H3D: Heterogeneous Resources Aware Global Router for Face-to-Face Bonded 3D ICs

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

***Abstract*—The emerging 3D ICs have brought challenges to traditional routers in deciding the intra-die and inter-die interconnects. Existing pseudo-3D flows rely on 2D IC routing engines, combined with a 3D via legalization step to complete routing. The separation of intra-die and inter-die routing significantly degrades solution quality. To address this issue, we propose H3D, the first native 3D global router designed for face-to-face bonded 3D ICs. H3D constructs a heterogeneous routing grid to represent routing and hybrid bonding terminal (HBT) resources. We develop dedicated dynamic programming-based algorithms to optimize the HBT number and locations for 3D Steiner trees on the heterogeneous grid. Specifically, H3D minimizes the number of HBTs by traversing the Steiner tree in reverse depth-first order and relocates HBTs to legal locations with minimal wirelength in reverse breadth-first order, leveraging the convexity of L1 distance for efficient optimization. Experimental results on various real-world designs demonstrate that H3D achieves 12% shorter wirelength, 25% fewer HBTs, and 1.9× speedup compared to state-of-the-art approaches.**

I. Introduction

The continuous drive for higher performance in integrated circuits (ICs) has pushed traditional 2D IC technology to its limits, necessitating the exploration of three-dimensional (3D) integration solutions. 3D ICs can overcome the scaling challenges while offering enhanced performance, reduced power consumption, and improved cost factor. Among various 3D integration technologies, face-to-face (F2F) bonding [1], [2] has gained significant attention due to its unique advantages in achieving ultra-high-density 3D interconnects with low cost. For example, Meta recently implemented the prototype logic-on-logic AR SoC [3] for augmented reality applications, achieving huge performance gains from F2F bonding.

Routing is a critical stage in the VLSI design flow, responsible for establishing physical connections between circuit elements using metal tracks across multiple layers. In 3D ICs, the routing challenge extends beyond traditional planar connections to encompass 3D interconnects [2], [4], [5] between stacked dies. Due to its complexity, the routing problem is divided into two sub-problems: global routing and detailed routing. Global routing partitions the layout into a grid map and constructs a rough routing solution in grid units. Guided by the global routing solution, detailed routing connects all the nets using actual metal tracks and vias while satisfying all the design rules. Global routing plays a vital role in reducing overall routing time and improving the quality of layout after detailed routing.

The global routers can be summarized into two categories: sequential and concurrent routers. *Sequential routers* [6]–[10] first route nets one by one, followed by iterative rip-up and reroute procedures with maze routing to resolve routing congestion. FastRoute [6] constructs congestion-aware Steiner trees and proposes an edge shifting technique to minimize overflow. NCTU-GR [8] enhances the rip-up and reroute stage by implementing a negotiation-based scheme with historical congestion data. These early routers typically compress the 3D routing grid into a 2D representation, perform 2D global routing, and then project the solution back to 3D through layer assignment [11]. In contrast, CUGR [9] and CUGR 2.0 [10] propose 3D pattern routing and generate optimized 3D topology by dynamic programming. Compared to L-shape-only pattern routing in CUGR [9], CUGR 2.0 [10] constructs a routing DAG for a net that can represent various routing patterns, achieving state-of-the-art performance. *Concurrent routers* tackle the routing problem holistically using techniques such as integer linear programming (ILP) [12], [13] or multi-commodity flow [14]. BoxRouter [12] attempts to address computational complexity through box expansion and progressive ILP, while GRIP [13] employs regional decomposition to create more manageable sub-problems. Despite these efforts, concurrent routers face significant scalability challenges with modern large-scale designs. Their solution quality heavily depends on the initial set of candidate routing solutions.

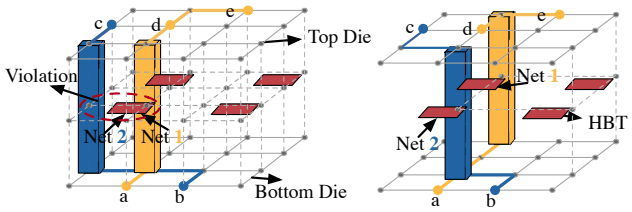
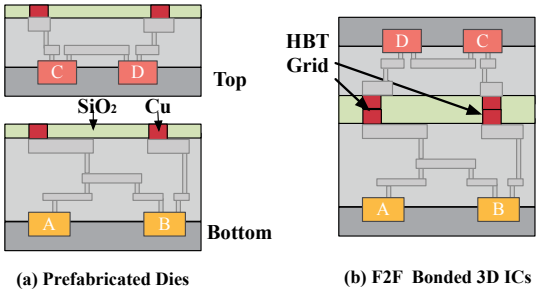


Fig. 1 Comparison between uniform and heterogeneous grid graph for F2F bonded 3D IC global routing. We compress the multiple metal layers of each die into a single layer for visualization. (a) Treating hybrid bonding terminals (HBTs) as vias leads to spacing violations. (b) Heterogeneous grid graph can explicitly model routing and HBT resources.

State-of-the-art physical design methodologies [15], [16] for F2F bonded 3D ICs leverage commercial 2D placement and routing tools. After placement and tier partitioning, cells are separated into two dies. To enable 3D routing using commercial tools designed for single-die designs, Compact-



(a) Prefabricated Dies (b) F2F Bonded 3D ICs

Fig. 2 Hybrid bonding technology. (a) Top and bottom dies are fabricated in parallel before bonding. (b) Top die is flipped, and two dies are aligned and bonded. Hybrid bonding terminals (HBTs) form a 2D grid array on the top-most metal layer.

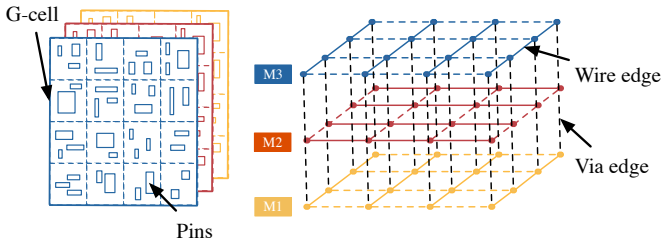


Fig. 3 Uniform grid graph model. There are three metal layers with 4 × 4 G-cells in each layer. The solid edge on the same layer indicates the preferred routing direction (horizontal or vertical).

