Mixed-structure Double-sided Redistribution Layer Routing for Glass Interposer-based 5.5D ICs

Haoyang Xu[1], Zhen Zhuang[1*], Leilei Jin[1], Zheng Yang[2], Chen Wu[3],
Lei He[3], Sung Kyu Lim[4], Tsung-Yi Ho[1]
[1] The Chinese University of Hong Kong [2] Georgia Institute of Technology
[3] Eastern Institute of Technology [4] University of Southern California [*] Corresponding author

ABSTRACT

Glass interposers are promising for 5.5D integration by facilitating the embedding of chiplets within interposers while offering superior mechanical and electrical properties compared to their silicon counterparts. Routing in such dense environments is exceptionally challenging. This challenge is further compounded by the presence of double-sided interconnections and the unique design rules inherent to glass-based technologies. Existing routers cannot handle the specific constraints. This paper presents a novel mixed-structure routing algorithm for 5.5D redistribution layers, which strategically processes the upper and lower RDLs using distinct routing strategies. Double-sided global routing is initially performed using a SAT-based routing resource allocation method. Subsequently, a pixelization-based detailed routing phase is executed, which is tailored to address the specific requirements of each layer, to yield the final layout. Experimental results demonstrate that our method achieves 10% routability improvement to the state-of-the-art 2.5D RDL router extended for 5.5D integration, while reducing total wirelength by 37% compared to the original 2.5D approach.

1 INTRODUCTION

With the growing demand for high-performance computing and heterogeneous integration, advanced packaging technologies have become essential to extending Moore's Law beyond the limits of monolithic integration. Chiplet integration has emerged as a promising approach to improve the yield of highly complex packages [1]. Among various integration technologies, 2.5D interposers and 3D stacking have attracted significant attention for enabling the integration of multiple heterogeneous chiplets with diverse functionalities and process nodes. 2.5D integration employs a silicon interposer to provide high-density interconnects between laterally placed chiplets. As shown in Fig. 1, a silicon interposer offers fine-pitch redistribution layers (RDLs) that route signal and power nets between chiplets. 3D integration extends this concept by stacking chiplets vertically using through-silicon vias (TSVs) and micro-bumps. Despite their widespread adoption, silicon interposers face several limitations, including high fabrication costs, limited scalability, and notable parasitic effects caused by TSVs.

Recently, glass interposers have emerged as a promising alternative, offering excellent electrical properties, low dielectric loss, and the potential for large-panel processing at reduced cost [2]. Unlike conventional solutions, glass is the material that allows chiplets to be embedded directly within the interposer, leading to a more compact and highly integrated system architecture than a 2.5D interposer, as illustrated in Fig. 1. We refer to this glass interposer-based integration as "5.5D" stacking, denoting the advantage of combining 2.5D interposers and 3D stacking. Fig. 2 shows a schematic of glass interposer-based 5.5D stacking technology, where multiple RDLs are utilized for signal redistribution. Benefiting from the high-dimensional stability of glass, double-sided RDL routing is adopted to support high-density interconnects. The upper RDLs and lower RDLs are connected by through-glass vias (TGVs). Chiplets such as high-bandwidth memories can be embedded within the glass interposer, while the lower side RDLs are connected to the package substrate through C4 bumps.

Despite these advantages, the double-sided RDL routing in glass interposers introduces unprecedented design challenges. The coexistence of embedded chiplets, TGVs, and high-density interconnects creates a complex 5.5D routing environment that conventional 2.5D routers cannot adequately handle. The adoption of 5.5D stacking for integrating chiplets of different technology nodes into a compact package further complicates the routing problem, necessitating an efficient routing scheme that minimizes wirelength while implementing complex routing paths.

Several methods have been proposed to address 2.5D RDL routing challenges. [3] first introduced a method for multi-layer multi-chip

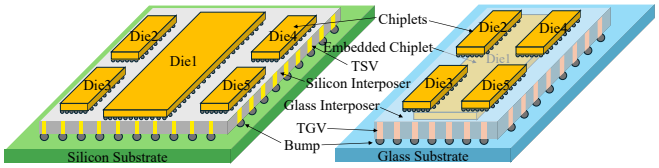

Figure 1: The conventional 2.5D packaging vs. the glass-interposer-based 5.5D packaging. A denser design with embedded chiplets can be implemented with the advantages of a glass interposer.

