CoPlace: Coherent Placement Engine with Layout-aware Partitioning for 3D ICs

Bangqi Fu
CSE Department, CUHK
bqfu21@cse.cuhk.edu.hk

Lixin Liu
CSE Department, CUHK
lxliu@cse.cuhk.edu.hk

Yang Sun
CSE Department, CUHK
ysun22@cse.cuhk.edu.hk

Wing-Ho Lau
CSE Department, CUHK
whlau22@cse.cuhk.edu.hk

Martin D.F. Wong
CSE Department, CUHK
mdfwong@cuhk.edu.hk

Evangeline F.Y. Young
CSE Department, CUHK
fyyoung@cse.cuhk.edu.hk

***Abstract*—The emerging technologies of 3D integrated circuits (3DICs) unveil a new avenue for expanding the design space into the 3D domain and present the opportunity to overcome the bottleneck of Moore's Law for the traditional 2DICs. Among various technologies, the face-to-face bonding structure provides high integration density and reliable performance. Most commercial EDA tools, however, do not support 3DIC and cannot give a convincing solution. To exploit the benefits of stacking multiple tiers vertically, placement algorithms for 3DIC are imperatively in need. In this paper, we proposed a design flow that optimizes partitioning and placement quality for 3DICs in a unified way. Experimental results on the ICCAD2022 contest benchmark show that our work outperforms the first-place team by 3.35% in quality with less runtime and terminals used.**

***Index Terms*—Physical design, 3D IC, Global placement, Partitioning**

I. Introduction

3D integration circuit (3DIC) technology has shown exciting potential for improving circuit quality like power, performance, and area compared to 2D circuits and is a promising technology to pursue while the circuit scale rapidly expands. Traditional EDA tools, however, are developed for 2D circuits and cannot provide reliable design solutions for 3D circuits, thus bringing problems on interconnect, thermal, timing issues, etc. for the design of 3DICs.

Various technologies have been proposed in the past decades to overcome the interconnect issue in 3DIC. The through-silicon via (TSV) technology adopts TSVs for interconnection between multiple dies. TSVs are large vias that occupy silicon areas and bring significant quality overhead. The Monolithic 3D (M3D) technology fabricates dies sequentially with monolithic inter-tier vias (MIV) and achieves much higher integration density[1].

The Die-to-Die (D2D) bonded 3DIC stacks two dies vertically with hybrid bonding terminals and offers high flexibility compared to the previous two technologies [2]. The hybrid bonding terminals are small patches on the topmost padding layers of each die and will not occupy placement resources. By doing so, a large die is split into 2 small dies and stacked vertically in order to have a smaller area and better timing. The unique advantage of D2D IC is that each of the 2 dies can be fabricated independently with different technologies, which would result in much higher flexibility in circuit optimization and cost-effectiveness. This feature means that cell configurations like size and layout are different on each die, making the design of D2D 3DICs even more challenging.

To better exploit the benefits of 3DICs, many previous works have been proposed to explore the co-optimization of netlist partitioning and cell placement. The work [3] transforms a 2D design into a 3D design by stacking and folding in order to obtain balanced areas and an even via density distribution among dies. The work [4] derives

The work described in this paper was partially supported by a grant from the Research Grants Council of the Hong Kong Special Administrative Region, China (Project No. CUHK14209320).

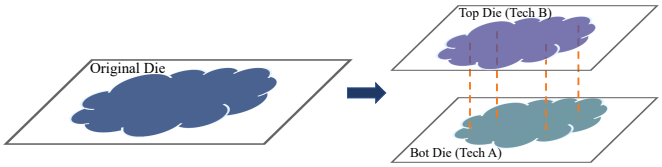

Fig. 1: Illustration of a netlist partitioning.

3D placement from a 2D placement by applying a bin-based FM-partition to obtain an area-balanced partitioning for two dies. The paper [5] proposed a design flow using a 2D commercial tool in which they expand the die scale to place all the standard cells on a single die and then contract the layout to 3D. The work [6] models the 3D electrostatic field and performs placement in the 3D space to optimize wirelength and the z-direction of the 3D placement is then discretized to obtain the layer assignment solution.

Most of the previous works isolate the partitioning and placement stages to approximate the final quality of a 3DIC which can lead to significant quality loss. A good circuit partitioning alone lacks the layout information and cannot guarantee good placement quality, while a good placement result may not be well preserved after partitioning. A placer that is strongly coupled with a partitioner is imperatively needed. In this paper, we propose a coherent placement engine that integrates the placement and partitioning stages seamlessly. Our main contribution in this work can be concluded as follows:

- We propose a coherent framework that leverages a single placement engine for the placement and partitioning co-optimization. The framework consists of 3 stages built with a unified placement engine. A compact 2D placement that serves as an initial solution; a 3D placement on the expanded z-dimension that pushes cells to each die; a 2.5D multi-layer placement that places the cell and terminal layers simultaneously.
- We propose a wirelength-driven FM partitioner that effectively improves the post-partition HPWL with a negligible number of terminal overhead.
- A row-based legalization and detailed placement for cells and matching-based legalization for terminals are proposed.
- We achieve 3.35% improvement in terms of the overall score with competitive runtime and number of terminals compared to the top 3 teams of the ICCAD 2022 Contest.