2D [15] and Macro-3D [16] merge the metal stacks of two dies into a full 3D metal stack. The routing solution for the 3D design is generated based on the full metal stack. However, these pseudo-3D flows suffer from a fundamental limitation in their routing approach, stemming from an incorrect assumption regarding hybrid bonding terminal (HBT) pitch sizes. Originally developed for monolithic 3D integration [17], [18] and later adapted for F2F bonding, these flows assume that HBT pitches are comparable to monolithic via dimensions (approximately 0.1 µm). In reality, the achievable fine HBT pitch size ranges from 1 to 2 µm [3], an order of magnitude larger. To resolve the violations, a bipartite-matching-based 3D via legalization technique [19] is adopted for these flows, followed by rerouting the corresponding 3D nets. BTAssign [20] proposes to assign the bonding terminals for 3D nets before routing for legality, but it fails to consider accurate routing tree topology information. Moreover, directly applying existing 3D sequential routers [9], [10] to the full 3D metal stack faces the same problems, as the HBT pitch can be even larger than the grid cell size. Treating HBTs as vias leads to spacing violations, necessitating legalization, as illustrated in Fig. 1. Resolving the violations by maze routing is unscalable for designs with high 3D integration density. In summary, the separation of intra-die and inter-die routing significantly impairs the solution quality and efficiency.

To address above issues, this paper presents an efficient native 3D global router for F2F bonded 3D ICs. Our approach performs global routing on a heterogeneous grid graph that models both routing and hybrid bonding terminal (HBT) resources, as shown in Fig. 1. We develop dedicated algorithms to simultaneously optimize HBT placement and routing tree. Our contributions are summarized as follows.

- We propose the first native 3D global router, H3D, for F2F bonded 3D ICs. H3D performs routing directly on a heterogeneous grid graph, accurately accounting for both routing and HBT resource constraints.
- We develop dedicated dynamic programming algorithms to optimize HBT utilization. H3D minimizes the number of HBTs through a reverse depth-first traversal and relocates HBTs to legal locations with minimal wirelength using a reverse breadth-first traversal, leveraging the convexity of the L1 distance.
- Experimental results on extensive real-world benchmarks demonstrate that H3D achieves 12% shorter wirelength, 25% fewer HBTs, and 1.9× speedup compared to state-of-the-art approaches.

II. Preliminaries

A. Hybrid Bonding Technology

The advent of hybrid bonding technology has revolutionized F2F integration by enabling direct metal-to-metal and dielectric-to-dielectric bonding at the back-end-of-lines (BEOLs) of stacked dies. Each die is fabricated in parallel using conventional process before bonding. After planarization of both dies, the top die is flipped, and two dies are aligned, bonded to form the 3D design [15]. Fig. 2 illustrates the manufacturing process. With HBT pitches below 2 µm becoming feasible, hybrid bonding enables unprecedented levels of vertical integration density, making it particularly attractive for high-performance computing and memory-intensive applications.

However, the adoption of F2F hybrid bonding introduces unique challenges in the routing stage of physical design. Min-cut-based approaches are proposed for TSV-based 3D ICs to minimize TSV number due to its large pitch size. A simple pre-assignment [21], [22] of TSV locations is sufficient for these designs. Treating monolithic intertier vias [17] as regular vias enables direct application of 2D commercial tools for monolithic 3D ICs. But such approach will result in spacing violations for F2F bonded designs. In this work, we consider realistic HBT pitch size (3 µm) for F2F bonded 3D ICs. And HBT locations must conform to legal positions arranged in a 2D **grid array**, which improves manufacturability. Innovations are needed to make full use of the high integration density while satisfying design rules.

B. Global Routing

The grid graph model is widely adopted in global routing [6], [9], [10]. In this model, the 3D routing region is divided into global routing cells (G-cells), which represent coarse-grained regions of routing space, as illustrated in Fig. 3.

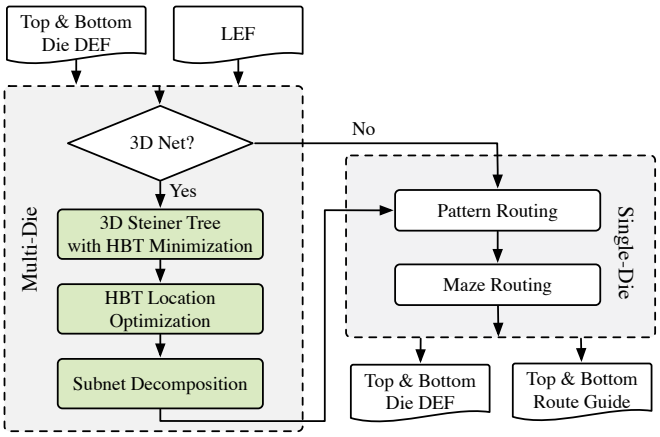


Fig. 4 Overall flow of H3D global router.

A grid graph $G(V, E)$ consists of vertices $v \in V$ representing G-cells and edges $e(u, v) \in E$ representing connections between adjacent G-cells. The edge $e(u, v)$ between vertices u and v on the same layer is *wire edge*, which has a preferred routing direction (horizontal or vertical). The capacity $c(u, v)$ of a wire edge is the number of tracks going through the edge. The edge $e(u, v)$ between vertices u and v with same 2D locations but on different layers is *via edge*. The capacity of a via edge [9] is derived from the capacity of its G-cell, $c(u)$, which is the average capacity of its two abutting wire edges. Accordingly, the resource of an edge or G-cell is defined as:

$$r(u, v) = c(u, v) - d(u, v), \tag{1}$$

$$r(u) = c(u) - d(u), \tag{2}$$

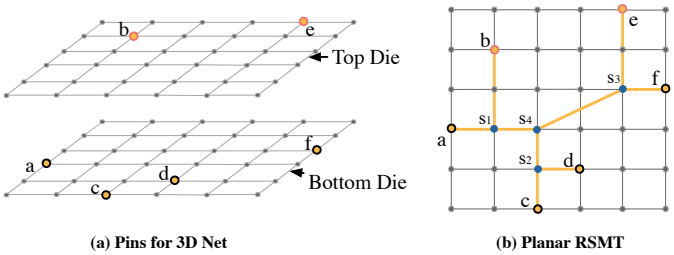
where $d(u, v)$ and $d(u)$ represent the demand on an edge or G-cell by routed nets. A routing overflow occurs when $r(u, v) < 0$. The objective of global routing is to find a path for each net on the grid graph such that all pins are connected without overflow, while minimizing total wirelength. Then, the detailed router utilizes the routing guide generated by the global router to establish precise connections between pins, minimizing the routing cost.