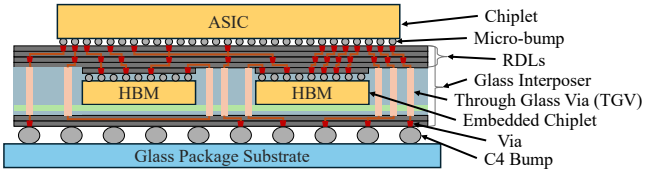

Figure 2: Cross-sectional view of the glass interposer-based 5.5D packaging, enabling double-sided routing structure for heterogeneous integration.

DAC '26, July 26–29, 2026, Long Beach, CA, USA

ACM ISBN 979-8-4007-2254-7/2026/07…$15.00
https://doi.org/10.1145/3770743.3804040

routing. Wen [4] explored the use of irregular vias, while [5] developed a simultaneous routing algorithm for chip-to-board and chip-to-chip interconnections. [6] proposed an obstacle-aware RDL router, and [7] introduced the first any-angle RDL router. [8] efficiently solved the inter-chip RDL routing problem using a hierarchical partitioning-based strategy. More recently, [9] presented an RDL routing algorithm that iteratively optimizes the substrate layout using cross-boundary timing context from both chiplets and the package. However, these methods face limitations when applied to 5.5D packaging. On one hand, they struggle to handle the extreme density and routing flexibility of 5.5D structures. On the other hand, existing approaches cannot consider the impact of TGVs and other glass-specific design rules.

To address the critical challenge of RDL routing in 5.5D integration and overcome the shortcomings of existing methods, this paper presents an efficient and systematic routing solution. Our key contributions are as follows:

- We propose a mixed-structure double-sided RDL routing flow for glass interposer-based 5.5D packaging to maximize the routability and minimize the wirelength.
- We develop double-sided global routing using a SAT-based resource allocation method, which employs contrasting structures for the upper and lower RDLs to account for the former's overlapping dense I/O regions and the latter's sparse routing characteristics.
- We present a highly versatile, detailed routing method based on pixelization and binary indexed trees to reduce detours.
- Experimental results demonstrate that our method achieves 10% routability improvement to the state-of-the-art 2.5D RDL router extended for 5.5D integration, while reducing total wirelength by 37% compared to the original 2.5D approach.

2 PRELIMINARIES

In this section, we first describe the notations and design rules used in the glass interposer-based RDL routing. Afterward, the RDL routing problem in 5.5D packaging is formulated.

2.1 Notations and Design Rules

Different from 2.5D packaging that integrates all chiplets above the RDL, chiplets can be embedded in the glass interposer, allowing not only 2.5D horizontal integration but also 3D vertical stacking. Moreover, extra RDL is deployed for a more compact design. In this work, both 5.5D stacking and double-sided routing are considered during RDL routing. The terminologies used are listed below:

- $P = \{p_i \mid 1 \leqslant i \leqslant |P|\}$ is the set of I/O pads.
- $B = \{b_i \mid 1 \leqslant i \leqslant |B|\}$ is the set of C4 bumps.
- $L_t = \{l_i^t \mid 1 \leqslant i \leqslant |L_t|\}$ is the set of routing layers above the glass interposer. We note the upper RDLs.
- $L_b = \{l_i^b \mid 1 \leqslant i \leqslant |L_b|\}$ is the set of routing layers beneath the glass interposer. We note the lower RDLs.
- $N_p = \{n_i^p \mid 1 \leqslant i \leqslant |N_p|\}$ is the set of routing nets representing I/O pads to I/O pads connections. Both endpoints can be either from chiplets or embedded chiplets.
- $N_f = \{n_i^f \mid 1 \leqslant i \leqslant |N_f|\}$ is the set of routing nets representing I/O pads to C4 bumps connections.
- $G = \{g_i \mid 1 \leqslant i \leqslant |G|\}$ is the set of TGVs punching through the glass interposer to connect double-sided RDLs.

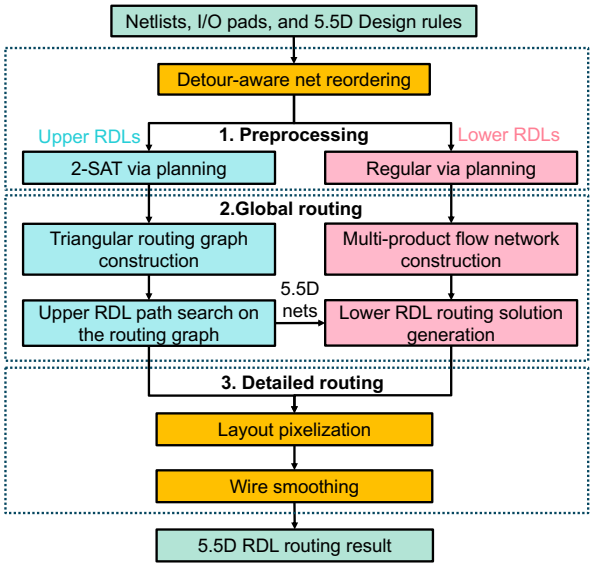

Figure 3: The overall flow of the proposed algorithm.