II. Preliminaries

A. Overview of 2D Analytical Placement

A typical 2D analytical placer minimizes the wirelength of a given circuit $G = (V, E)$ with a series of constraints, where V is the set of cells and E is the set of nets. Let $v = \{(x_1, y_1), ..., (x_N, y_N)\} \in$

$\mathbb{R}^{N\times 2}$ denote the 2D positions of the cells, and N is the number of cells. The objective of the global placement is formulated as:

$$\begin{aligned} \min_{v} \quad & HPWL(v) = \min_{v} \sum_{e\in E} HPWL_e(v) \\ \text{s.t.} \quad & D(x,y) \le D_t \end{aligned} \tag{1}$$

where $D(x,y)$ is the density of a local location in the placement region R, and and D_t is the given target density constraint.

The cell overlapping constraint would be hard to integrate into the objective, thus the analytical global placement models relax the overlapping constraints into the objective function that can be solved by gradient solvers. We formulate the cell density constraints as a penalty term and replace the HPWL with a smooth and differentiable approximation. The unconstrained objective function is defined as:

$$\min_{v} \sum_{e\in E} WL_e(v) + \lambda D(v) \tag{2}$$

where the wirelength $WL_e(v) = WL_e(x)+WL_e(y)$ is the weighted average (WA) wirelength [7]:

$$WL_e(x) = \frac{\sum_{i\in e} x_i e^{x_i/\gamma}}{\sum_{i\in e} e^{x_i/\gamma}} - \frac{\sum_{i\in e} x_i e^{-x_i/\gamma}}{\sum_{i\in e} e^{-x_i/\gamma}} \tag{3a}$$

and $WL_e(y)$ correspondingly. The coefficient γ determines the precision of the WA wirelength model. And the density weight λ controls the spreading of cells. A global placement flow usually starts with a large γ and a small λ and gradually updates them to optimize the wirelength objective and remove overlaps.

The cell density function follows the ePlace[8], which models the layout of placement as an electrostatic system and cells as positive charges and the fields as computed using Poisson's equation:

$$\begin{cases} \bigtriangledown \cdot \bigtriangledown \psi(x,y) = -\rho(x,y), \\ \hat{\mathbf{n}} \cdot \bigtriangledown \psi(x,y) = 0, (x,y) \in \partial R, \\ \iint_R \rho(x,y) = \iint_R \psi(x,y) = 0, \end{cases} \tag{4}$$

where $\rho(x,y)$ is the electron density map, $\psi(x,y)$ is the potential distribution and ∂R is the boundary of placement region. The numerical solution of the equation is obtained by discrete cosine transformation[9].

B. D2D Bonded 3DIC

The emerging technologies of semiconductor fabrication make it possible and practical to expand circuit integration into the 3D domain. Among the various 3DIC technologies, the D2D bonded 3DIC provides high integration density and flexibility since the two dies can be fabricated in parallel and stacked vertically afterward with hybrid bonding terminals as interconnection. The hybrid bonding terminals are small pitches placed at the topmost padding layer of a die and thus will not occupy extra placement resources. The terminals are required to follow the spacing constraint of padding layers. We denote the size of a terminal by (w^b, h^b), and the spacing required between two terminals by s^b.

The two dies can be fabricated with different technologies to achieve cost reduction. With different technologies, characteristics like cell height, cell size, and cell pin locations would be different. In other words, a logical library is mapped to two different definitions corresponding to the two dies. Suppose we have two dies, $\mathcal{C} = \{0, 1\}$, let $s^c = \{(w_1^c, h_1^c), ..., (w_N^c, h_N^c)\} \in \mathbb{R}^{N\times 2}, c \in \mathcal{C}$ be the cell sizes on die c, And $s^0 > s^1$ without loss of generation.

Our objective is to partition the given circuit $G = (V, E)$ into two sub-circuits $G^c = (V^c, E^c)$ and place cells on each die to minimize the total cross-die HPWL. For a net that is cut, we need to create a bonding terminal on the bonding terminal layer as the interconnection between the cells on the two dies. The bonding terminals are represented as stand-alone vertices V^b. A terminal is regarded as a pin of the corresponding net. Let $v = \{v^0, v^1\} = \{(x_1, y_1), ..., (x_N, y_N)\} \in \mathbb{R}^{N\times 2}$ denote the 2D projected positions of the cells on the two dies and v^b be the positions of the bonding terminals, our goal is to minimize the cross-die HPWL defined as follows:

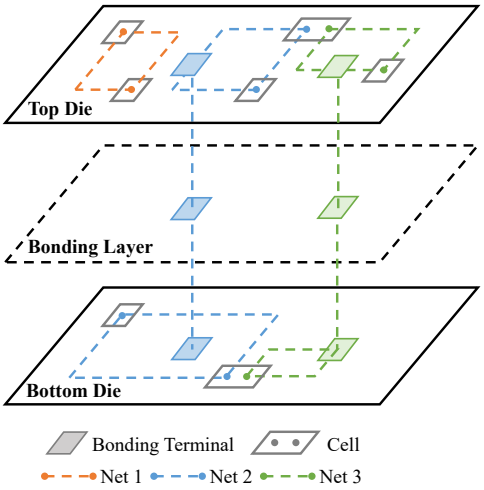

Fig. 2: An example of placement of the standard cells on 2 dies and hybrid bonding terminals and the interconnection.