The grid graph model can naturally represent the full 3D metal stack of F2F bonded 3D ICs. However, the HBT pitch size can be larger than the G-cell size, and each HBT can only be used by one net. While one may attempt to model HBTs using G-cell capacity $c(u)$ in a uniform grid graph, this approach requires multiple rounds of maze routing to create detours for available HBTs, which cannot be effectively addressed by pattern routing. To overcome these limitations, a heterogeneous grid graph provides a better representation by explicitly modeling both routing and HBT resources.

III. H3D Global Router

A. Overview

H3D is a sequential global router designed to handle both 3D and 2D nets in F2F bonded 3D ICs. The overall flow of H3D is illustrated in Fig. 4. H3D takes the 3D placement results (top and bottom die DEF files) and technology LEF file as input, and outputs updated DEF files for the top and bottom dies with legal hybrid bonding terminal (HBT) placement and global routing guides.



(a) Pins for 3D Net (b) Planar RSMT

Fig. 5 Planar RSMT construction for a 3D net. Yellow points indicate pins of a 3D net. Blue points in (b) indicate Steiner points. The actual routing pattern for Steiner nodes s_3 and s_4 is not decided yet.

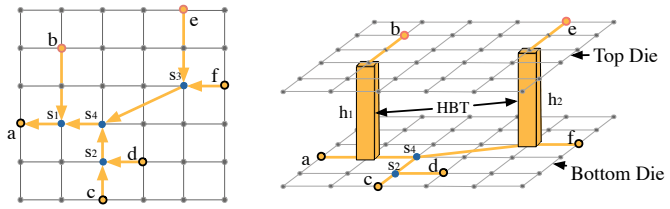
H3D processes nets in ascending order of their half-perimeter wirelength (HPWL). For 2D nets, where all pins are located on the same die, conventional pattern routing and maze routing algorithms (Section III-E) are used. For 3D nets, with pins distributed across both dies, H3D first constructs a 3D rectilinear Steiner minimum tree (RSMT) to minimize the number of HBTs (Section III-B). Then, leveraging the heterogeneous grid graph, a dedicated dynamic programming-based algorithm is applied to relocate HBTs to legal positions while minimizing total wirelength (Section III-C). Finally, 3D nets are decomposed into single-die subnets for final routing (Section III-D).

B. 3D Steiner Tree Construction with HBT Minimization

Similar to conventional 2D global routing, the routing topology for 3D nets must first be generated. At this stage, the rectilinear Steiner minimum tree (RSMT) is the de facto choice. However, unlike 2D routing, 3D routing must also account for minimizing the number of HBTs to conserve these scarce resources.

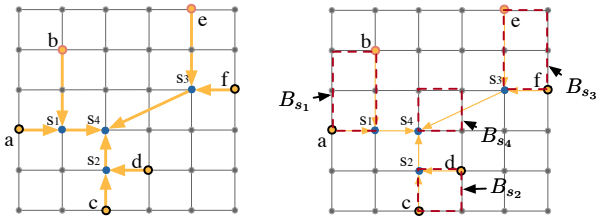
Planar RSMT Construction. Constructing an RSMT is a classic NP-hard problem, and significant efforts have been made to develop efficient algorithms. FLUTE [23], which uses a precomputed lookup table to enable fast and accurate RSMT construction, is widely adopted in modern global routers [9], [10]. While FLUTE has also been extended to construct 3D RSMTs [24], the additional complexity introduced by the z direction makes this approach practical only for a few pins. Hence, we use FLUTE [23] to efficiently construct a 2D RSMT and then apply a dynamic programming-based method to map the 2D topology into 3D while minimizing the number of HBTs.

We begin by projecting all pins of the 3D net onto a planar grid. At this stage, no HBTs are involved, so the process works on a conventional uniform grid, as illustrated in Fig. 5. Using FLUTE, we generate the RSMT, which connects all pins and may include additional Steiner points. The next step is to map the 2D topology into 3D with minimal HBT insertions.



(a) Two-Pin Net Ordering (b) HBT Insertion for 3D Net

Fig. 6 3D RSMT with minimal HBT insertions. Node a is chosen as the root in (a). After the dynamic programming-based die assignment, two HBTs are inserted in (b).



(a) BFS Ordering (b) Optimal Regions

Fig. 7 BFS ordering to determine the optimal region for each Steiner node. Steiner node s_4 is chosen as the root in (a). The optimal region is calculated in a bottom-up order in (b).

Dynamic Programming-based HBT Minimization. HBTs are required to establish connections between pins on different dies. At current stage, we temporarily ignore HBT placement constraints and focus solely on determining the routing topology. As a result, the process continues to work on the uniform grid graph. Given the planar RSMT, the minimal number of HBTs required is determined by the pin distribution across the two dies. This problem is similar to layer assignment [11] with two layers in 2D global routing. Therefore, we adopt a similar dynamic programming-based approach to solve this problem efficiently.

The multi-pin 3D net is broken into a set of two-pin nets according to the RSMT. Then, we determine the order in which the two-pin nets will be assigned. A degree one node will be randomly chosen as the root, and a depth-first search (DFS) traversal will be performed to visit the rest of nodes. The two-pin nets will be assigned to the corresponding die in the reverse DFS order, as illustrated in Fig. 6.

Let $par(v)$ denote the parent of node v, $ch(v)$ denote the set of children of v, and $Y(v)$ denote the subtree rooted at node v. Take Fig. 6 as an example, $par(s_4)$ refers to s_1, $ch(s_4)$ refers to $\{s_2, s_3\}$, and $Y(s_4)$ refers to the subtree consisting of $\{s_4, s_2, c, d, s_3, e, f\}$. We now define the building blocks of our dynamic programming algorithm. Let $mhc(v, l)$ denote the minimal HBT cost of $Y(v)$ when the edge $e(v, par(v))$ is assigned to die l.

For each node v, we can compute $mhc(v, l)$ for $l = 1, \ldots, L$ using recursive Equation (3), assuming there are L dies (*i.e.,* $L = 2$), and v has k children $ch(v) = \{v_{c(1)}, \ldots, v_{c(k)}\}$.

$$mhc(v, l) = \min_{\substack{1 \le l_1 \le L \\ \cdots \\ 1 \le l_k \le L}} \left(hc(v) + \sum_{j=1}^{k} mhc(v_{c(j)}, l_j) \right), \quad (3)$$

where $hc(v)$ is the number of HBTs required to connect all k child subtrees and pins located at v. This value can be calculated using Equation (4):

$$\begin{aligned} hc(v) &= \max(\zeta) - \min(\zeta), \\ \zeta &= \{d^h(v), d^l(v), l, l_1, \ldots, l_k\}. \end{aligned} \quad (4)$$

Here, $d^h(v)$ and $d^l(v)$ represent the highest and lowest die, respectively, of all pins located at v. After computing the minimal HBT cost for the root, we can trace back the solution to determine the die assignment achieving the minimal number of HBTs.