- w_e/w_v is the width of wires/vias.
- w_g is the width of TGVs. Each TGV has a comparable size to TSV, and is much larger than via.

Besides 5.5D stacking and double-sided routing, the design rules that need to be considered in RDL routing are as follows:

- *X-architecture wires*: Wire segments can only be routed in ±0°, ±90°, 45°, and 135° directions. Other angles are forbidden.
- *Non-crossing rule*: Any two routing wire segments cannot be implemented across each other in an RDL.
- *Minimum spacing rule*: Any two components need to satisfy the specified minimum spacing, including wires, vias, and TGVs.
- *Connection-angle rule*: Only a 90°/135° turn is allowed if two wire segments are connected.
- *Glass interposer rule*: TGVs cannot penetrate embedded chiplets.

2.2 Problem Formulation

Based on the above analysis, **the double-sided RDL routing problem in 5.5D packaging** can be formulated as follows:

Given a set N of routing nets, a set P of I/O Pads on different layers, two sets L_t and L_b of RDL layers on the different sides of glass interposer, bump array B, and design rules, establish the connections of all the nets with vias and TGVs such that the routability is maximized, the total wirelength is minimized, and no design rules are violated.

3 PROPOSED ALGORITHM

In this section, we present our 5.5D RDL routing algorithm. Fig. 3 shows the overall flow of the proposed algorithm. There are three major stages: 1) Preprocessing, 2) Global routing, and 3) Detailed routing. To address the challenges in double-sided designs, we employ a mixed-structure routing strategy to perform global routing separately in the upper RDLs and the lower RDLs. The technical details are discussed below.

3.1 SAT-Based Routing Resource Allocation

Detour-aware net ordering To mitigate the adverse effects of routing congestion and detours caused by net crossings, we design

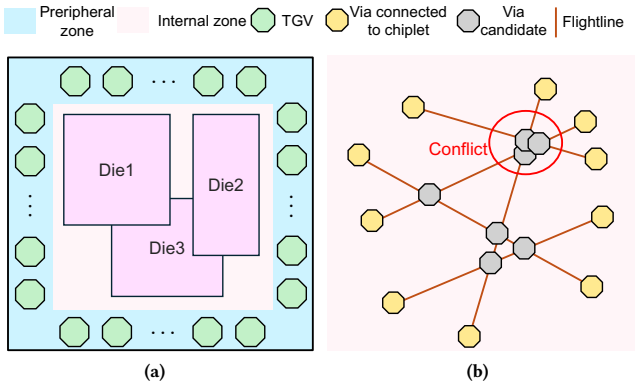

Figure 4: (a) Abstract layout partitioning with TGVs inserted on the peripheral zone, and (b) an example of via planning, the conflict is resolved by a 2-SAT formulation.

a net reordering strategy. The key insight is that when the flight lines of two nets intersect, the later-routed net l_2 needs to detour around the earlier-routed net l_1. The additional detour length for l_2 is approximately equal to the distance from the intersection point to the closer endpoint of l_1.

Formally, let $\text{dist}(p, l_1)$ denote the minimum Euclidean distance from point p to either endpoint of net l_1. For an intersection point q between the flight lines of nets l_1 and l_2, the detour cost for l_2 is estimated as the minimal Euclidean distance of q to any endpoint of l_1, noted as $\text{dist}(q, l_1)$.

To minimize the total detouring effect, we employ an iterative greedy algorithm. At each step i, we select from the set of unrouted nets U the net n^* that minimizes the total potential detour cost it would impose on all other nets in U. For each candidate net $n \in U$,

$$C(n) = \sum_{m \in U \setminus \{n\}} \text{detour}(m, n) = \sum_{m \in U \setminus \{n\}} \text{dist}(q_{n,m},\ n), \qquad (1)$$

where $q_{n,m}$ is the intersection point between the flight lines of n and m. The selected net n^* is the best solution to minimize $C(n)$.