$$\begin{aligned} \min_{v,v^b} \sum_{c\in\mathcal{C}} HPWL(v^c \cup v^b) &= \min_{v,v^b} \sum_{c\in\mathcal{C}} \sum_{e\in E^c} HPWL_e(v^c \cup v^b) \\ \text{s.t.} \quad U^c &\le U_M^c, \forall c \in \mathcal{C} \end{aligned} \tag{5}$$

where the U^c is the actual utilization of die c and U_M^c is the corresponding maximum utilization constraint. An example of the placement of 2 dies and bonding terminals is shown in Fig. 2.

III. Proposed Framework

The overall flow of our proposed framework is illustrated in Fig. 3. The framework mainly consists of three stages using a unified placement engine coherently: (1) compact 2D global placement. (2) 3D placement on the expanded z-dimension. (3) 2.5D multi-layer placement.

The first stage of compact 2D global placement provides an initial solution for the later partitioning and 2.5D placement. All the cells are placed on a single layer in a compact way with double of the target density, which gives an essential view of the final placement solution. The placement domain is then expanded into 3D and cells are spread out in the z-direction. The position on z is discretized into binary values as the partitioning result. The placement field is then split into 3 layers corresponding to the partitioned cells and the generated bonding terminals. This multi-electrostatic-based 2.5D placement engine optimizes precisely the cross-die HPWL and the density on each layer simultaneously. Besides, an efficient wirelength-driven FM partitioning algorithm is proposed to reduce the HPWL overhead of the initial partitioning solution in the 3D placement stage. At last, the row-based legalization and detailed placement for cells and the matching-based legalization for terminals are applied to refine the solution. The proposed algorithms will be discussed in detail in the following sections.

A. Compact 2D Global Placement

The objective of compact 2D placement is to offer a good solution in the projected 2D space, which can then be utilized in the subsequent 3D placement stage. The primary focus at this stage is to minimize the wirelength and to optimize the normalized cell density. What sets this problem apart from previous works is that the cells are

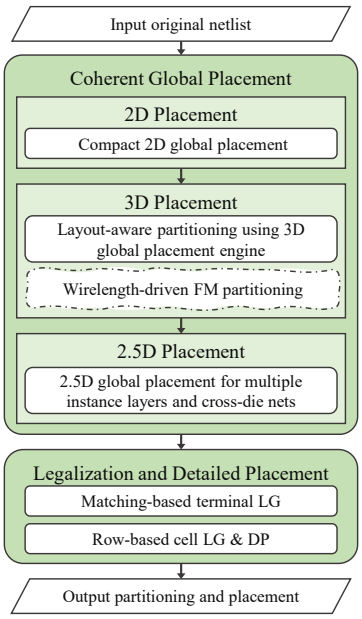

Fig. 3: The overall flow of our proposed framework.

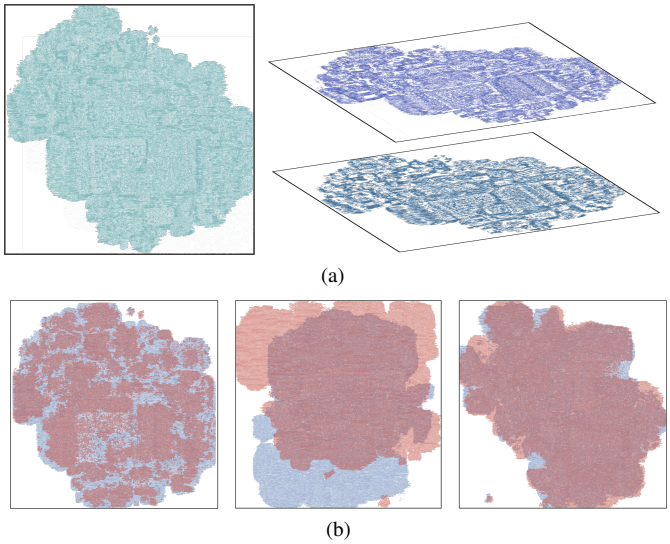

Fig. 4: (a) An example of local density-aware partitioning. (b) Placement results of 3 partitioning methods. Unbalanced utilization (Left), Cutsize-driven partitioning (Mid) and local-density-driven (Right) where the colors represent cells on different dies.

defined in multiple technologies so that the total areas for each die are varying. A good partitioning as depicted in Fig. 4b on the right has a balanced area on each die and thereby can exploit the benefit of stacking 2 dies vertically, whereas an unfavorable partitioning on the left introduces large wirelength overhead for inter- and intro-die connections. To tackle this issue, we normalize the cell sizes in a way that ensures equal actual utilization for both dies, so that the cell areas on the two dies are appropriately aligned.