The number of children for a node is bounded by $k \le 4$. Enumerating all die combinations for the four child subtrees takes $O(L^4)$ time. Therefore, calculating $mhc(v, l)$ for $l = 1, \ldots, L$ requires $O(L^5)$ time. Updating all nodes across the tree will take $O(L^5|V|)$ time, as outlined in [11]. Instead of enumerating the die assignment for each child subtree, CUGR 2.0 [10] demonstrates that enumerating only the lowest and highest die is sufficient to update the cost, which significantly reduces the complexity to $O(L^2|V|)$. We adopt the latter updating scheme for better efficiency.

C. Dynamic Programming-Based HBT Location Optimization.

After constructing the 3D Steiner tree, the minimal number of HBTs are inserted into the routing topology on the uniform grid graph. However, the HBT placement constraints are ignored in the earlier stages. HBTs must be relocated to available locations in the HBT grid array. Leveraging our heterogeneous grid graph, we propose a novel dynamic programming algorithm that utilizes the convexity of the L1 distance to find legal HBT locations with minimal wirelength. Before diving into the discrete optimization approach, we first introduce the continuous scenario, which serves as the theoretical foundation for our method.

Continuous Rectilinear Steiner Tree Problem with Fixed Topology. The rectilinear Steiner tree problem with fixed topology (RSTPFT) [25], [26] is derived from the RSMT construction problem and serves as a post-optimization step to refine the output generated by heuristic algorithms for RSMT. In RSTPFT, we are given a Steiner tree T that connects a set of pins P placed in $\mathbb{R}^2$. The goal is to compute the coordinates (x, y) of the remaining Steiner nodes, $V(T) \setminus P$, such that the wirelength of the tree, under the given topology, is minimized. Sankoff and Rousseau give a linear-time algorithm solving RSTPFT [25]. The RSMT generated by FLUTE [23] is one of the solutions under the specific topology. As our problem involves relocating HBTs, it is closely related to this Steiner optimization problem.

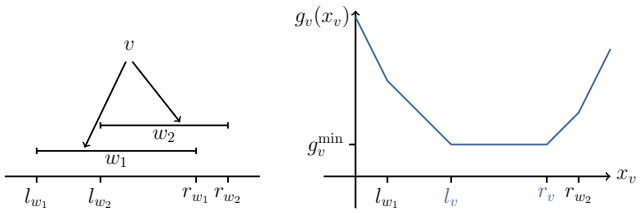


Fig. 8 An example of the piecewise linear and convex function $g_v(x_v)$. For visualization, we omit the direct pins to Steiner node v. The node v has two Steiner children w_1 and w_2. The $[l_v, r_v]$ is determined by $[l_{w_i}, r_{w_i}]$.

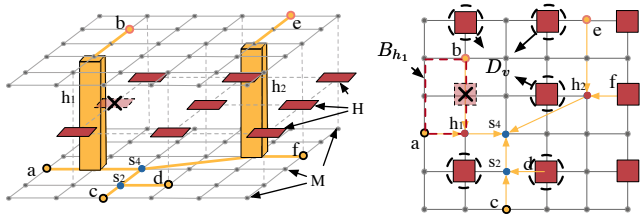
Since we only care about the locations of Steiner nodes, the relevant edges are those between Steiner nodes and between Steiner nodes and pins. To handle this more effectively, we traverse the nodes using breadth-first search (BFS) instead of the DFS approach used in Section III-B. First, a Steiner node is randomly selected as the root, and a BFS traversal is performed. When a branch encounters a pin node, the search stops, and no more nodes will be added to the current branch. If any Steiner nodes remain uncovered, the process is repeated, creating additional trees and forming a forest. This occurs when a pin node lies on the path between two Steiner nodes. As a result, all leaves in the trees are pin nodes, and the Steiner nodes serve as the inner nodes. Fig. 7 shows the final traversal order.

Since the wirelength is measured using the L1 distance, the problem can be solved independently for the x and y coordinates. Therefore, we demonstrate our method using the x coordinate. For the children of a Steiner node v, $ch(v)$ consists of two subsets: $pch(v)$ and $sch(v)$. Here, $pch(v) \subset P$ represents pin nodes, while $sch(v) \subset V \setminus P$ represents Steiner nodes. For a node $w \in pch(v)$, its x-coordinate, x_w, is a fixed value. For a node $w \in sch(v)$, x_w is a variable that can be optimized. We now define a function $g_v(x_v)$ that represents the minimal x-wirelength of the subtree $Y(v)$ rooted at the Steiner node v, given its x-coordinate has value x_v. $g_v(x_v)$ can be computed using Equation (5):

$$g_v(x_v) = \sum_{w \in pch(v)} |x_v - x_w| + \sum_{w \in sch(v)} \min_{x_w} \{g_w(x_w) + |x_v - x_w|\}. \tag{5}$$

$g_v(x_v)$ can be divided into two components: $f_v(x_v) = \sum_{w \in pch(v)} |x_v - x_w|$, representing the total distance to all child pins, and $h_v(x_v) = \sum_{w \in sch(v)} \min_{x_w} \{g_w(x_w) + |x_v - x_w|\}$, representing the sum of the minimal wirelength costs for connecting to the child subtrees. It is obvious that $f_v(x_v)$ is a piecewise linear and convex function. And it can be shown inductively that $h_v(x_v)$ and $g_v(x_v)$ have the same property.

We define the interval $[l_v, r_v] := \{x_v \in \mathbb{R} \mid g_v(x_v) = g_v^{\min}\}$, in which $g_v(x_v)$ achieves its minimal value $g_v^{\min}$. It follows that $h_v(x_v)$ can be rewritten as $h_v(x_v) = \sum_{w \in sch(v)} \big(g_w^{\min} + \max\{0, x_v - r_w\} + \max\{0, l_w - x_v\}\big)$. For child $w \in pch(v)$, we can set $g_w^{\min} = 0$ and $l_w = r_w := x_w$.



(a) Heterogeneous Grid Graph (b) Candidate HBT Locations

Fig. 9 Discrete HBT location optimization. (a) The HBT array and uniform grid have different granularity. The HBT with cross indicates it is occupied by other nets. (b) We consider additional closest points to B_v in 8 directions in our candidate set S_v^c. The selected additional HBTs D_v are marked with circles.