This net n^* is assigned the i-th routing order and removed from U. The process repeats until all nets are ordered, producing a complete net sequence that reduces overall routing detours and improves wirelength efficiency.

2-SAT via planning Due to the special design rules of TGV, we treat TGV planning as a distinct problem. Following conventional layout strategies [10], we preferentially place TGVs along the periphery of the routing region since TGVs would become serious obstacles if placed in a highly congested routing area.

In our approach, we conceptually partition the layout into two abstract zones as shown in Fig. 4(a):

- Peripheral zone: Reserved for TGVs, which provide routing resources for nets requiring long detours and for connections to bumps for better routability.
- Internal zone: A high-density routing area corresponding to the projection of chiplets. Standard vias are inserted in the internal zone to maximize connectivity and shorten detours.

We estimate the required number of TGVs based on the number of free-assignment nets and overall routing density. These TGVs are uniformly distributed along the layout periphery. The structure is

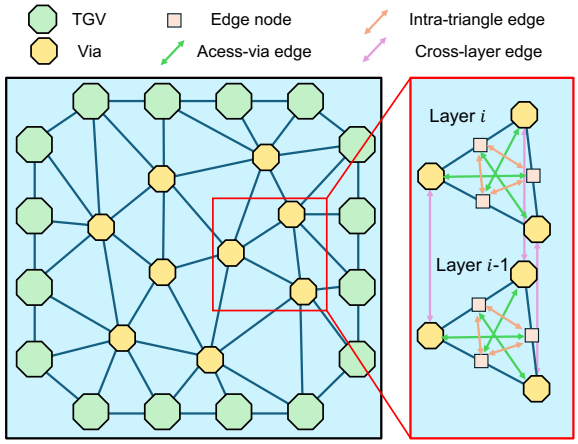

Figure 5: Illustration of routing graph construction based on DT result, both horizontal and vertical connections are included.

illustrated in Fig. 4(a). For the internal zone, via insertion is guided by net interaction analysis. All net endpoints automatically receive vias for die-to-RDL signal connections. Additionally, we consider intersection points between flightlines of nets since the segment of the flightline is the shortest path from the start to the end. Let S be the set of all pairwise intersection points of flightlines, represented via candidates in our method. For each point $p \in S$, we define a binary variable $x_p \in \{0, 1\}$ indicating whether to insert a via at p.

We formulate it as a constraint optimization problem. Two points $p_i, p_j \in S$ cannot both be selected if their corresponding vias would violate design rules (e.g., minimum spacing). This conflict is represented as a clause $(\neg x_i \vee \neg x_j)$ in a 2-SAT model. The objective is to maximize the number of selected points:

$$\text{Maximize} \sum_{p \in S} x_p, \quad s.t. \bigwedge_{(i,j) \in T} (\neg x_i \vee \neg x_j), \qquad (2)$$

where T is the set of conflicting pairs. An optimal solution to this 2-SAT instance yields a maximal via assignment that enhances connectivity while respecting design rules. Take Fig. 4(b) as a via planning example of 5 nets. For the planning of the lower RDL, we insert the vias directly on the center of each bump since the bump array is very regular. Additionally, regular vias are more efficient for finding the optimal lower RDL routing solutions.

3.2 Doubled-sided Global Routing

Triangulation-based upper RDL global routing Based on the planned via and TGV positions obtained from the previous stage, we construct a three-dimensional routing graph to facilitate efficient interconnect routing. The construction begins with projecting all via and TGV locations onto the 2D plane and performing Delaunay Triangulation (DT) to partition the routing area into basic triangular elements. This is a common partitioning strategy, which perfectly meets the requirement of the highly congested upper RDL inter-chiplet routing. Moreover, DT offers significant advantages: the maximized minimum angle property avoids skinny triangles, ensuring better numerical stability and more uniform routing resource distribution, while the empty circle property provides natural obstacle avoidance and congestion-aware partitioning.

After triangulation, we construct the routing graph based on the DT result. We define the routing graph vertices to include all via/TGV locations, and the edge nodes representing each triangle edge. Each via and TGV vertex is assigned a capacity of 1, while the capacity of edge vertices is calculated based on available routing space. Specifically, the capacity of an edge vertex is computed as:

$$Cap_e = \max_{\theta \in \{0°,45°,90°,135°\}} \left\lfloor \frac{L_e - W_{\text{via}}}{\sqrt{2}\delta P_\theta} \right\rfloor, \tag{3}$$

where L_e represents the edge length, W_{via} equals w_v or w_g, and P_θ is the pitch requirement for routing direction θ. The coefficient $\sqrt{2}$ is used to avoid the potential conflicts introduced in detailed routing.