Let $A^c = \sum_{v_i \in V} w_i^c \times h_i^c$ be the total cell areas of die c, and A^{die} be the die area. We set the utilization $U^0 = U^1 = \frac{A^0 \times A^1}{(A^0+A^1)\times A^{die}}$ according to our assumption of equal utilization. Note that if utilization U^c exceeds the maximum utilization U_M^c for die c, we will clamp the utilization as $U^c = U_M^c$ and update U^{1-c} accordingly. This will introduce some small imbalance. We then define a normalization factor $\tau = \frac{U^0 \times A^{die}}{A^0}$, denoting the percentage of cell areas that are placed on die 0 that can achieve the expected utilization.

The normalized cell sizes for this compact 2D placement are computed as:

$$\bar{s} = \tau \times s^0 + (1-\tau) \times s^1 \tag{6}$$

Then we adopt the 2D non-linear placement given in Equation (2) to place the normalized cells on a single layer with doubling of the target density. The placed results serve as an initial solution so that the cell areas can be well-aligned with the subsequent partitioning process.

B. Layout Aware Partitioning Using 3D Global Placement

With an initial compact 2D placement solution, we want to partition the circuit in a way that preserves the placement quality to the biggest extent possible. Conventional partitioning methods typically minimize costs such as cutsize for a weighted hypergraph and lack the physical layout consideration. Lower cutsize does not necessarily guarantee better HPWL for 3DICs since it may not facilitate sufficient vertical connections.

An example of the global placement result of three different partitioning methods is shown in Fig. 4b. The cutsize-driven partitioning in the mid pushes the circuit in a way to have the least number of vertical connections. The layout shows significant separation between the cells on the two dies and leads to extra interconnection wirelength because the cells are more flattened rather than stacked. In contrast, an ideal partitioning on the right establishes a larger number of vertical connections so that the overall layouts of the 2 dies can be well-aligned and consequently resulting in shorter wirelength. The expected partitioning can be achieved by taking the local cell density into consideration based on the compact 2D placement solution as illustrated in the two top sub-figures of Fig. 4a.

To achieve a partitioning that maintains the good quality of the compact 2D solution and ensures balanced local cell density for each die, we leverage the capabilities of our placement engine by expanding the spatial domain into 3D and performing placement along the expanded z-direction.

In our model, each cell occupies a continuous 3D space (x, y, z), where the x, y are the projected 2D position and z is the extended direction. The extended 3D Poisson's equation with spatial density $\rho(x, y, z)$ is defined as:

$$\begin{cases} \bigtriangledown \cdot \bigtriangledown \psi(x,y,z) = -\rho(x,y,z), \\ \hat{\mathbf{n}} \cdot \bigtriangledown \psi(x,y,z) = 0, (x,y,z) \in \partial R, \\ \iiint_R \rho(x,y,z) = \iiint_R \psi(x,y,z) = 0 \end{cases} \tag{7}$$

The gradient of the spatial density is numerically solved with the discrete cosine transformation (DCT):

$$\begin{cases} E_x = \sum_{j,k,l} \frac{a_{j,k,l} w_j}{w_j^2+w_k^2+w_l^2} sin(w_j x) cos(w_k y) cos(w_l z) \\ E_y = \sum_{j,k,l} \frac{b_{j,k,l} w_k}{w_j^2+w_k^2+w_l^2} cos(w_j x) sin(w_k y) cos(w_l z) \\ E_z = \sum_{j,k,l} \frac{a_{j,k,l} w_l}{w_j^2+w_k^2+w_l^2} cos(w_j x) cos(w_k y) sin(w_l z) \end{cases} \tag{8}$$

By applying the 3D electrostatic field as pushing forces, cells spread out to the whole 3D space evenly. In our flow, the cells are pre-placed in the 2D space (from the compact 2D placement), and the 3D placer serves as a partitioner, so we set $E_x = 0, E_y = 0$ to freeze the cells in the x, y directions and only apply gradients in the z-direction to spread the cells. The z positions are then discretized into binary values as the tier assignment [10].

As the standard cells on each die are defined in the same logical library but with different technologies, the difference in cell sizes will bring bias between the cell areas of the two tiers. To achieve area balance correctly, we will initialize all the cells by the top die technology cell sizes s^1 and whenever a cell is moved to the bottom half, its size will be enlarged to its true size s^0 for the bottom die technology.

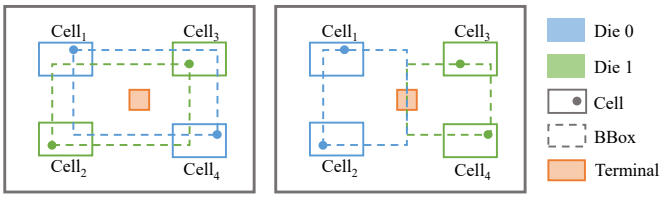

Fig. 5: A 2D projected placement of a bad (Left) and a good (Right) partitioning.

A naive wirelength model of the 3D placement is the sum of the independent HPWL in the 3 directions, $HPWL(v) = HPWL_x(v) + HPWL_y(v) + HPWL_z(v)$. Since we freeze the x, y-direction and optimize only the wirelength of $HPWL_z(v)$, the objective is similar to achieving min-cut partitioning. So it becomes necessary to relax the wirelength objective and allow some nets to be cut.

