \tag{2}$$

and then the corresponding total potential energy of this electrostatics system is defined by the following integral,

$$U = \frac{1}{2} \iiint_{\Omega} \rho\phi \, \mathrm{d}\Omega, \tag{3}$$

where functions $\rho = \rho(\mathbf{v}) : \mathbb{R}^3 \to \mathbb{R}$ and $\phi := \phi(\mathbf{v}) : \mathbb{R}^3 \to \mathbb{R}$ represent the electric density map and electric potential map given a placement solution $\mathbf{v}$, respectively. The Neumann boundary condition in Equation (2) is a manual setting [6] to restrict the sytem to be "stable" on the boundary $\partial\Omega$.

The representation in Equation (3) is interpreted in a continuous manner, and it can be separated into several integrals of potential maps as the system is composed of finite cuboids including both macros and standard cells. Given a placement solution $\mathbf{v}$, the density function $\rho : \mathbb{R}^3 \to \mathbb{R}$ is defined by

$$\rho(x, y, z) = \sum_{i=1}^{n} \chi_{D_{v_i}}(x, y, z), \tag{4}$$

where $\chi_{D_{v_i}}(x, y, z)$ is the indicator function of D_{v_i} which takes value 1 when $(x, y, z) \in D_{v_i}$ and 0 otherwise. Since $\rho = \rho(\mathbf{v})$ is a function on Ω, it is represented discretely by a grid map $\hat{\rho}$ defined only on grid bins B. Given a placement $\mathbf{v}$, the process of accumulate Equation (4) to $\hat{\rho}(\mathbf{v})$ is called the forward density accumulation, or *density map accumulation*. Once the grid map $\hat{\rho}$ is computed, the electric potential grid map $\hat{\phi}$ and the electric field maps $\hat{\mathbf{E}}$ are solved by `DCT/IDCT` families accordingly [6],

$$\begin{aligned} a_{jkl} &= \frac{1}{|B|} \sum_{x,y,z} \rho(x, y, z) \cos(\omega_j x) \cos(\omega_k y) \cos(\omega_l z), \\ \phi &= \sum_{j,k,l} \frac{a_{jkl}}{\omega_j^2 + \omega_k^2 + \omega_l^2} \cos(\omega_j x) \cos(\omega_k y) \cos(\omega_l z), \\ E_x &= \sum_{j,k,l} \frac{a_{jkl}\omega_j}{\omega_j^2 + \omega_k^2 + \omega_l^2} \sin(\omega_j x) \cos(\omega_k y) \cos(\omega_l z), \end{aligned} \tag{5}$$

where $|B|$ is the total number of grid bins, and $(\omega_j, \omega_k, \omega_l)$ tuple is the frequency indices. More details are discussed in several previous works [6], [11]–[14]. For any node instance $v_i \in V$, the process of accumulating field maps into gradient ∇U is called the *density gradient accumulation*.

B. Density Gradient Accumulation

The electrostatics-based 3D density model [6] describes the density penalty U as the potential energy of the system of n cuboid charges. In classical electromagnetism, an electric charge q within an electric field $\mathbf{E}$ that is independent of q itself receives the electric force F is equal to qE, however, the situation becomes much more complex if $\mathbf{E}$ is dependent on q because $\mathbf{E}$ can no longer be considered a fixed vector field. Omitting such effects for convenience of analysis and implementations, the electric force of $v_i \in V$ is treated as a value proportional to the following integral

$$\frac{\partial U}{\partial x_i} \propto \iiint_{D_{v_i}} \frac{\partial \phi}{\partial x} \, \mathrm{d}D_{v_i} = \iiint_{D_{v_i}} -E_x \, \mathrm{d}D_{v_i}. \tag{6}$$

Hence, the **vanilla** approach of density gradient accumulation is to traverse all bins that have overlap with node v_i, which is equivalent to a numerical approximation to Equation (6).

Problem 1 (Density Gradient Accumulation)**.** *Given the instance set $V = \{v_1, \cdots, v_n\}$ with geometric cuboids D_{v_i} for every i, an $N_x \times N_y \times N_z$ grid bins B, the electric field $\mathbf{E}$. The density gradient accumulation in x-direction is to compute the following gradient*

$$\frac{\partial U}{\partial x_i} \approx \sum_{b \in B(D_{v_i})} -\mu(D_{v_i} \cap b) E_x(b), \tag{7}$$

for any i, where $\mu(D_{v_i} \cap b)$ is the overlapped volume of node cuboid D_{v_i} and a grid bin $b \in B$, $B(D_{v_i}) = \{b \in B \mid \mu(D_{v_i} \cap b) > 0\}$ is the set of bins that overlap with D_{v_i}, and $E_x(b)$ is the discretized electric field value at the grid bin b.

The total number of affected bins is directly proportional to the volume of v_i. Contemporary implementations of analytical placers [7] employ node-level parallelism for all instances, encompassing both standard cells and macros. Consequently, the density gradient accumulation for each node v_i according to the vanilla scheme is computed sequentially within the loop through instances. This vanilla approach encounters a significant challenge, as large macros in 3D placement encompass numerous bins, leading to substantial workload imbalance. This imbalance can severely compromise the efficiency of parallelism.

III. Algorithms

The system potential energy in Equation (3) is reasonable to be interpreted by the sum of electric energy of D_{v_i},

$$U = \frac{1}{2} \iiint_\Omega \sum_{i=1}^{n} \chi_{D_{v_i}} \phi \, \mathrm{d}\Omega = \frac{1}{2} \sum_{i=1}^{n} \iiint_{D_{v_i}} \phi \, \mathrm{d}D_{v_i}. \tag{8}$$

The *density gradient* $\nabla U := (\nabla_{\mathbf{x}} U, \nabla_{\mathbf{y}} U, \nabla_{\mathbf{z}} U)$ is defined as the gradient with respect to the node coordinates $\mathbf{v} = (\mathbf{x}, \mathbf{y}, \mathbf{z})$, which will be fed into the optimizer as a part of $\nabla f(\mathbf{v}) = \nabla W(\mathbf{v}) + \lambda \nabla U(\mathbf{v})$, indicating that the computation of density gradient plays a crucial role in placement optimization.