The graph edges are then established through four types of connections, as shown in Fig. 5: 1) Cross-layer edge: vertical via connections create zero-weight edges between corresponding vias across different metal layers, representing vertical interconnect resources. 2) Intra-triangle edge: connections from edges between the vertices of the triangle edges, with weights equal to their Euclidean distances. 3) Access-via edge: connections between vias or TGVs and the corresponding non-adjacent edge node. 4) TGV connections create edges between all TGV pairs, enabling potential lower-layer routing. To discourage unnecessary use of these valuable resources, TGV interconnection edges are assigned infinite weight, ensuring they are only utilized when absolutely necessary.

Based on this comprehensive graph structure, we employ the conflict detection method proposed by [7] to identify and resolve routing conflicts when searching for routing paths. The detection is implemented by a double-linked list data structure to find the conflict topologically. Finally, we use A* search to find the routing guide for all nets. Specifically, for pad-to-bump nets, our search terminates when it reaches a TGV. The lower part of the net is routed following the lower RDL routing structure.

Multi-product flow-based lower RDL global routing It is extremely difficult to route the inter-chiplet nets only relying on the upper RDLs since they are highly congested and hard to achieve perfect routability. Therefore, we must put down the 5.5D nets that failed in the upper global routing. Based on the routing solution obtained from the upper RDL, we acquire the necessary information for the lower RDL global routing. The nets to be routed in the lower layer can be categorized into two types: 1) Free Nets: Connections from a TGV to an arbitrary bump (pad-to-bump nets). 2) Fixed Nets: Connections from one TGV to another TGV, which correspond to nets in the upper RDL that require extra lower RDL routing resources. A flow-based algorithm is the traditional but effective solution to pad-to-board connections. However, it can be directly applied to our problem. We formulate the lower RDL global routing problem as a multi-commodity flow problem, which is solved by graph construction, multi-commodity flow formulation, and conflict resolution.

First, we adopt the network flow model based on [11]. The bump array is represented by a grid of nodes and edges, where each node corresponds to a bump position, and edges represent routing channels between adjacent vias. The orthogonal capacity (*O-cap*) and diagonal capacity (*D-cap*) constraints are captured by the design rules. The detailed edge capacity is shown in Fig. 6. In our extension

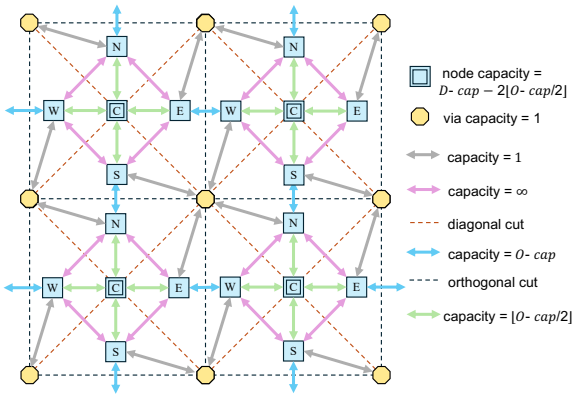

Figure 6: Our network flow model within vias based on [11]. We introduced the extra multi-layer flow constraint.

to multi-layer scenarios, we enhance the model as follows. Bidirectional edges are added between via nodes and their corresponding connected abstract nodes, with capacities set according to 1. For TGV nodes, we connect each TGV to its nearest abstract node via a bidirectional edge with capacity 1. This connection allows TGVs to access the lower RDL routing resources.

To distinguish between free nets and fixed nets, we employ a multi-commodity flow formulation. Let $G = (V, E)$ denote the routing graph, where V is the set of nodes and E is the set of edges. Let K be the set of nets to be routed. Each net $k \in K$ is associated with a source node s_k and a target node t_k.

For free nets, s_k is a TGV node and t_k is a bump node. For fixed nets, both s_k and t_k are TGV nodes. We define a binary variable f_e^k for each edge $e \in E$ and each net $k \in K$, which indicates whether net k uses edge e. The objective is to minimize the total wirelength while satisfying the capacity constraints.