In this scenario, cutting nets with shorter wirelength will be more favorable. This is because larger nets are more likely to generate higher HPWL overhead if they are cut. Therefore in our algorithm, we define a net weight ω according to a net's size and degree $\mathcal{D}$ so that the nets with the least possible influence on wirelength are more likely to be cut. Consider a random tier assignment, a net of degree $\mathcal{D}$ will have a probability of $1 - 0.5^{\mathcal{D}-1}$ to be cut. With the most pessimistic assumption, a cut net will double the 2D projection HPWL. A step function of a weight coefficient $\alpha < 1$ is used to further discourage high-degree nets to be cut with a threshold T. We thus define a net weight $\omega(\mathcal{D}) = prob(\mathcal{D}) \times step(\mathcal{D})$ where:

$$\begin{cases} prob(\mathcal{D}) = (1 - 0.5^{\mathcal{D}-1}) \times HPWL_{x,y}(v) \\ step(\mathcal{D}) = \alpha \text{ if } \mathcal{D} < T; 1 \text{ otherwise} \end{cases} \tag{9}$$

The variable α is precomputed. The objective of the 3D placement is then:

$$\min_{v} \sum_{e \in E} sHPWL_e(v) + \lambda D(v) \tag{10}$$

where $sHPWL(v) = \omega \times HPWL_z(v)$. In this way, the 3D placement result can preserve the good quality of the compact 2D placement solution while maintaining local cell density balance, and can serve as a good initial partitioning solution.

C. Multi-electrostatic 2.5D Global Placement

To further optimize the cross-die HPWL and remove overlaps, we adopt a multi-electrostatic 2.5D placement to refine the placement solution on each layer. A multi-electrostatic-based method is able to solve placement problem with fence constraints [11]. We extended the model to place cells and bonding terminals on multiple layers simultaneously, which are the bottom die, top die and bonding terminal layer respectively $D = (D^0, D^1, D^b)$. The objective of the 2.5D multi-electrostatic placement is to minimize the cross-die HPWL of the partitioned circuits such that the cells and terminals on the three layers have minimal overlap. To satisfy the spacing constraint of the bonding terminals, we instantiate each terminal with a size of $(w^b + s^b, h^b + s^b)$, where w^b, h^b, s^b are the width, height and spacing respectively. The densities of the three layers are then relaxed as a penalty term in the objective function:

$$\min_{v, v^b} \sum_{c \in \mathcal{C}} HPWL(v^c \cup v^b) + \langle \lambda, D \rangle \tag{11}$$

where $\lambda = (\lambda_0, \lambda_1, \lambda_2)$ is a vector of the density terms in each layer and $\langle \cdot, \cdot \rangle$ denotes the inner product. Fillers are inserted into each layer independently The objective is to optimize the total wirelength of two reconstructed netlists with bonding terminals, $v^0 \cup v^b$ and $v^1 \cup v^b$ while removing the overlaps of instances in each layer.

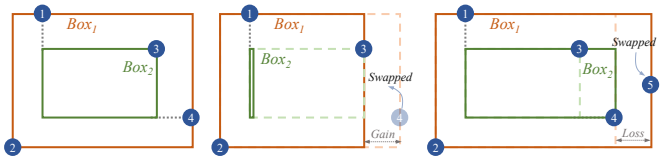

Fig. 6: An illustration of our proposed bounding box (BBox) updating scheme in wirelength-driven FM partitioning: (a) current BBox on a die. (b) updated BBox and gain when swapping a cell to the other die. (c) updated BBox and loss when swapping a cell from the other die to this die.

D. Wirelength-driven FM Partitioning

As mentioned in Section III-B, improper partitioning of a net might double its contribution to the HPWL. An example of a bad and a good partitioning is shown in Fig. 5. Suppose there are 4 standard cells in the shown positions and the outline color of the cell denotes its tier assignment. The partitioning on the left almost doubles the HPWL compared to the original single-chip HPWL, whereas the right partitioning significantly reduces the cross-die HPWL. To handle this issue, we propose an efficient wirelength-driven FM partitioning in Algorithm 1 after the 3D placement stage to further improve the wirelength [12]. The input of the algorithm is a partitioned netlist, obtained by the 3D placement in Section III-B. All the cells are set to be free at the beginning. Several lists are initialized, where the $GainList$ is the reduction in HPWL if we move a cell from one die to the other and $CutList$ is the reduction of the number of cuts. Every time we move a $cell$ to the opposite die, the gain of its neighboring cell (connected by the same net) may need to be updated. Apart from the traditional FM algorithm that targets minimizing the cut size, we also target minimizing the total HPWL on the two tiers directly. A naive algorithm for updating the gain runs in a high time complexity and is very time-consuming.