Substituting these values, we have:

$$g_v(x_v) = \sum_{w \in ch(v)} \big(g_w^{\min} + \max\{0, x_v - r_w\} + \max\{0, l_w - x_v\}\big), \tag{6}$$

which is a piecewise linear and convex function, as illustrated in Fig. 8. Moreover, the extreme points of the interval of $g_v(x_v)$ are given by the interval boundaries $\{l_w, r_w \mid w \in ch(v)\}$ of its k children. With $\{l_w, r_w \mid w \in ch(v)\} = \{a_1 \leq \cdots \leq a_{2k}\}$, we have $l_v = a_k, r_v = a_{k+1}$. This leads to a linear-time algorithm: interval $[l_v, r_v]$ can be computed in a bottom-up order by applying constant-time median search on the child intervals. Then, in a top-down order, we can choose $x_v \in [l_v, r_v]$ such that the distance $|x_{par(v)} - x_v|$ to its parent $par(v)$ is minimized. The computation in y-coordinate is similar. Then, for each Steiner node v, we have the optimal region $B_v = [l_v^x, r_v^x] \times [l_v^y, r_v^y]$, where the superscript indicates the x or y interval, containing the possible locations for v, as shown in Fig. 7.

Discrete HBT Location Optimization. HBTs must reside within the HBT grid array to satisfy placement constraints. Furthermore, the pitch size of HBT grid is larger than the size of uniform grid cells, resulting in a discrete optimization problem. Moreover, HBTs used by routed 3D nets cannot be shared with other nets, which makes the coarse HBT grid array non-uniform. Consequently, the x- and y-coordinates become inseparable in the optimization process.

We first introduce the construction process of the **heterogeneous grid graph**. Since the HBT array has a larger pitch size, we use a coarse grid graph to represent the HBT resources. HBTs occupied by routed nets must be removed, resulting in a non-uniform coarse grid. To enable efficient spatial queries, we represent the coarse grid using a quad-tree data structure. This allows us to quickly identify the closest available HBT locations relative to the uniform grid where the pin and Steiner nodes are located. Fig. 9 illustrates our heterogeneous grid graph.

According to the routing topology generated in Section III-B, HBTs can be inserted at Steiner nodes or pin nodes if the pin directly connects to a pin on the other die. We also

Algorithm 1 DP-based HBT Location Optimization

Input: Steiner Tree T with topology in Section III-B, root r.
Output: Steiner Tree T with legal HBT placement.

1: ComputeSolution(r);
2: TraceSolution(r); ▷ top-down order
3: **function** ComputeSolution(v)
4: **for** each child $w \in ch(v)$ **do**
5: ComputeSolution(w); ▷ bottom-up order
6: **end for**
7: Compute B_v in Equation (6); ▷ continuous scenario
8: $S_v^c \leftarrow (B_v \cap S_v) \cup D_v$; ▷ candidate HBT points
9: Compute $g_v(s), \forall s \in S_v^c$ in Equation (7);
10: **end function**

consider such HBTs in our BFS traversal. If an inner node v is an HBT, v can only be placed within the set of currently available HBT array locations, denoted as $S_v := H$. If v is a Steiner node, v can be placed at any point on the uniform grid, denoted as $S_v := M$. In the general discrete optimization problem, the minimal wirelength of the subtree $Y(v)$, given that v is placed at point $s \in S_v$, can be calculated as follows:

$$g_v(s) = \sum_{w \in pch(v)} \|s-w\|_1 + \sum_{w \in sch(v)} \min_{t \in S_w} \{g_w(t) + \|s-t\|_1\}. \tag{7}$$

One can use Equation (7) to compute $g_v(s)$ for $\forall s \in S_v$ in a bottom-up order, then trace back to get the optimal HBT locations. But such approach will take $O(|V||S_v|^2)$ time.

It is unnecessary to check the entire set S_v to calculate the minimal wirelength for v. We can utilize the optimal region B_v in continuous scenario to select a candidate set $S_v^c \subset S_v$, where $|S_v^c| \ll |S_v|$, to accelerate the dynamic programming (DP) algorithm. Firstly, we add the available HBT points, $B_v \cap S_v$, within the optimal region B_v to the candidate set. This process is efficiently supported by our quad-tree-based heterogeneous grid graph, which enables rapid spatial queries. In the discrete scenario, extra points need to be evaluated. Namely, we collect the closest points to B_v in 8 directions, D_v, to the candidate set S_v^c, as illustrated in Fig. 9. Hence, we consider the candidate set $S_v^c := (B_v \cap S_v) \cup D_v$ in our approach.

Our detailed dynamic programming-based HBT location optimization algorithm is presented in Algorithm 1. It takes $O(|V||S_v^c|^2)$ time, where $|S_v^c|$ is a small constant. After the bottom-up computation, the HBT locations for all HBT nodes can be traced back from the root (line 2). In a special case where multiple HBTs in the same net are assigned to the same HBT point, it indicates that relocating HBTs to farther points will result in excessive detours. Our DP-based method considers such scenarios. Finally, we remove all the HBT points used by the current net from HBT grid array H, ensuring that no other nets can use them in the following routing steps.

TABLE I The statistics of 7 evaluated real-world designs. The 3D nets consist of pins located on different dies. The ratio indicates the proportion of 3D nets relative to the total number of nets.

Bench.	#Cells	#Macros	#Nets	#3D Nets	ratio
rocket [27]	24647	2	26084	1940	0.07
aes [28]	13158	0	13158	381	0.03
ethmac [29]	23714	0	23900	3487	0.15
jpeg [30]	175352	0	205921	16001	0.08
mor1kx [31]	56951	0	57521	2927	0.05
ariane [32]	145684	132	157129	12718	0.08
BP [33]	273187	24	265585	3615	0.01

TABLE II The physical information of the technology node. Metal6 is the top-most metal layer of each die. The pitch size of HBT is larger than G-cell size.

Name	Metal6	Via5	HBT	G-cell
pitch (μm)	0.28	0.3	3.0	2.1

D. Subnet Decomposition

After minimizing the number of HBTs and optimizing their locations, 3D nets are routed by the 3D Steiner trees, with all HBTs placed at legal locations. Then, we can decompose the 3D net into single-die subnets for final routing. The decomposition process is straightforward: during a reverse DFS traversal, the tree is split at each HBT node to form single-die subnets. HBTs are treated as special pins in both the top and bottom subnets.

E. Pattern and Maze Routing

We employ conventional pattern routing and maze routing techniques to perform global routing for single-die nets across multiple metal layers. CUGR 2.0 [10] constructs routing DAGs that represent various routing patterns, including L-shape, z-shape, and monotonic routing, achieving excellent performance. Therefore, we adopt the 3D pattern routing algorithm from CUGR 2.0. If there is routing overflow after pattern routing, the sparse graph maze routing algorithm [10] is utilized to perform rip-up and reroute.