A. Density Gradient Accumulation with Divergence Theorem

Imprecisely speaking, $U_i := \frac{1}{2} \iiint_{D_{v_i}} \phi \, \mathrm{d}D_{v_i}$ in the rightmost term of Equation (8) approximately represents the electric potential energy of v_i iteself. It is decribed as "approximately" because the electric potential ϕ is not solely induced by v_i. Therefore, the system energy is roughly a decomposition into the sum of the potential energy of every node. As introduced in Equation (6), empirically, common practice suggests to calculate $\frac{\partial U_i}{\partial x_i}$ as $\frac{\partial U}{\partial x_i}$ [5], [6], which is approximated by the accumulation of the gradient field $\nabla_{\mathbf{x}} \phi$ over D_{v_i}. Results of $\frac{\partial U}{\partial y_i}$ and $\frac{\partial U}{\partial z_i}$ for the vanilla density gradient accumulation in Equation (7) are similarly established given the electric fields $\mathbf{E} = (E_x, E_y, E_z)$.

From a physics perspective, the rationale behind the vanilla scheme is intuitive: the electric forces acting at each point within D_{v_i} should be accumulated to determine the resultant electric force experienced by v_i in the electric field. This explains the vanilla accumulation described in Equation (7), which serves as an approximation to the actual derivative of potential energy U. We will theoretically discuss the variations in potential energy U as a functional of the potential map ρ in Section III-C, where we also present the exact derivative $\frac{\partial U}{\partial x_i}$ to support our proposed algorithm.

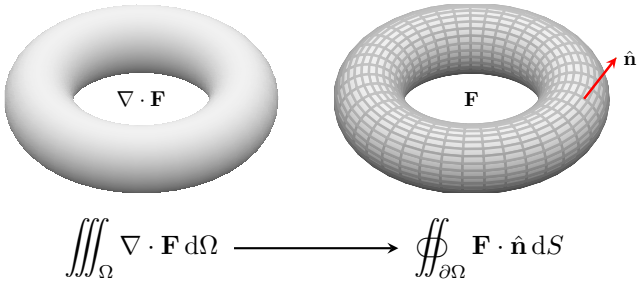


Fig. 1 An illustration of the application of the divergence theorem to a torus in $\mathbb{R}^3$. Specifically, the theorem posits that the integral of the divergence of a vector field $\mathbf{F}$ over the volume of the torus is equal to the integral of the flux of $\mathbf{F}$ through the toroidal surface.

In this subsection, we will demonstrate that the dimensionality of the vanilla approach in Equation (7) is generally reducible via the divergence theorem. The **divergence theorem** in vector calculus, also known as Gauss's theorem [20], or Ostrogradsky's theorem [21], is stated as follows.

Theorem 1 (Divergence Theorem)**.** *Let function $\mathbf{F}$ be a continuously differentiable vector field defined on a neighborhood of compact set $\Omega \subset \mathbb{R}^n$ with a piecewise smooth boundary $S := \partial\Omega$, then we have*

$$\int_\Omega \nabla \cdot \mathbf{F} \, \mathrm{d}\Omega = \oint_{\partial\Omega} \mathbf{F} \cdot \hat{\mathbf{n}} \, \mathrm{d}S, \tag{9}$$

where $\hat{\mathbf{n}}$ is the outward pointing unit normal on $\partial\Omega$.

This theorem is well known to relate the integral of the flux of a field $\mathbf{F}$ through a closed surface or hypersurface to the integral of divergence $\operatorname{div} \mathbf{F} = \nabla \cdot \mathbf{F}$ over the enclosed region. Fig. 1 illustrates an example of applying the divergence theorem to a torus in Euclidean space $\mathbb{R}^3$. It can be utilized to compute the flux through a closed surface that entirely encloses a given volume. In topological terms, the genus of a torus in Fig. 1 is 1, indicating the presence of a single "hole" in its structure. Given that the region of the torus is completely enclosed by its surface, the divergence theorem can be applied to this genus-1 shape. Ordinary cuboids are much simpler to apply this theorem. In our 3D placement application, it is therefore intuitive to transform a volume integral into a surface integral over its 6 faces according to the divergence theorem.

Observing that the dimension of $\partial\Omega$ is always 1 less than the dimension of Ω, we are inspired to reduce the computation complexity by relating the field integral over a region to

the flux integral over its boundary. As a case study of the applications of divergence theorem, we take $\mathbf{F}_x = (\phi, 0, 0)$ in $\mathbb{R}^3$, where ϕ is the electric potential in Equation (7). It is straight-forward to obtain the following results,

$$\iiint_{D_{v_i}} \frac{\partial \phi}{\partial x} \,\mathrm{d}D_{v_i} = \iiint_{D_{v_i}} \nabla \cdot \mathbf{F}_x \,\mathrm{d}D_{v_i} = \oiint_{\partial D_{v_i}} \hat{n}_x \phi \,\mathrm{d}S, \tag{10}$$

where $\hat{n}_x$ is the x-component of the outward pointing unit normal $\hat{\mathbf{n}}$. Results of y, z dimensions can be derived similarly.

Notice that D_{v_i} is always a cuboid with 6 faces. Assume that the cuboid D_{v_i} with sizes (w_i, h_i, d_i) has coordinate $\mathbf{r}_i = (x_i, y_i, z_i)$ in $\mathbb{R}^3$, then the two yz faces can be represented by

$$\begin{aligned} \partial D_{v_i}^{yz+} &= \{x_i + w_i\} \times [y_i, y_i + h] \times [z_i, z_i + d], \\ \partial D_{v_i}^{yz-} &= \{x_i\} \times [y_i, y_i + h] \times [z_i, z_i + d]. \end{aligned} \tag{11}$$

The other 4 faces $D_{v_i}^{xz+}$, $D_{v_i}^{xz-}$, $D_{v_i}^{xy+}$, and $D_{v_i}^{xy-}$ of the cuboid are defined similarly. These 4 faces are parallel to the x-direction, therefore we must have $n_x = 0$ on these faces, leading to a similified form of Equation (10) summarized in the following *corollary* of the divergence theorem.

Corollary 1. *Assume that region D_{v_i} is a cuboid of sizes (w_i, h_i, d_i) and coordinate $\mathbf{r}_i = (x_i, y_i, z_i)$ within an electric field $\mathbf{E} = (E_x, E_y, E_z)$ in $\mathbb{R}^3$. Then, the integral of $\mathbf{E}$ over D_{v_i} can be represented by several double integrals described as follows,*

$$\begin{aligned} \iiint_{D_{v_i}} E_x \,\mathrm{d}D_{v_i} &= \iint_{\partial D_{v_i}^{yz-}} (\phi(x, y, z) - \phi(x + w_i, y, z)) \,\mathrm{d}S, \\ \iiint_{D_{v_i}} E_y \,\mathrm{d}D_{v_i} &= \iint_{\partial D_{v_i}^{xz-}} (\phi(x, y, z) - \phi(x, y + h_i, z)) \,\mathrm{d}S, \\ \iiint_{D_{v_i}} E_z \,\mathrm{d}D_{v_i} &= \iint_{\partial D_{v_i}^{xy-}} (\phi(x, y, z) - \phi(x, y, z + d_i)) \,\mathrm{d}S, \end{aligned} \tag{12}$$

where S represents the respective integral surface.