To speed up the gain updating, we maintain and reuse the net bounding box information to avoid redundant netlist traversal. In our proposed algorithm UPDATEGAINHPWL, two boxes are initialized for each sub-net on the 2 layers. The variable B_1 is the outermost bounding box of the net in each die, and B_2 is the second outermost bounding box being the updated bounding box if we remove the cells that lie on the original bounding box, as depicted in Fig. 6. Suppose we try to move $cell_n$ from die c to $1 - c$. If $cell_n$ lies on $B_1[c]$, it means that $cell_n$ is one of the outermost cells of this net on die c. By moving $cell_n$ to die $1 - c$, the second outermost box $B_2[c]$ will become the bounding box. On the other hand, if $cell_n$ lies outside $B_1[1 - c]$, $cell_n$ becomes the outermost cell of this net on die $1 - c$. The change in HPWL can be easily calculated by using these bounding boxes information. We pop the cell with the maximum weighted gain out of the $FreeCell$ list to swap by POPMAX function according to the $Gain, Cut$ and $AreaWeight$ in the current status.

Now we introduce the $AreaWeight$ in the POPMAX function. When swapping cells, the utilization on each die are changing. To keep a stable utilization as expected and to maintain the maximum utilization constraint, we propose a utilization-driven area weight η to adjust the wirelength-driven cell gain dynamically. We first define the relative utilization ratio $r = \frac{Area^0}{Area^1}$, where $Area^c = \sum_{v_i \in V^c} w_i^c \times h_i^c$ are the cell areas of each die, and r changes according to the tier assignment. We then assign a cell at die c with an area weight η^c, formulated as follows,

$$\eta^c = exp(\frac{\Delta R^c}{r_u - r_l}) \tag{12a}$$

$$\Delta R^c(r) = \begin{cases} r - r_l, & \text{if } c = 0 \\ r_u - r, & \text{otherwise} \end{cases} \tag{12b}$$

Algorithm 1 Wirelength-Driven FM Partitioning

$FreeCell \leftarrow cells$
Initialize $GainList, CutList, AreaWeightList, tier, pos$
while $FreeCell \neq \emptyset$ **do**
  $cell \leftarrow$ PopMax()
  UpdateGainHPWL($cell$)
  $FreeCell \leftarrow FreeCell \setminus cell$
  Track($Gain, Cutsize$)
ApplyBestSolutionTrack()

function UPDATEGAINHPWL($cell$)
  $tier[cell] \leftarrow 1 - tier[cell]$ ▷ Swap to the other die
  for each $net : cell \in net$ **do**
    $box \leftarrow xl = \infty, yl = \infty, xh = -\infty, yh = -\infty$
    $B_1 \leftarrow [box, box]$ ▷ The out-most bbox for 2 dies
    $B_2 \leftarrow [box, box]$ ▷ The second out-most bbox for 2 dies
    for each $i : cells[i] \in net$ **do**
      $c \leftarrow tier[i]$
      $(x, y) \leftarrow pos[i]$
      if $x > B_1[c].xh$ **then** $B_1[c].xh = x$
      else if $x \leq B_1[c].xh$ and $x > B_2[c].xh$ **then**
        $B_2[c].xh = x$
      if $x < B_1[c].xl$ **then** $B_1[c].xl = x$
      else if $x \geq B_1[c].xl$ and $x < B_2[c].xl$ **then**
        $B_2[c].xl = x$
    for each $i : cells[i] \in net$ **do**
      $B_s \leftarrow B_1$ ▷ The intermediate bbox
      $c \leftarrow tier[i]$
      $(x, y) \leftarrow pos[i]$
      if $x == B_1[c].xh$ **then** $B_s[c].xh = B_2[c].xh$
      if $x == B_1[c].xl$ **then** $B_s[c].xl = B_2[c].xl$
      if $x > B_1[1-c].xh$ **then** $B_s[1-c].xh = x$
      if $x < B_1[1-c].xl$ **then** $B_s[1-c].xl = x$
      $GainList[i]$ += $B_1 - B_s$

where r_l and r_u are the pre-computed lower and upper bound of r, which are only related to the given utilization constraint and design technology. Thus, $\eta^c(r)$ is only determined by the variable r and the tier assignment c. A normalization factor ϕ is applied to stress the number of cuts in the objective.

The weighted gain for a cell i at die c is formulated as follows,

$$WeightGainList[i] = \eta^c \times (GainList[i] + \phi CutList[i]) \quad (13)$$

which is applied in the POPMAX function and serves as a key criterion to select cells to swap. In Equation (13), η^c measures the importance of utilization constraint during the swapping process. When the value of η^0 is close to η^1, the POPMAX function will choose a cell with a large wirelength-driven gain. When more cells are swapped from die 0 to die 1, the ratio r will decrease, and we have $\eta^0 < \eta^1$. In this case, a larger η^1 will encourage a swap from die 1 to die 0 so that the areas are stable, and vice versa.

CoPlace iteratively swaps cells and the best solution is selected as $argmax(Gain \times (\#Nets - Cutsize))$, which can achieve the biggest HPWL improvement with acceptable cutsize overhead.