IV. Experimental Results

A. Experimental Setup

We evaluated our global router on 7 real-world designs from OpenCores [35] and open-source RISC-V designs. The detailed statistics of the benchmarks are shown in TABLE I. These benchmarks represent a diverse range of design characteristics, including standard-cell-only designs and mixed-size designs with numerous macros and many routing obstacles.

The RTL designs were synthesized using Yosys [36] within the OpenROAD project [37], and the NanGate 45nm PDK [38] was adopted for both the top and bottom dies, where each die has 6 metal layers. The hybrid bonding terminal (HBT) pitch size was set to 3 μm, and all HBTs were constrained to be placed within the 2D HBT grid array. The detailed physical information of the technology node is listed in TABLE II.

TABLE III Comparison of experimental results between our router and state-of-the-art (SOTA) global routers on real-world designs. **RT** (s) stands for the runtime for global routing. Wirelength (µm) and via number are reported after detailed routing using the global routing guide. All the results are validated by a commercial tool with no DRC violations.

Bench.	FastRoute [34] + Legal [19]				CUGR 2.0 [10] + Legal [19]				Ours			
	WL	#Vias	#HBTs	RT	WL	#Vias	#HBTs	RT	WL	#Vias	#HBTs	RT
rocket	374421	200585	2800	9.163	369069	210891	2581	4.370	**336830**	205913	**2179**	**2.441**
aes	156045	119885	703	4.996	157048	125413	831	2.466	**151657**	123480	**526**	**1.335**
ethmac	557884	221480	5912	13.436	545503	238256	5738	6.338	**439322**	223486	**4954**	**2.975**
jpeg	3056841	1402528	25552	64.726	2679224	1441034	22064	33.901	**2514129**	1413384	**19507**	**18.648**
mor1kx	1516476	559389	10443	26.012	1380750	577274	9413	14.769	**1330973**	562741	**7664**	**7.684**
ariane	3684778	1377210	25837	72.921	3506201	1459846	25163	38.048	**3142161**	1396919	**19504**	**18.736**
BP	6401377	2018460	34430	113.263	5634172	2094227	28428	72.662	**4700736**	1986244	**24315**	**35.960**
Average	1.186	**0.989**	1.318	3.701	1.116	1.036	1.249	1.937	**1.000**	1.000	**1.000**	**1.000**

The 3D placements for all benchmarks were generated using state-of-the-art 3D placers [39] and [40], and the resulting placements were transformed into standard DEF format. To thoroughly evaluate our 3D global router, we generated benchmarks with a wide range of 3D net ratios. After global routing, detailed routing was performed using TritonRoute [41] in OpenROAD, utilizing the route guides generated by the global routers.

We implemented H3D using C++ based on the open-source global router CUGR 2.0 [10]. All the experiments were performed on a Linux machine with 20-core Intel Xeon Silver 4210R CPU (2.40GHz) and 24GB RAM using a single thread.

B. Comparison with SOTA Global Routers for 2D ICs

To evaluate the effectiveness of our method, we compared H3D with state-of-the-art (SOTA) global routers for 2D ICs, including CUGR 2.0 [10] and FastRoute [34] which is integrated into OpenROAD [37]. CUGR 2.0 and FastRoute cannot handle the HBT placement constraints as they treated HBTs as regular vias. Therefore, we implemented the windowed bipartite-matching algorithm in [19] to assign the HBTs to their nearest legal locations. Then, the affected nets were rerouted.

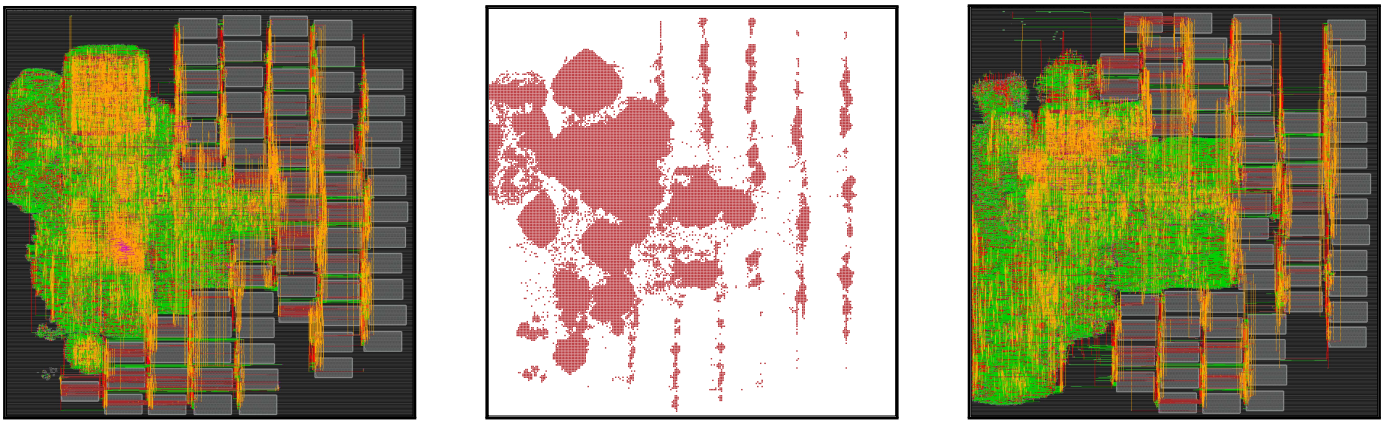
The experimental results are shown in TABLE III, including wirelength after detailed routing, via count, HBT count, and runtime. The final routing results were validated by both OpenROAD and a commercial tool with no DRC violations. Our H3D global router consistently achieved the best results in terms of wirelength and HBT count, while maintaining a comparable via count to FastRoute across all benchmarks, as shown in TABLE III. Specifically, compared to FastRoute, H3D achieved a 19% improvement in wirelength and a 32% reduction in HBT count. And H3D outperformed CUGR 2.0 by 12% shorter wirelength and by 25% fewer HBTs. Fig. 10 shows the layout of ariane after detailed routing using the global routing guide generated by H3D.