Proof. We provide a quick proof here. Consider the x-direction in Equation (10). The surface integral is actually over 6 faces of the cuboid. Since we have $\hat{n}_x(\mathbf{r}) = 1$ for any point $\mathbf{r} = (x, y, z) \in \partial D_{v_i}^{yz+}$, $\hat{n}_x(\mathbf{r}) = -1$ for $\mathbf{r} = (x, y, z) \in \partial D_{v_i}^{yz-}$, and $\hat{n}_x(\mathbf{r}) = 0$ for $\mathbf{r}$ on the other 4 faces, we have

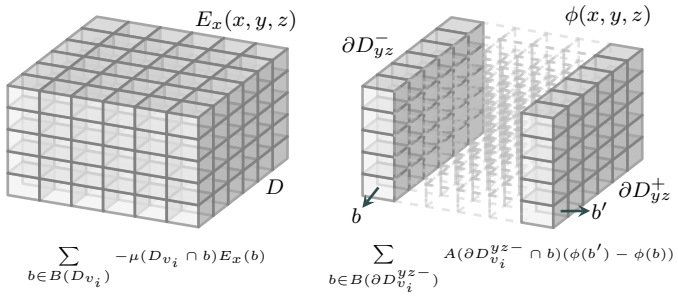
$$\begin{aligned} \oiint_{\partial D_{v_i}} \hat{n}_x \phi \,\mathrm{d}S &= \iint_{\partial D_{v_i}^{yz+}} \phi \,\mathrm{d}S + \iint_{\partial D_{v_i}^{yz-}} -\phi \,\mathrm{d}S \\ &= \iint_{\partial D_{v_i}^{yz-}} (\phi(x + w_i, y, z) - \phi(x, y, z)) \,\mathrm{d}y\mathrm{d}z. \end{aligned} \tag{13}$$

The other two equations are derived similarly. □

Corollary 1 indicates that the vanilla density gradient accumulation has been theoretically proven to be reducible in dimension. Therefore, in practice, we do not need to traverse the entire region, but only its boundary. This is the key idea of utilizing the divergence theorem to significantly reduce the computational complexity of Equation (7). More concretely, let $\mathbf{r}_b = (x_b, y_b, z_b)$ be the center coordinate of any bin box $b \in B$, we have the following discretized computation following Equation (7),

$$\frac{\partial U}{\partial x_i} \approx \sum_{b \in B(\partial D_{v_i}^{yz-})} A(\partial D_{v_i}^{yz-} \cap b) \left(\phi(b') - \phi(b)\right), \tag{14}$$

where the bin boxes $b' \in B(\partial D_{v_i}^{yz+})$ and $b \in B(\partial D_{v_i}^{yz-})$ have the same projection onto the yz-plane, *i.e.*, $(y_{b'}, z_{b'}) = (y_b, z_b)$. We use Equation (14) as our accelerated density gradient accumulation scheme.



(a) Field Accumulation (b) Potential Accumulation

Fig. 2 An example illustrating the density gradient accumulation via the potential map ϕ in Equation (14).

Fig. 2 provides an example of the potential accumulation described by Equation (14). The vanilla approach calculate the results via field accumulation by traversing all bins covered by D_{v_i}, while the potential accumulation only requires values on the two faces orthogonal to x-direction. Assume potential values ϕ within the two faces are not required to calculate the gradient, resulting in a reduction on the time complexity. Besides, the `DCT/IDCT` steps to acquire the three electric fields $\mathbf{E} = (E_x, E_y, E_z)$ are reasonably omitted as they are no longer a part of computation. This advantage also appeared in which acts as a special case of 2D degradation. This advantage is also evident in [18], which serves as a specific instance of 2D degradation. We will present our accelerated density gradient accumulation algorithm in Section III-B.

B. Algorithm Descriptions

Based on Equation (14), the algorithm of our implementation is described in Algorithm 1. For any $v \in V$, we first acquire the bins covered by the boundary faces of D_v. More concretely, the faces ∂D_v^{yz+} and ∂D_v^{yz-} are actually described by two planar rectangles $\{j_{\max}\} \times [k_{\min}, k_{\max}] \times [l_{\min}, l_{\max}]$ and $\{j_{\min}\} \times [k_{\min}, k_{\max}] \times [l_{\min}, l_{\max}]$, respectively.

Lines 3–11 demonstrate how to accumulate the potential values $\phi(\cdot)$ into the gradient $g_x := \frac{\partial U}{\partial x_i}$. Note that the vanilla approach in Equation (7) undoubtedly requires three levels of nested loops to complete the accumulation, while our proposed algorithm only requires two levels of nested loops to iterate over all the grid bins on a base surface. Algorithm 1 executes the core computation of the accumulation algorithm, which has the same form of Equation (14). A_{yz} indicates the overlap area of the face ∂D_{yz}^{-} and bin b (or the projection of bin b onto ∂D_{yz}^{-}).

Algorithm 1 DENSITY GRADIENT ACCUMULATION

Input: The coordinate (x, y, z) and size (w, h, d) of instance $v \in V$; An $N_x \times N_y \times N_z$ grid B with bin size (w_b, h_b, d_b) for any $b \in B$; The discrete electric potential map $\phi \in \mathbb{R}^{N_x \times N_y \times N_z}$.