The formulation is as follows:

$$\text{Minimize} \quad \sum_{k \in K} \sum_{e \in E} l_e \cdot f_e^k \tag{4}$$

$$\text{s. t.} \quad \sum_{e \in \delta^+(v)} f_e^k - \sum_{e \in \delta^-(v)} f_e^k = \begin{cases} 1, & \text{if } v = s_k \\ -1, & \text{if } v = t_k \\ 0, & \text{otherwise} \end{cases} \qquad \forall v \in V, k \in K, \tag{5}$$

$$\sum_{k \in K} f_e^k \le c_e \quad \forall e \in E, \quad f_e^k \in \{0,1\} \quad \forall e \in E, k \in K. \tag{6}$$

Here l_e is the length of edge e, c_e is the capacity of edge e, and $\delta^+(v)$ and $\delta^-(v)$ denote the outgoing and incoming edges of node v, respectively. Because all the constraints and objectives are linear. This integer linear programming (ILP) problem can be relaxed to an LP problem. The Equation (6) can be relaxed as:

$$\sum_{k \in K} f_e^k \le c_e \quad \forall e \in E, \quad f_e^k \in [0,1] \quad \forall e \in E, k \in K. \tag{7}$$

3.3 Pixelization-based Detailed Routing

After completing double-sided global routing, employ a pixelation-based detailed routing method for both sides. Moreover, our detailed routing algorithm can reduce detours.

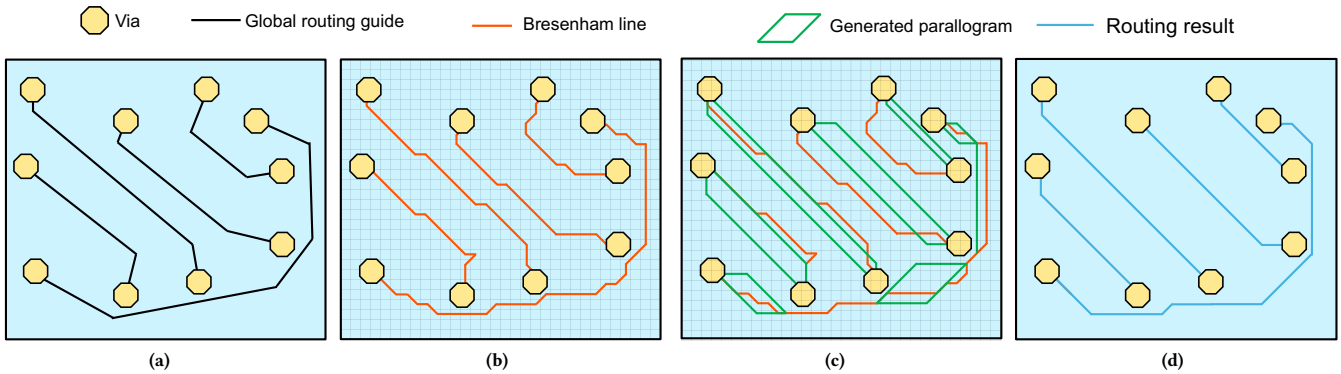

Figure 7: Illustration of pixelization-based detailed routing. (a) The global routing result of 5 nets, (b) The generated 5 Bresenham lines of (a), (c) Parallelograms used in the wire smoothing algorithm, and (d) final detailed routing result of (a).

Algorithm 1 : Bresenham lines smoothing

Input : Pixelized wires P for all nets N and all layers L, obstacle maps O_l for all layers.
Output : Smoothed wire paths S for all nets and all layers.

Initialize $S \leftarrow \emptyset$ for all k, l;
Initialize $O_1, O_2, ..., O_L$ according to Bresenham lines P;
Build BIT from obstacle maps $O_1, O_2, ..., O_L$;
for *each layer* $l \in \{1, 2, ..., L\}$ **do**
  for *each net* $k \in \{1, 2, ..., N\}$ **do**
    Let $P \leftarrow P_l^k$, $n \leftarrow |P|$, $i \leftarrow 1$;
    while $i \leq n$ **do**
      $j_{max} \leftarrow i$; // Initialize the candidate parallelogram
      for $j \leftarrow i + 1$ ***to*** n **do**
        if *ParallelogramArea*(p_i, p_j) *is obstacle-free according to BIT* **then**
          $j_{max} \leftarrow j$; // Valid parallelogram found
      Add segment $(p_i, p_{j_{max}})$ to S_l^k;
      Update O_l and BIT by $(p_i, p_{j_{max}})$;
      $i \leftarrow j_{max} + 1$; // Search for next parallelogram
return $\{S_l^k\}$ *for all* k, l;