For multi-pin 3D nets, multiple HBTs are often required to achieve minimal wirelength in the Steiner tree. However, unlike regular vias, HBT resources are limited and must be placed on the top-most metal layer, which demands additional routing resources for connection. Given an RSMT, H3D inserts the minimal number of HBTs necessary to achieve the shortest wirelength. Moreover, existing 2D global routers require post-routing HBT legalization [19] to meet HBT placement constraints. This legalization process simply moves HBTs to the nearest legal locations without considering routing topology. In contrast, our dynamic programming-based algorithm efficiently relocates HBTs to legal positions by traversing the Steiner tree in a reverse BFS order. Leveraging the convexity of the L1 distance, H3D minimizes wirelength while maintaining legal HBT placement. Additionally, our method enables the merging of multiple HBTs in the same net to a single location, avoiding unnecessary detours caused by a lack of available HBT resources.

Regarding runtime, H3D is the fastest router among the evaluated methods. Compared to FastRoute and CUGR 2.0, H3D achieves 3.7× and 1.9× speedup, respectively. This runtime efficiency is attributed to two key factors. First, in contrast to the unscalable matching algorithm, H3D employs the efficient dynamic programming algorithm to find legal HBT locations. By leveraging the convexity of L1 distance, H3D evaluates only a small number of candidate locations. Second, H3D maintains the legality of HBT placement throughout the routing process, avoiding the need for rerouting affected 3D nets in 2D global routers.

C. Impact of HBT Pitch Size

The HBT pitch size plays a critical role in routing performance. A smaller pitch size allows for the use of more HBTs and reduces the distance between adjacent HBTs. A well-designed 3D global router must account for such scenarios to fully leverage these benefits. To evaluate H3D under these conditions, we evaluated it on the same designs using a smaller HBT pitch size of 2 µm. The experimental results are presented in TABLE IV. With the 2 µm pitch, H3D achieved a 5% improvement in wirelength compared to the 3 µm pitch scenario, while using 9% more HBTs. Instead of merging multiple HBTs into a single one, our method effectively utilizes the increased number of available HBTs to further optimize wirelength. Additionally, our dynamic programming-based approach demonstrates excellent scalability. Even with a greater number of HBTs to process, H3D maintains a similar runtime, showcasing its efficiency in handling more complex scenarios.



(a) Top Die (b) Hybrid Bonding Terminals (c) Bottom Die

Fig. 10 The layouts for top, HBT, and bottom dies of ariane.

TABLE IV Comparison of different HBT pitch sizes. **RT** (s) stands for the runtime for global routing. Wirelength (μm) and via number are reported after detailed routing using the global routing guide of our global router.

Bench.	Ours-2 μm				Ours-3 μm			
	WL	#Vias	#HBTs	RT	WL	#Vias	#HBTs	RT
rocket	**317907**	204640	2275	2.427	336830	205913	**2179**	2.441
aes	**150940**	123640	628	1.398	151657	123480	**526**	1.335
ethmac	**382196**	221734	5563	2.666	439322	223486	**4954**	2.975
jpeg	**2442417**	1413078	20297	18.067	2514129	1413384	**19507**	18.648
mor1kx	**1315771**	561775	8030	7.708	1330973	562741	**7664**	7.684
ariane	**3037041**	1394232	20739	18.645	3142161	1396919	**19504**	18.736
BP	**4300681**	1974723	27324	35.543	4700736	1986244	**24315**	35.960
Average	**0.950**	**0.997**	1.091	**0.985**	1.000	1.000	**1.000**	1.000

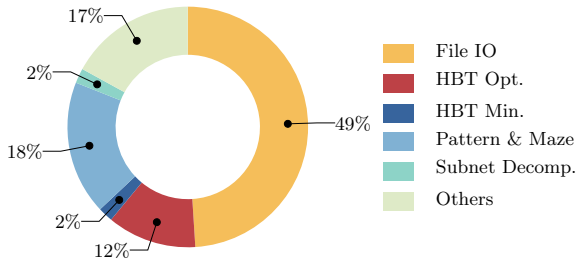


Fig. 11 Runtime breakdown of H3D global router on ariane.

D. Runtime Breakdown

We provide a detailed runtime breakdown of the H3D global router on the benchmark ariane, as shown in Fig. 11. The results demonstrate that our proposed dynamic programming algorithms for HBT number minimization and HBT location optimization are highly efficient, contributing only 2% and 12% of the total runtime, respectively. This low computational overhead underscores the effectiveness of our algorithms in handling the complex challenges of HBT management without significantly impacting the overall runtime. File IO emerges as the primary bottleneck, highlighting the efficiency of our implementation.

V. Conclusion

In this work, we present H3D, the first native 3D global router tailored for face-to-face (F2F) bonded 3D ICs. Unlike existing pseudo-3D routing approaches that rely on 2D IC routing engines with post-routing legalization, H3D directly performs intra-die and inter-die routing, significantly improving solution quality. By constructing a heterogeneous grid graph that captures both routing and hybrid bonding terminal (HBT) resources, and employing our dedicated dynamic programming-based algorithms, H3D efficiently minimizes the number of HBTs and optimizes their placement to achieve minimal wirelength. This optimization leverages the convexity of the L1 distance for computational efficiency. Our experiments on various real-world designs demonstrate the superiority of H3D over state-of-the-art approaches, achieving 12% shorter wirelength, 25% fewer HBTs, and 1.9× speedup, while maintaining scalability and efficiency. The ability to handle the unique challenges posed by F2F bonded 3D ICs, such as the limited availability and critical placement of HBTs, highlights the robustness and practicality of H3D for next-generation 3D IC physical design.

Acknowledgements

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

REFERENCES

[1] E. Beyne, "The 3-D interconnect technology landscape," *IEEE Design & Test*, vol. 33, no. 3, pp. 8–20, 2016.

[2] D. W. Fisher, S. Knickerbocker, D. Smith, R. Katz, J. Garant, J. Lubguban, V. Soler, and N. Robson, "Face to face hybrid wafer bonding for fine pitch applications," in *IEEE Electronic Components and Technology Conference*. IEEE, 2020, pp. 595–600.

[3] T. F. Wu, H. Liu, H. E. Sumbul, L. Yang, D. Baheti, J. Coriell, W. Koven, A. Krishnan, M. Mittal, M. T. Moreira *et al.*, "11.2 a 3D integrated prototype system-on-chip for augmented reality applications using face-to-face wafer bonded 7nm logic at $< 2\mu$m pitch with up to 40% energy reduction at Iso-Area footprint," in *IEEE International Solid-State Circuits Conference (ISSCC)*, vol. 67. IEEE, 2024, pp. 210–212.

[4] T. Lu, C. Serafy, Z. Yang, S. K. Samal, S. K. Lim, and A. Srivastava, "TSV-based 3-D ICs: Design methods and tools," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 36, no. 10, pp. 1593–1619, 2017.