1: Acquire boundary bin indices of D_v:

$$(j_{\min}, k_{\min}, l_{\min}) \leftarrow (\lfloor \tfrac{x}{w_b} \rfloor, \lfloor \tfrac{y}{h_b} \rfloor, \lfloor \tfrac{z}{d_b} \rfloor),$$
$$(j_{\max}, k_{\max}, l_{\max}) \leftarrow (\lfloor \tfrac{x+w}{w_b} \rfloor, \lfloor \tfrac{y+h}{h_b} \rfloor, \lfloor \tfrac{z+d}{d_b} \rfloor)$$

covered by the cuboid D_v of instance $v \in V$;
2: $(g_x, g_y, g_z) \leftarrow (0, 0, 0)$; ▷ Initialize density gradient.
3: **for** k in $k_{\min}, \cdots, k_{\max}$ **do**
4: **for** l in $l_{\min}, \cdots, l_{\max}$ **do**
5: Let $S_{yz} \leftarrow [y, y+h] \times [z, z+d]$;
6: Let $\partial b_{yz} \leftarrow [kh_b, (k+1)h_b] \times [ld_b, (l+1)d_b]$;
7: Acquire overlap $A_{yz} \leftarrow A(S_{yz} \cap \partial b_{yz})$;
8: Set two bins $b \leftarrow b(j_{\min}, k, l)$ and $b' \leftarrow b(j_{\max}, k, l)$;
9: Accumulate gradient $g_x \leftarrow g_x + A_{yz}(\phi(b') - \phi(b))$;
10: **end for**
11: **end for**
12: $g_x \leftarrow \frac{w}{\max(1, j_{\max} - j_{\min})} g_x$ ▷ Normalization in x-direction.
13: **for** j in $j_{\min}, \cdots, j_{\max}$ **do**
14: **for** l in $l_{\min}, \cdots, l_{\max}$ **do**
15: Let $S_{xz} \leftarrow [x, x+w] \times [z, z+d]$;
16: Let $\partial b_{xz} \leftarrow [jw_b, (j+1)w_b] \times [ld_b, (l+1)d_b]$;
17: Acquire overlap $A_{xz} \leftarrow A(S_{xz} \cap \partial b_{xz})$;
18: Set two bins $b \leftarrow b(j, k_{\min}, l)$ and $b' \leftarrow b(j, k_{\max}, l)$;
19: Accumulate gradient $g_y \leftarrow g_y + A_{xz}(\phi(b') - \phi(b))$;
20: **end for**
21: **end for**
22: $g_y \leftarrow \frac{h}{\max(1, k_{\max} - k_{\min})} \cdot \frac{w_b}{h_b} g_y$ ▷ Normalization in y-direction.
23: **for** j in $j_{\min}, \cdots, j_{\max}$ **do**
24: **for** k in $k_{\min}, \cdots, k_{\max}$ **do**
25: Let $S_{xy} \leftarrow [x, x+w] \times [y, y+h]$;
26: Let $\partial b_{xy} \leftarrow [jw_b, (j+1)w_b] \times [kh_b, (k+1)h_b]$;
27: Acquire overlap $A_{xy} \leftarrow A(S_{xy} \cap \partial b_{xy})$;
28: Set two bins $b \leftarrow b(j, k, l_{\min})$ and $b' \leftarrow b(j, k, l_{\max})$;
29: Accumulate gradient $g_z \leftarrow g_z + A_{xy}(\phi(b') - \phi(b))$;
30: **end for**
31: **end for**
32: $g_z \leftarrow \frac{d}{\max(1, l_{\max} - l_{\min})} \cdot \frac{w_b}{d_b} g_z$ ▷ Normalization in z-direction.
33: **return** the gradient (g_x, g_y, g_z) for the given instance $v \in V$;

The accumulation operations in the y- and z-direction are similarly demonstrated in Lines 13–21 and Lines 23–31. It worths mentioning that we have additional normalization in three directions, marked in Algorithm 1, Algorithm 1, and Algorithm 1. These normalizations are essential, as all maps must be normalized prior to being submitted to the `DCT/IDCT` solvers. For example, the $(\omega_j, \omega_k, \omega_l) = (\frac{2j\pi}{N_x}, \frac{2k\pi}{N_y}, \frac{2l\pi}{N_z})$ frequency tuples in Equation (5) are, in practice, modified to $(\omega_j, \frac{w_b}{h_b}\omega_j, \frac{w_b}{d_b}\omega_k)$ in real implementations to keep the variables of trigonometric functions in the `DCT/IDCT` families at a consistent scale. Without the normalization, the scale of the accumulation results would differ from that of the traditional approach, potentially compromising the quality of the final solution after numeric optimization.

Let integers $M_x = j_{\max} - j_{\min} + 1$, $M_y = k_{\max} - k_{\min} + 1$, and $M_z = l_{\max} - l_{\min} + 1$ represent the number of grid bins covered by D_v for $v \in V$ in the three respective directions. The vanilla approach requires $O(M_x M_y M_z)$ primitive operations to calculate the gradient $\frac{\partial U}{\partial x_i}$, whereas Algorithm 1 necessitates only $O(M_x M_y + M_y M_z + M_z M_x)$. This reduction in complexity offers significant advantages, particularly for large macros that encompass numerous grid bins.

C. Theoretical Analysis via Functionals

Note that we use the symbol $\propto$ in Equation (6), as it is empirical based on human knowledge. It is non-trivial to acquire the equality as node v_i serves as a component of the electric field. In this subsection, we will theoretically show that the partial derivative $\frac{\partial U}{\partial x_i}$ of the energy model $U = \frac{1}{2} \iiint_\Omega \rho\phi \, \mathrm{d}\Omega$ is consistent with the result of Corollary 1. Note that the factor $\frac{1}{2}$ has been discarded.

Consider the density map $\rho : \mathbb{R}^3 \to \mathbb{R}$ in Equation (4). It is the sum of n indicator functions. Given any point $(x, y, z) \in \Omega$ within the placement region, $\rho(x, y, z)$ measures how many nodes cover this point. ePlace-3D [6] computes the potential mapping $\phi : \mathbb{R}^3 \to \mathbb{R}$ by solving the 3D Poisson's equation in Equation (2) under Neumann boundary condition,

The numerical solution to Equation (2) has been adequately discussed in previous works [6], [11], [12]. Regardless of the exact form of the potential function ϕ, we can always consider it as an functional operator of ρ, and thus denote it by $\phi(\cdot) = \phi[\rho](\cdot)$. Hence, the electric potential energy $U = U[\rho]$ is a functional of ρ. The partial derivative $\frac{\partial U}{\partial x_i}$ measures how the energy U will change when moving v_i along the x-direction which results in a perturbation on ρ. We will prove the following **main result** that supports our theory and algorithm in this subsection.

Theorem 2. *Assume that we have n cuboids $D_{v_1}, \cdots, D_{v_n}$ within the cuboid region $\Omega \subset \mathbb{R}^3$. Each of them has a uniform charge distribution with density 1. The electric potential distribution ϕ of this system is determined by Poisson's equation in Equation (2). We have*

$$\frac{\partial U}{\partial x_i} = \iint_{\partial D_{v_i}^{yz-}} (\phi(x + w_i, y, z) - \phi(x, y, z)) \, \mathrm{d}y\mathrm{d}z, \quad (15)$$

where $U = \frac{1}{2} \iiint_\Omega \rho\phi \, \mathrm{d}\Omega$ is the system potential energy.