E. Legalization and Detailed Placement

We perform row-based legalization for cells [13]. The legalization consists of Greedy legalization and Abacus legalization. A matching-based legalization for bonding terminals is proposed. We discretize the die into uniform tile sites and assign a local search region to each bonding terminal. A min-cost bipartite matching is performed between the terminal nodes and the site nodes [14], where the cost of an edge is the HPWL overhead to put a terminal in a site. After finding legal positions, detailed placement is applied to further refine the solution quality, which consists of 3 strategies: global swap, local reordering, and independent set matching [13] .

TABLE I: Benchmark statistics

design_name	#Cell	#Nets	max #Term	$U_M^0(\%)$	$U_M^1(\%)$
case2	2735	2644	2000	70	75
case2_hidden	2735	2644	2000	79	79
case3	44764	44360	36481	78	78
case3_hidden	44764	44360	36100	68	78
case4	220845	220071	183612	66	70
case4_hidden	220845	220071	178929	66	76

IV. EXPERIMENTS

We develop our algorithms using C++, and the experiments are conducted on a Linux machine with a 2.90GHz Intel Xeon CPU. We test our performance on the ICCAD 2022 contest benchmarks [2] of 3D Placement with D2D Vertical Connections. The evaluator provided by the contest organizer is used to validate the legality and quality of the solution. Eight threads are used for CPU computation.

A. Comparison with the Top-3 Teams

The benchmark consists of 7 cases, in which the toy case is removed in our experiments. In Table I, we show the statistics of each case, including the number of cells, the number of nets, the maximum number of terminals allowed, and the maximum utilization. The cases encompass different design scales, technologies and terminal sizes, presenting a diverse view of the problem.

The HPWL, number of terminals, and runtime results are presented in Table II and the normalized ratio is shown in the last row. The runtime of top 3 teams are reported from the contest. To narrow the influence caused by different test machines and make the comparison fair enough, we normalize our runtime by $\frac{RT}{RT_{local}}$, where RT is the reported runtime of the top teams and RT_{local} is their runtime obtained on our local server. Results show that our proposed framework outperforms the top 3 teams of the ICCAD 2022 contest with better HPWL and faster runtime. In specific, we achieve 3.35% improvement in the total wirelength compared with the first place while also being 35% faster and using 45% fewer terminals. Results show that our proposed framework performs better on larger cases possibly because our 3D placement-based partitioner can achieve a better global view for large-scale designs. The runtime breakdown of case4 in Fig. 7 shows the percentage of runtime for each component of our framework.

Note that although the evaluation metric doesn't focus on the possible influence on PPA like timing and bandwidth introduced by an improper number of terminals, our algorithm still provides the capability to reduce the critical terminals by enlarging the cutsize factor ϕ in Section III-D and the net weight coefficient α in Section III-B.

B. Flexibility with Varying Terminal Size

To demonstrate the flexibility of our proposed framework in handling varying terminal sizes, we show in Fig. 8 the total HPWL of case4 with different terminal sizes. Our approach outperforms the top 3 teams for all three types of terminals with varying sizes. Smaller size allows more terminals to be integrated for vertical connections, and thus will result in shorter wirelength. The result indicates that the net weight coefficient α in Equation (9) adapts to the terminal sizes, resulting in using an appropriate number of terminals to minimize the wirelength.

TABLE II: Experimental results on the ICCAD 2022 contest benchmarks[2]. Runtime is normalized for a fair comparison.

Benchmark	1st Place Team			2nd Place Team			3rd Place Team			Ours		
	HPWL	#Terminal	RT (s)	HPWL	#Terminal	RT (s)	HPWL	#Terminal	RT (s)	HPWL	#Terminal	RT (s)
case2	2072075	1131	45	2080647	477	14	2097487	163	10	**2064852**	195	52
case2_hidden	2555461	1083	40	2735158	687	15	2644791	151	9	**2530288**	321	54
case3	30580336	16820	635	30969011	11257	473	33063568	14788	145	**30490760**	7597	338
case3_hidden	27650329	16414	412	27756492	8953	482	28372567	11211	133	**26998021**	8213	305
case4	281315669	84069	2580	274026678	51480	3284	281378079	46468	925	**270141472**	58708	1783
case4_hidden	301193374	84728	2239	308359159	59896	3283	307399565	58860	983	**292195136**	65395	1862
Sum	645367244	204245	5951	645927145	132750	7551	654956057	131641	2205	**624420529**	140429	4395
Ratio	1.034	1.454	1.354	1.034	0.945	1.718	1.049	0.937	0.502	**1.000**	1.000	1.000

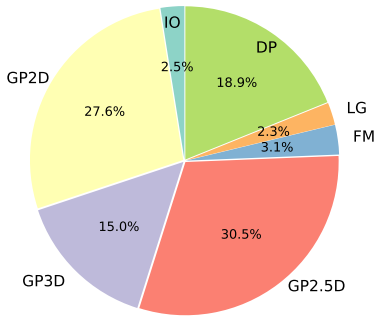

Fig. 7: Runtime breakdown of case4.