[5] S. S. K. Pentapati and S. K. Lim, "Heterogeneous monolithic 3D ICs: EDA solutions, and power, performance, cost tradeoffs," in *ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2021, pp. 925–930.

[6] M. Pan and C. Chu, "FastRoute: A step to integrate global routing into placement," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2006, pp. 464–471.

[7] ——, "FastRoute 2.0: A high-quality and efficient global router," in *IEEE/ACM Asia and South Pacific Design Automation Conference (ASPDAC)*. IEEE, 2007, pp. 250–255.

[8] K.-R. Dai, W.-H. Liu, and Y.-L. Li, "NCTU-GR: Efficient simulated evolution-based rerouting and congestion-relaxed layer assignment on 3-D global routing," *IEEE Transactions on Very Large Scale Integration Systems (TVLSI)*, vol. 20, no. 3, pp. 459–472, 2011.

[9] J. Liu, C.-W. Pui, F. Wang, and E. F. Young, "CUGR: Detailed-routability-driven 3D global routing with probabilistic resource model," in *ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2020, pp. 1–6.

[10] J. Liu and E. F. Young, "EDGE: Efficient DAG-based global routing engine," in *2023 60th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2023, pp. 1–6.

[11] T.-H. Lee and T.-C. Wang, "Congestion-constrained layer assignment for via minimization in global routing," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 27, no. 9, pp. 1643–1656, 2008.

[12] M. Cho and D. Z. Pan, "BoxRouter: A new global router based on box expansion and progressive ILP," in *ACM/IEEE Design Automation Conference (DAC)*, 2006, pp. 373–378.

[13] T.-H. Wu, A. Davoodi, and J. T. Linderoth, "GRIP: Scalable 3D global routing using integer programming," in *ACM/IEEE Design Automation Conference (DAC)*, 2009, pp. 320–325.

[14] C. Albrecht, "Global routing by new approximation algorithms for multicommodity flow," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 20, no. 5, pp. 622–632, 2001.

[15] 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 (TCAD)*, vol. 39, no. 6, pp. 1151–1164, 2019.

[16] L. Bamberg, A. García-Ortiz, L. Zhu, S. Pentapati, S. K. Lim *et al.*, "Macro-3D: A physical design methodology for face-to-face-stacked heterogeneous 3D ICs," in *IEEE/ACM Proceedings Design, Automation and Test in Europe (DATE)*, 2020, pp. 37–42.

[17] S. Panth, K. Samadi, Y. Du, and S. K. Lim, "Shrunk-2-D: A physical design methodology to build commercial-quality monolithic 3-D ICs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 36, no. 10, pp. 1716–1724, 2017.

[18] S. S. K. Pentapati, K. Chang, V. Gerousis, R. Sengupta, and S. K. Lim, "Pin-3D: A physical synthesis and post-layout optimization flow for heterogeneous monolithic 3D ICs," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2020, pp. 1–9.

[19] Y.-H. Huang, S. Pentapati, A. Agnesina, M. Brunion, and S. K. Lim, "On legalization of die bonding bumps and pads for 3D ICs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 2024.

[20] S. Liu, J. Jiang, Z. He, Z. Wang, Y. Lin, B. Yu, and M. Wong, "Routing-aware legal hybrid bonding terminal assignment for 3D face-to-face stacked ICs," in *ACM International Symposium on Physical Design (ISPD)*, 2024, pp. 75–82.

[21] M.-C. Tsai, T.-C. Wang, and T. Hwang, "Through-silicon via planning in 3-D floorplanning," *IEEE Transactions on Very Large Scale Integration Systems (TVLSI)*, vol. 19, no. 8, pp. 1448–1457, 2010.

[22] D. H. Kim, R. O. Topaloglu, and S. K. Lim, "Block-level 3d ic design with through-silicon-via planning," in *IEEE/ACM Asia and South Pacific Design Automation Conference (ASPDAC)*. IEEE, 2012, pp. 335–340.

[23] C. Chu and Y.-C. Wong, "FLUTE: Fast lookup table based rectilinear steiner minimal tree algorithm for VLSI design," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 27, no. 1, pp. 70–83, 2008.

[24] M. I. Dewan, S.-E. D. Lin, and D. H. Kim, "Construction of all multilayer monolithic rsmts and its application to monolithic 3D IC routing," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 29, no. 1, pp. 1–28, 2023.

[25] D. Sankoff and P. Rousseau, "Locating the vertices of a steiner tree in an arbitrary metric space," *Mathematical Programming*, vol. 9, pp. 240–246, 1975.

[26] A. K. Kiefner, "Minimizing path lengths in rectilinear steiner minimum trees with fixed topology," *Operations Research Letters*, vol. 44, no. 6, pp. 835–838, 2016.

[27] "Rocket Chip," https://github.com/chipsalliance/rocket-chip, accessed: 2024-04-03.

[28] "aes," https://opencores.org/projects/tiny_aes, accessed: 2024-05-07.

[29] "ethmac," https://opencores.org/projects/ethmac, accessed: 2024-05-07.

[30] "jpegencoder," https://opencores.org/projects/mkjpeg, accessed: 2024-05-07.

[31] "mor1kx," https://github.com/openrisc/mor1kx, accessed: 2024-05-07.

[32] "Ariane RISC-V CPU ," https://github.com/openhwgroup/cva6, accessed: 2024-04-08.

[33] "BlackParrot," https://github.com/black-parrot/black-parrot, accessed: 2024-04-10.

[34] Y. Xu, Y. Zhang, and C. Chu, "Fastroute 4.0: Global router with efficient via minimization," in *IEEE/ACM Asia and South Pacific Design Automation Conference (ASPDAC)*. IEEE, 2009, pp. 576–581.

[35] OpenCores, "OpenCores," https://opencores.org/, 2024, accessed: 2024-03-04.

[36] C. Wolf, "Yosys open synthesis suite," 2016.

[37] T. Ajayi, V. A. Chhabria, M. Fogaça, S. Hashemi, A. Hosny, A. B. Kahng, M. Kim, J. Lee, U. Mallappa, M. Neseem *et al.*, "Toward an open-source digital flow: First learnings from the openroad project," in *ACM/IEEE Design Automation Conference (DAC)*, 2019, pp. 1–4.

[38] "NanGate 45nm," https://si2.org/open-cell-library/.

[39] 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)*, 2023.

[40] 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.

[41] A. B. Kahng, L. Wang, and B. Xu, "Tritonroute: The open-source detailed router," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 40, no. 3, pp. 547–559, 2020.