Proof. Slightly moving v_i by Δx along the x-direction will introduce a change $\Delta\rho$ on the density map ρ. Assume that the cuboid D_{v_i} becomes $D_{v_i}^{\Delta x}$ after the movement, the change of the density function is represented by

$$\Delta\rho = \chi_{D_{v_i}^{\Delta x}} - \chi_{D_{v_i}}. \quad (16)$$

The solution of electric potential $\phi[\rho]$ to Equation (2) is

$$\phi[\rho](\mathbf{r}) = \iiint_\Omega \rho(\mathbf{r}') G(\mathbf{r}, \mathbf{r}') \, \mathrm{d}^3\mathbf{r}', \quad (17)$$

where $G(\mathbf{r}, \mathbf{r}')$ is the Green's function for Equation (2). We do not have a closed-form representation of $G(\mathbf{r}, \mathbf{r}')$ under the Neumann boundary condition. However, it is quite useful in analysis. According to Equation (17), the electric potential energy can be represented as a functional of ρ,

$$U[\rho] = \frac{1}{2} \iiint_\Omega \iiint_\Omega \rho(\mathbf{r}) \rho(\mathbf{r}') G(\mathbf{r}, \mathbf{r}') \, \mathrm{d}^3\mathbf{r}' \mathrm{d}^3\mathbf{r}, \quad (18)$$

From the definition of derivatives, we have

$$\frac{\partial U}{\partial x_i} = \lim_{\Delta x \to 0} \frac{U[\rho + \Delta\rho] - U[\rho]}{\Delta x}. \tag{19}$$

The expression for which the limit is taken is

$$\begin{aligned}
&\frac{U[\rho + \Delta\rho] - U[\rho]}{\Delta x} \\
&= \frac{1}{2\Delta x} \iiint_\Omega \iiint_\Omega \big[(\rho(\mathbf{r}) + \Delta\rho(\mathbf{r}))(\rho(\mathbf{r}') + \Delta\rho(\mathbf{r}')) \\
&\quad - \rho(\mathbf{r})\rho(\mathbf{r}')\big] G(\mathbf{r}, \mathbf{r}') \, \mathrm{d}^3\mathbf{r}' \mathrm{d}^3\mathbf{r} \\
&= \frac{1}{2\Delta x} \iiint_\Omega \iiint_\Omega \big[\rho(\mathbf{r})\Delta\rho(\mathbf{r}') + \rho(\mathbf{r}')\Delta\rho(\mathbf{r})\big] G(\mathbf{r}, \mathbf{r}') \, \mathrm{d}^3\mathbf{r}' \mathrm{d}^3\mathbf{r} \\
&\quad + \frac{1}{2\Delta x} \iiint_\Omega \iiint_\Omega \Delta\rho(\mathbf{r})\Delta\rho(\mathbf{r}') G(\mathbf{r}, \mathbf{r}') \, \mathrm{d}^3\mathbf{r}' \mathrm{d}^3\mathbf{r} \\
&\overset{\text{(a)}}{=} \frac{1}{\Delta x} \iiint_\Omega \iiint_\Omega \rho(\mathbf{r}')\Delta\rho(\mathbf{r}) G(\mathbf{r}, \mathbf{r}') \, \mathrm{d}^3\mathbf{r}' \mathrm{d}^3\mathbf{r} \\
&\quad + \frac{1}{2\Delta x} \iiint_\Omega \iiint_\Omega \Delta\rho(\mathbf{r})\Delta\rho(\mathbf{r}') G(\mathbf{r}, \mathbf{r}') \, \mathrm{d}^3\mathbf{r}' \mathrm{d}^3\mathbf{r}.
\end{aligned} \tag{20}$$

The equation marked (a) holds because of the symmetry of $\mathbf{r}$ and $\mathbf{r}'$. The righthand side of (a) consists of two integrals I_1 and I_2. Consider the first integral I_1, we have

$$\begin{aligned}
I_1 &= \frac{1}{\Delta x} \iiint_\Omega \iiint_\Omega \rho(\mathbf{r}')\Delta\rho(\mathbf{r}) G(\mathbf{r}, \mathbf{r}') \, \mathrm{d}^3\mathbf{r}' \mathrm{d}^3\mathbf{r} \\
&= \iiint_\Omega \frac{\Delta\rho(\mathbf{r})}{\Delta x} \phi(\mathbf{r}) \, \mathrm{d}^3\mathbf{r} = \iiint_\Omega \frac{\chi_{D_{v_i}^{\Delta x}}(\mathbf{r}) - \chi_{D_{v_i}}(\mathbf{r})}{\Delta x} \phi(\mathbf{r}) \, \mathrm{d}^3\mathbf{r}
\end{aligned} \tag{21}$$

By Lebesgue's dominated convergence theorem, it is not difficult to conclude that

$$\begin{aligned}
\lim_{\Delta x \to 0} I_1 &= \iiint_\Omega \phi(\mathbf{r}) \cdot \lim_{\Delta x \to 0} \frac{\chi_{D_{v_i}^{\Delta x}}(\mathbf{r}) - \chi_{D_{v_i}}(\mathbf{r})}{\Delta x} \mathrm{d}^3\mathbf{r} \\
&= \iiint_\Omega \phi(\mathbf{r}) \cdot \delta_{\partial D_{v_i}}(\mathbf{r}) \hat{n}_x \mathrm{d}^3\mathbf{r} = \oiint_{\partial D_{v_i}} \hat{n}_x \phi \, \mathrm{d}S,
\end{aligned} \tag{22}$$

where $\delta_{\partial D_{v_i}}(\mathbf{r})$ is the surface Dirac delta function. Similarly, as for the second integral I_2, we have

$$\lim_{\Delta x \to 0} I_2 = \frac{1}{2} \iiint_\Omega \iiint_\Omega \frac{\Delta\rho(\mathbf{r})}{\Delta x} \Delta\rho(\mathbf{r}') G(\mathbf{r}, \mathbf{r}') \, \mathrm{d}^3\mathbf{r}' \mathrm{d}^3\mathbf{r} = 0, \tag{23}$$

as $\frac{\Delta\rho(\mathbf{r})}{\Delta x}$ tends to a surface Dirac delta function and $\Delta\rho(\mathbf{r}')$ tends to zero. Combining Equation (22) and Equation (23) into Equation (20) and Equation (19), we have

$$\begin{aligned}
\frac{\partial U}{\partial x_i} &= \lim_{\Delta x \to 0} \frac{U[\rho + \Delta\rho] - U[\rho]}{\Delta x} = \oiint_{\partial D_{v_i}} \hat{n}_x \phi \, \mathrm{d}S \\
&\overset{\text{(b)}}{=} \iint_{\partial D_{v_i}^{yz-}} (\phi(x + w_i, y, z) - \phi(x, y, z)) \, \mathrm{d}y\mathrm{d}z.
\end{aligned} \tag{23}$$

The equation marked (b) is directly from Equation (13). Therefore, we have completed the proof. □

Note that there exists a factor $\frac{1}{2}$ in the definition of the potential energy. However, it is discarded in Equation (15) due to the effect of the dependence of the node v_i and the induced electrostatics system.