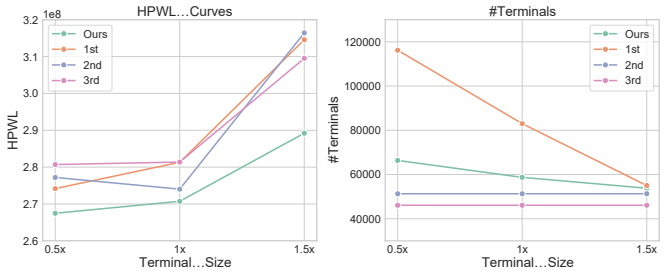

Fig. 8: The HPWL and the number of terminals with varying terminal size.

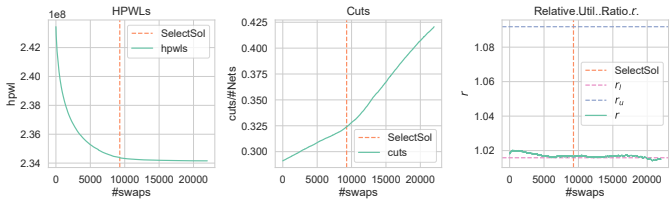

Fig. 9: The history curves of HPWL, Cutsize, and r in a pass of swapping cells of case4.

C. Effectiveness of Wirelength-driven FM Partitioning

In Fig. 9, we demonstrate the effectiveness of the partitioning algorithm proposed in Section III-D. The $Gain$ and $Cutsize$ of each iteration are tracked and the best solution indicated by the vertical orange dashed line is selected. The solution achieves significant HPWL reduction with less than 5% cells being swapped and less than 4% more nets being cut. Besides, we also track the relative utilization ratio r defined in Section III-D. As the curve shown, the fluctuation of r is small when the number of swaps increases, which indicates the stableness of utilization. The results further imply our proposed area weight successfully maintains a stable and valid utilization of the two dies.

V. Conclusion

In this paper, we propose CoPlace, a coherent placement framework that effectively couples placement and partitioning for the 3D placement with D2D bonding terminals. Compared with the top teams of ICCAD 2022 contest, our framework exhibits better quality.

References

[1] L. Zhu, A. Chaudhuri, S. Banerjee, G. Murali, P. Vanna-Iampikul, K. Chakrabarty, and S. K. Lim, "Design automation and test solutions for monolithic 3d ics," vol. 18, no. 1, nov 2021.

[2] K.-S. Hu, I.-J. Lin, Y.-H. Huang, H.-Y. Chi, Y.-H. Wu, and C.-F. C. Shen, "2022 iccad cad contest problem b: 3d placement with d2d vertical connections," in *Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design*, ser. ICCAD '22. New York, NY, USA: Association for Computing Machinery, 2022.

[3] J. Cong, G. Luo, J. Wei, and Y. Zhang, "Thermal-aware 3d ic placement via transformation," in *2007 Asia and South Pacific Design Automation Conference*, 2007, pp. 780–785.

[4] S. Panth, K. Samadi, Y. Du, and S. K. Lim, "Placement-driven partitioning for congestion mitigation in monolithic 3d ic designs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 4, pp. 540–553, 2015.

[5] B. W. Ku, K. Chang, and S. K. Lim, "Compact-2d: A physical design methodology to build two-tier gate-level 3-d ics," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 6, pp. 1151–1164, 2020.

[6] J. Lu, H. Zhuang, I. Kang, P. Chen, and C.-K. Cheng, "Eplace-3d: Electrostatics based placement for 3d-ics," in *Proceedings of the 2016 on International Symposium on Physical Design*, ser. ISPD '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 11–18.

[7] M.-K. Hsu, Y.-W. Chang, and V. Balabanov, "Tsv-aware analytical placement for 3d ic designs," in *Proceedings of the 48th Design Automation Conference*. New York, NY, USA: Association for Computing Machinery, 2011, p. 664–669.

[8] 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 TCAD*, vol. 34, no. 5, pp. 685–698, 2015.

[9] L. Liu, B. Fu, M. D. F. Wong, and E. F. Y. Young, "Xplace: An extremely fast and extensible global placement framework," in *Proceedings of the 59th ACM/IEEE Design Automation Conference*, ser. DAC '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 1309–1314.

[10] J. Cong and G. Luo, "A multilevel analytical placement for 3d ics," in *2009 Asia and South Pacific Design Automation Conference*, 2009, pp. 361–366.

[11] J. Gu, Z. Jiang, Y. Lin, and D. Z. Pan, "Dreamplace 3.0: Multi-electrostatics based robust vlsi placement with region constraints," in *2020 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, 2020, pp. 1–9.

[12] C. Fiduccia and R. Mattheyses, "A linear-time heuristic for improving network partitions," in *19th Design Automation Conference*, 1982, pp. 175–181.

[13] Y. Lin, W. Li, J. Gu, H. Ren, B. Khailany, and D. Z. Pan, "Abcdplace: Accelerated batch-based concurrent detailed placement on multithreaded cpus and gpus," *IEEE TCAD*, vol. 39, no. 12, pp. 5083–5096, 2020.

[14] B. Dezs, A. Jüttner, and P. Kovács, "Lemon - an open source c++ graph template library," *Electron. Notes Theor. Comput. Sci.*, vol. 264, no. 5, p. 23–45, jul 2011.