Layout pixelization To ensure design rule compliance, we perform pixelization on each layer using a pixel size of $\sqrt{2}(w_e + \delta)$. This guarantees that wires placed along pixel centers will satisfy width and spacing constraints. The pixelization process consists of two steps: 1) we snap all endpoints of wire segments from the global routing solution to the centers of their nearest pixels. 2) We connect these snapped points using Bresenham's line algorithm [12], which efficiently determines the pixels that approximate a straight line between two points using only integer arithmetic. The algorithm works by incrementally selecting pixels that minimize the error between the ideal line and the rasterized path. Fig. 7(b) shows the Bresenham lines corresponding to the 5 nets of the global routing result in Fig. 7(a). After pixelization, each wire segment is represented as a polyline connecting pixel centers. While this naturally ensures compliance with X-architecture constraints, the resulting paths suffer from excessive bends and potentially violate non-acute angle constraints shown in Fig. 7(c). Therefore, further wire smoothing is required for better manufacturability.

Table 1: Details of the benchmarks used in the experiments*.

Benchmarks (Area, #Chips, $\lvert P\rvert$, $\lvert N_p\rvert$, $\lvert N_f\rvert$, $\lvert L_t\rvert$, $\lvert L_b\rvert$, w_e)	
dense1-glass (3×5, 2, 44, 22, 0, 2, 1, 4)	dense2-glass (7.5×5, 3, 92, 46, 0, 2, 1, 4)
dense3-glass (5×3, 5, 158, 79, 0, 2, 1, 2)	dense4-glass (6×5, 6, 222, 111, 0, 3, 1, 2)
dense5-glass (9×7, 9, 522, 261, 0, 3, 2, 2)	pkg1-glass (4×4, 3, 153, 10, 64, 2, 1, 2)
pkg2-glass (5×5, 4, 266, 28, 69, 2, 1, 2)	pkg3-glass (6×6, 7, 403, 54, 77, 2, 1, 2)
pkg4-glass (8×6, 12, 588, 90, 93, 2, 1, 2)	pkg5-glass (12×12, 20, 1508, 246, 120, 3, 2, 2)

*The unit of area is $10^3\mu m \times 10^3\mu m$; The unit of $w_v/w_e/\delta$ is μm.

Wire smoothing using BITs We employ a greedy optimization algorithm to smooth the pixelized wires while reducing wirelength. The key idea is to replace jagged segments with longer straight segments where possible. For each layer l and each net k, let $P_l^k = \{p_1, p_2, ..., p_n\}$ be its sequence of pixel centers on layer l. The algorithm processes all nets across all layers to generate smoothed paths. The smoothing process for each net-layer combination attempts to find the farthest reachable point p_j $(j > i)$ from the current point p_i such that the parallelogram path between p_i and p_j is obstacle-free, as shown in Fig. 7(c). This parallelogram is defined by the Manhattan-like bounding box between the two points when considering 45° routing constraints. To efficiently check for obstacles across all layers, we use a binary indexed tree (BIT) [13] to maintain and query the occupancy status of pixels in each layer. The BIT is a data structure that supports single-point modifications and range queries for summation in Logarithmic complexity, which perfectly fits our fast conflict-checking requirement. The overall complexity is $O((L_t + L_b)WH \log(W + H))$, where W and H are the dimensions of the pixel grid, L is the number of layers, and N is the number of nets. We can see that the wirelength in Fig. 7(c) is shorter than in Fig.7(d), because our smoothing method not only legalizes the wire segments, but reduces detours as well. The complete smoothing algorithm is shown in Algorithm 1.

Table 2: Comparisons with 2.5D routing results, HP-ext [8] regarding the percentage of routability, the total wirelength, and the CPU time. The best performance of each metric is highlighted in blue.

2.5 D				5.5 D						
Benchmark	Routability (%)	Wirelength (μm)	Runtime (s)	Benchmark	Routability (%)		Total wirelength (μm)		CPU time (s)	
	HP				HP-ext	Ours	HP-ext	Ours(Δ2.5D, ΔHP-ext)	HP-ext	Ours
dense1	100	86695	1.02	dense1-glass	100	**100**	65526	**64424 (-21.9%, -1.7%)**	1.24	**0.96**
dense2	100	252134	6.12	dense2-glass	84.78	**100**	>159340	**197009 (-21.9%, -)**	8.57	**3.83**
dense3	100	307721	35.13	dense3-glass	72.15	**100**	>184660	**180436 (-41.3%, -)**	55.01	**10.57**
dense4	100	475437	88.67	dense4-glass	88.88	**100**	>342497	**339553 (-28.5%, -)**	88.67	**43.41**
dense5	100	1616681	455.34	dense5-glass	87.90	**100**	>1233889	**1221728 (-24.4%, -)**	279.12	**229.72**
Average	1.00	1.37	1.87	-	0.88	**1.00**	-	**1.00**	1.49	**1.00**

Table 3: Experimental results on mixed-type benchmarks compared with HP-ext [8] on 5 benchmarks.