IV. Experimental Results

We implement our proposed density gradient accumulation algorithm in `C++/CUDA` and conduct experiments on the recent ICCAD 2023 contest benchmarks [19] of 3D mixed-size placement for face-to-face (F2F) bonded 3D ICs. Since 3D ICs require additional *hybrid bonding terminals* (HBT) for die-to-die connections, every net $e \in E$ connecting instances on different dies needs one HBT for communications. Following the evaluation scheme of the contest [19], the **raw score** is defined as raw score = HPWL + $\beta \cdot$ #HBTs. Here, the weight factor specified by the contest organizer is β set to 10 by default for all designs.

All experiments were performed on a Linux workstation with two 56-core Intel(R) Xeon(R) Platinum 8480C CPUs (Max 3.8 GHz), an NVIDIA H800 GPU with 80GB VRAM. To better accommodate the scenarios with less CPU resources, all CPU programs are executed with maximum `num_threads` being 8. We compare our implementations with the state-of-the-art (SOTA) mixed-size 3D analytical placers [13] and [14]. The results of these baselines are directly cited from their respective publications. It should be noted that [14] offers both CPU and GPU implementations. Consequently, we compare our CPU version against both [13] and [14], while our GPU version is compared exclusively against [14].

A. Comparison with SOTA 3D Analytical Placers

The benchmark statistics and raw score comparison results are shown in TABLE I. We implemented the bistratal wirelength [12] as our 3D wirelength model $W(\mathbf{v})$ and the 3D prefix sum described in [14] in the forward density accumulation. The raw score indicates the overall quality incorporating both 3D die-to-die HPWL and the number of hybrid bonding terminals. The runtime demonstrates the end-to-end runtime. To ensure reproducibility, all results were evaluated in *deterministic* mode.

As illustrated in TABLE I. We achieved 4.029× on average and up to 4.75× end-to-end runtime speed on GPU compared to [14] without quality loss. On the largest benchmarks `case4` and `case4h`, our GPU implementation yielded comparable wirelength results while requiring approximately one-third of the runtime. When running on CPU, we can also achieve a comparable wirelength result with 3.18× and 2.69× end-to-end runtime speedup on average compared to [14] and [13], respectively. Notably, there also exists a 3% wirelength improvement on CPU compared to [13].

It worth mentioning that since the hardware and implementation differ from [14], the end-to-end runtime improvement might be affected by multiple factors. For example, [14] applies a subsequent 2D placement inheriting the partitioning from 3D global placement for medium and small cases except `case4` and `case4h`, which explains its high quality on tedious cases like `case3`. However, we only applies the subsequent 2D placement for `case2`. Despite that, we can still

TABLE I The raw score results compared to the SOTA placers on the ICCAD 2023 contest benchmarks [19] of 3D mixed-size placement. **RT** (s) demonstrates the *end-to-end* runtime. The CPU results are evaluated with 8 threads. $|V|$ and $|V_M|$ represent the total number of instances and movable macros, respectively.

Bench.	Statistics		[13]		[14]-CPU		Ours-CPU		[14]-GPU		Ours-GPU	
	$\|V\|$	$\|V_M\|$	Score	RT	Score	RT	Score	RT	Score	RT	Score	RT
case2	13901	6	16026889	177	15540090	76	15639790	26	15635352	38	15443965	15
case2h1	13901	6	17695111	74	16719713	80	16706960	20	16569703	35	16775356	9
case2h2	13901	6	17701682	77	16759058	80	16773536	20	16820960	36	16721757	9
case3	124231	34	98737450	519	97388944	236	100684926	78	98206238	92	99997072	21
case3h	124231	34	115694517	196	109518959	239	109096510	73	108166770	86	107974746	20
case4	740211	32	1045958081	3149	1041523590	3070	1040956218	964	1037676163	335	1038139180	77
case4h	740211	32	637119983	2266	635991850	2640	631972730	980	635259476	361	630601092	76
AVERAGE			1.028	4.182	0.996	3.300	1.000	1.000	1.000	4.029	1.000	1.000

TABLE II The results of ablation study on the same machine with different density backward accumulation algorithms.

Bench.		Vanilla		Ours	
		Score	RT	Score	RT
CPU	case2	15657924	39	15639790	26
	case2h1	16723968	29	16706960	20
	case2h2	16748488	29	16773526	20
	case3	101738290	96	100684926	78
	case3h	109580966	86	109096510	73
	case4	1037251354	1315	1040956218	964
	case4h	632423856	1310	631972730	980
AVERAGE		1.002	1.359	1.000	1.000
GPU	case2	15587229	21	15443965	15
	case2h1	16850092	12	16775356	9
	case2h2	16801938	12	16721757	9
	case3	101196054	34	99997072	21
	case3h	107621590	33	107974746	20
	case4	1035731150	102	1038139180	77
	case4h	632611980	103	630601092	76
AVERAGE		1.004	1.431	1.000	1.000

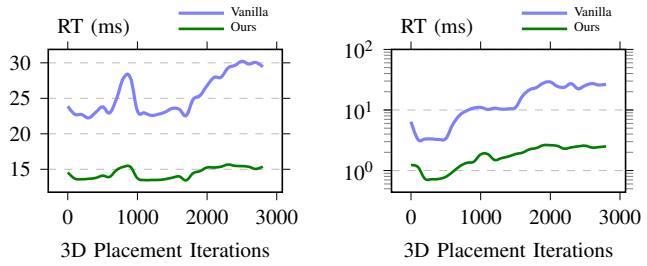
achieve the best score on case2, case2h2, and case3h, demonstrating that our acceleration is quality-preserving.

To fairly demonstrate the effectiveness of our algorithm, an ablation study with a vanilla density backward accumulation on the same machine will be presented in Section IV-B.

B. Comparison with Vanilla Accumulation

We compare the results with the vanilla density backward accumulation in TABLE II. The vanilla algorithm to calculate a gradient with respect to a macro simply sums all values of the 3D gradient maps covered by its box in a brute-force manner, as shown in Equation (14). The vanilla approach typically parallelizes the computation in the node-level, which means, a large macro will introduce significant workload unbalance and thus decrease the program efficiency. TABLE II demonstrates that vanilla density backward accumulation introduces 35.9% and 43.1% runtime overhead on CPU and GPU, respectively, compared to our accelerated algorithm.

We plot the average runtime required by the density gradient accumulation for a single iteration against the total placement iterations in Fig. 3. To ensure the quality of the final solution for case4h, the 3D analytical placer requires nearly 3,000 iterations to execute the global optimization (including macro rotation and re-initialization), therefore necessitate the high-performance algorithms to accumulate the density accumulation. Since Nesterov's method [22] has been integrated in the optimizer by default [5], [7], [12], [14], a single iteration may include multiple steps of line search, *i.e.* a single iteration may invoke multiple times of forward and backward operations. This is the reason why we plot the *average* density backward runtime within a single iteration Fig. 3.



(a) Backward runtime on CPU (b) Backward runtime on GPU

Fig. 3 The single-iteration time required to accomplish the backward computation with different schemes for case4h.

The runtime comparison on GPU is illustrated in Fig. 3(b). During the initial iterations, the macros and standard cells predominantly accumulate at the center of the placement region as a result of random initialization. This central concentration generates a significant runtime demand due to the resulting imbalance in the potential map ϕ. As illustrated in Fig. 3(b), we can achieve significant runtime speed up compared to the vanilla approach. The detailed breakdown is listed in TABLE III.

C. Comparison with Other Accumulation Techniques

In TABLE III, we compare our implementations with other density gradient accumulation techniques: vanilla, vin-level parallelism [14], and 3D prefix sum [14] on both CPU and GPU. Note that the bin-level parallelism yields to an extremely slow scheme on CPU with 8 threads so we did not consider the corresponding results in TABLE III.

The detailed density backward runtime results for all cases are listed in TABLE III. Note that the vanilla approach is significantly slower on GPU for small designs like case2 than 740K cases case4/case4h. This is due to the large macro-area property of the ICCAD 2023 benchmarks [19]. For example, the 3 macros in case2 cover almost 88% of

TABLE III The **density** operator forward and backward runtime (ms) per forward-backward step of different algorithms in the 3D global placement. **PS** stands for *3D prefix sum* [14] and **BP** represents the *bin-level parallelism* of the vanilla approach [14].

Bench.	CPU Forward			CPU Backward			GPU Forward				GPU Backward			
	Vanilla	PS	Ours	Vanilla	PS	Ours	Vanilla	BP	PS	Ours	Vanilla	BP	PS	Ours
case2	10.128	9.926	6.830	2.616	4.395	0.610	0.558	0.556	0.563	0.369	15.205	3.470	0.450	1.235
case2h1	11.100	11.079	7.715	2.278	5.273	0.791	0.566	0.569	0.559	0.370	9.050	2.777	0.447	0.856
case2h2	10.933	10.884	7.712	2.331	5.301	0.818	0.551	0.562	0.568	0.367	8.813	2.836	0.447	0.843
case3	21.228	21.038	17.886	3.155	11.494	1.995	1.699	1.706	1.717	1.129	12.601	5.378	1.252	1.451
case3h	23.287	23.207	20.236	3.420	12.194	2.169	1.710	1.716	1.731	1.133	10.717	4.637	1.266	1.252
case4	197.463	212.773	128.299	25.232	89.008	14.195	3.701	3.705	3.741	2.502	14.820	6.531	2.019	1.752
case4h	203.654	205.360	136.163	25.474	89.212	14.467	3.707	3.702	3.736	2.507	14.686	6.920	2.018	1.744
AVERAGE	1.387	1.399	1.000	2.388	6.310	1.000	1.502	1.508	1.517	1.000	9.638	3.503	0.800	1.000

the total area, leading to an extreme workload imbalance for node parallelism on GPU.

According to TABLE III, we can even achieve around 40% and 50% runtime improvement on density *forward* computation on CPU and GPU, respectively, by merely trying to optimize the backward computation. This phenomenon reflects the observation of Corollary 1 that there is no need to compute the field map $\mathbf{E}$ anymore. The forward computation of ePlace-3D [6] typically includes:

1) The computation of density map $\rho(\mathbf{r})$;
2) A `DCT_3D` operation to compute coefficients a_{jkl};
3) An `IDCT_3D` operation to compute potential map $\phi(\mathbf{r})$;
4) Subsequent 3D combinations of `IDCT` and `IDXST` operations to compute the three dimensions of field maps $\mathbf{E}(\mathbf{r}) = (E_x(\mathbf{r}), E_y(\mathbf{r}), E_z(\mathbf{r}))$.

During backward pass, gradients are obtained by 3D accumulation of $\mathbf{E}(\mathbf{r})$, while step 4 is omitted. The density gradient calculation operates directly on $\phi(\mathbf{r})$, resulting in the forward pass optimization shown in TABLE III.

The bin-level parallelism of the vanilla approach to calculate macro gradients in 3D placement is introduced in [14], where the brute-force computation of Equation (14) is parallelized in bin level. This approach is only applicable to large macros as they may cover too many bins and cause workload imbalance for node-level parallelism. Furthermore, this method is highly dependent on massively parallel hardware architectures (e.g., GPUs) to enable parallelized processing of potentially tens of millions of atomic computations. Consequently, we deliberately eschew this approach on CPU. As evidenced by TABLE III, this method achieves significant reduction in density backward runtime on a single H800 GPU.

The 3D prefix sum [1], [14] utilizes high-performance CUDA parallelism of the inclusive scan operation on field map $\mathbf{E}$ and achieved impressive speed up against the vanilla approach. The 3D prefix sum approach requires performing a 3D inclusive scan operation on the field map $\mathbf{E}$, which is implemented as three consecutive in-place cumulative sums (`cumsum`) on the field map tensor. This enables density backward computation in constant time once `cumsum` is done. Consequently, the 3D prefix sum method can highly leverage the parallelism of GPUs. Actually, the computational bottleneck of this approach lies in the `cumsum` operations, with even more runtime overhead of type casting between floating-point and fixed-point numbers in deterministic mode. Thus, while this solution performs exceptionally well on small cases, its scalability is limited. As shown in TABLE III, due to its superior performance on small cases, its average runtime is only 80% of our proposed algorithm's. However, for 740K designs (`case4`/`case4h`), it requires longer computation time. Moreover, since it cannot bypass the field map computation, the total combined time of the forward and backward operations of the density operator for `case4`/`case4h` is approximately 1.35× that of our proposed solution.

We observe distinct behaviors on CPU. The 3D inclusive scan operations on three super large 3D electric field maps $\mathbf{E} = (E_x, E_y, E_z)$ encountered a severe runtime degradation due to the necessity of typecasting from floating-point to fixed-point numbers after scaling. Consequently, the 3D prefix sum is inferior to the vanilla approach on CPU. Nonetheless, our proposed algorithm demonstrates a superior performance, achieving up to a 4.29× runtime speedup and averaging a 2.39× improvement compared to the vanilla accumulation scheme. As shown in TABLE III, a substantial proportion of the computational time reduction on CPU architectures is observed in the forward computation. For instance, in `case4h`, the elimination of field map computations reduces the forward pass duration from about 204 ms to 136 ms, yielding a saving of approximately 80 ms per complete forward-backward operation of the density operator. Given that a single iteration may include multiple line search operations, these incremental savings compound to produce an aggregate reduction in computational time of around 300 seconds.

V. Conclusion

This paper proposes an algorithm inspired by the divergence theorem to reduce the time complexity of density gradient accumulation in 3D analytical placement, thereby achieving significant runtime speedup while preserving the solution quality. Additionally, we provide a theoretical analysis to validate the correctness and rationale of the proposed algorithm. Experimental results demonstrate that our approach significantly accelerates the density forward and backward operation on both CPU and GPU, attesting to the efficiency and reliability of our algorithm.

Acknowledgements

The project is supported in part by Research Grants Council of Hong Kong SAR (No. RFS2425-4S02 and No. CUHK14211824).

REFERENCES

[1] Z. Guo, J. Mai, and Y. Lin, "Ultrafast CPU/GPU kernels for density accumulation in placement," in *ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2021, pp. 1123–1128.

[2] 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 (TCAD)*, vol. 27, no. 7, pp. 1228–1240, 2008.

[3] M.-K. Hsu, Y.-F. Chen, C.-C. Huang, S. Chou, T.-H. Lin, T.-C. Chen, and Y.-W. Chang, "NTUplace4h: A novel routability-driven placement algorithm for hierarchical mixed-size circuit designs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 33, no. 12, pp. 1914–1927, 2014.

[4] T. Lin, C. Chu, J. R. Shinnerl, I. Bustany, and I. Nedelchev, "POLAR: A high performance mixed-size wirelengh-driven placer with density constraints," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 34, no. 3, pp. 447–459, 2015.

[5] 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 Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 20, no. 2, pp. 1–34, 2015.

[6] J. Lu, H. Zhuang, I. Kang, P. Chen, and C.-K. Cheng, "ePlace-3D: Electrostatics based placement for 3D-ICs," in *ACM International Symposium on Physical Design (ISPD)*, 2016, pp. 11–18.

[7] 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 *ACM/IEEE Design Automation Conference (DAC)*, 2019, pp. 1–6.

[8] J. Lu, P. Chen, C.-C. Chang, L. Sha, D. J.-H. Huang, C.-C. Teng, and C.-K. Cheng, "FFTPL: An analytic placement algorithm using fast fourier transform for density equalization," in *IEEE International Conference on ASIC (ASICON)*. IEEE, 2013, pp. 1–4.

[9] J. Lu, H. Zhuang, P. Chen, H. Chang, C.-C. Chang, Y.-C. Wong, L. Sha, D. Huang, Y. Luo, C.-C. Teng *et al.*, "ePlace-MS: Electrostatics-based placement for mixed-size circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 34, no. 5, pp. 685–698, 2015.

[10] 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 (TCAD)*, vol. 38, no. 9, pp. 1717–1730, 2018.

[11] Y.-J. Chen, Y.-S. Chen, W.-C. Tseng, C.-Y. Chiang, Y.-H. Lo, and Y.-W. Chang, "Late breaking results: Analytical placement for 3D ICs with multiple manufacturing technologies," in *ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2023, pp. 1–2.

[12] P. Liao, Y. Zhao, D. Guo, Y. Lin, and B. Yu, "Analytical die-to-die 3D placement with bistratal wirelength model and GPU acceleration," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 43, no. 6, pp. 1624–1637, 2023.

[13] Y.-J. Chen, C.-H. Hsieh, P.-H. Su, S.-H. Chen, and Y.-W. Chang, "Mixed-size 3D analytical placement with heterogeneous technology nodes," in *ACM/IEEE Design Automation Conference (DAC)*, 2024, pp. 1–6.

[14] Y. Zhao, P. Liao, S. Liu, J. Jiang, Y. Lin, and B. Yu, "Analytical heterogeneous die-to-die 3D placement with macros," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 2024.

[15] C.-X. Lin and M. D. F. Wong, "Accelerate analytical placement with GPU: A generic approach," in *IEEE/ACM Proceedings Design, Automation and Test in Europe (DATE)*. IEEE, 2018, pp. 1345–1350.

[16] F. Gessler, P. Brisk, and M. Stojilovič, "A shared-memory parallel implementation of the RePlAce global cell placer," in *International Conference on VLSI Design*. IEEE, 2020, pp. 78–83.

[17] X. Li, K. Peng, F. Huang, and W. Zhu, "PeF: Poisson's equation-based large-scale fixed-outline floorplanning," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 42, no. 6, pp. 2002–2015, 2022.

[18] K. Peng and W. Zhu, "Pplace-MS: Methodologically faster Poisson's equation based mixed-size global placement," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 2023.

[19] K.-S. Hu, H.-Y. Chi, I.-J. Lin, Y.-H. Wu, W.-H. Chen, and Y.-T. Hsieh, "2023 iccad cad contest problem b: 3d placement with macros," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 2023, pp. 1–6.

[20] C. F. Gauss, "Theoria attractionis corporum sphaeroidicorum ellipticorum homogeneorum methodo nova tractata," *Commentationes Societatis Regiae Scientiarium Gottingensis Recentiores*, vol. 2, pp. 355–378, 1813.

[21] M. Ostragradsky, "Première note sur la théorie de la chaleur," *Mémoires de l'Académie impériale des sciences de St. Pétersbourg*, vol. 1, pp. 129–133, 1831, presented: November 5, 1828.

[22] Y. Nesterov, "A method of solving a convex programming problem with convergencerate $O(\frac{1}{k^2})$," *Doklady Akademii Nauk SSSR*, vol. 269, no. 3, pp. 543–547, 1983.