Benchmark	Routability (%)		Total wirelength (μm)		CPU time (s)	
	HP-ext	Ours	HP-ext	Ours	HP-ext	Ours
pkg1-glass	89.18	**100**	>88707	**93960**	10.75	**1.54**
pkg2-glass	98.96	**100**	>182336	**160092**	33.03	**3.45**
pkg3-glass	74.04	**100**	>234824	**330426**	47.82	**8.24**
pkg4-glass	95.18	**100**	>480374	**450993**	130.41	**17.40**
pkg5-glass	97.98	**100**	>1385678	**1335439**	363.19	**161.78**
Average	0.91	**1.00**	-	**1.00**	3.04	**1.00**

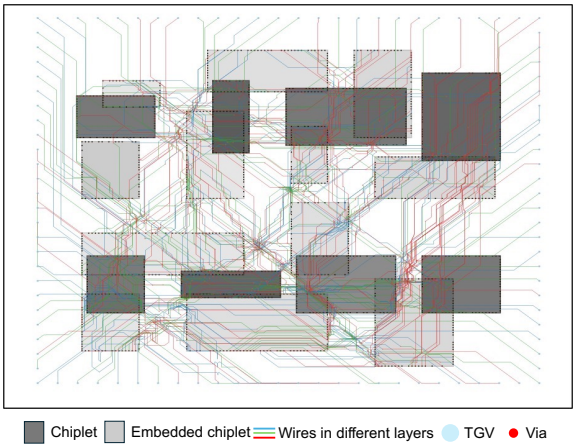

Figure 8: Upper RDL routing solution of circuit pkg5-glass.

4 EXPERIMENTAL RESULTS

The proposed algorithm was implemented in C++ with the Boost library [14] and evaluated on a workstation equipped with a 2.9-GHz CPU and 256 GB of memory. The Gurobi Optimizer [15] was employed to solve the multi-product MCMF formulation. We validated our approach using 10 benchmark circuits derived from industrial designs [5] by replacing 2.5D chiplets with 5.5D stacking, where chiplets are evenly embedded into the glass interposer. Table 1 summarizes the circuit characteristics, where #Chiplets, $|P|$, $|N_p|$, $|N_f|$, $|L_t|$, $|L_b|$, and w_e denote the number of chiplets, I/O pads, inter-chiplet nets, chiplet-to-bump nets, number of upper RDLs, number of lower RDLs, and wire width. The width of TGVs and vias was set to 50 μm and 20 μm, with a minimum spacing of 2 μm [16].

To assess the effectiveness of our method, we compared it with the SOTA RDL routing technique presented in [8]. Two configurations of [8] were considered: the 2.5D setup (denoted as HP), and an extended version (denoted as HP-ext) that incorporates an additional A* search to reroute failed nets resulting from additional constraints introduced by the glass interposer. Table 2 summarizes the results on benchmarks containing only inter-chiplet nets. The proposed algorithm achieved 100% routability across all testcases, outperforming HP-ext, which attained an average of 88% routability. In terms of wirelength, our approach reduced the total wirelength by an average of **37%** compared to the 2.5D configuration, highlighting the benefits of 5.5D stacking. Notably, the proposed method also yielded significantly shorter wirelength, even in cases where HP-ext failed to complete all routes. Regarding runtime, the proposed method was **1.87**× and **1.49**× faster than the 2.5D and HP-ext versions, respectively. Table 3 further reports the results on mixed-type benchmarks comprising both chiplet-to-chiplet and chiplet-to-bump nets. Our algorithm consistently achieved 100% routability, whereas HP-ext attained only 91% on average, with a **3**× runtime. Fig. 8 illustrates the upper RDL routing result for the highly dense circuit pkg5-glass, where all specified nets are successfully routed within the given RDL layers, demonstrating the practical viability of our approach.

We attribute the superior performance of our algorithm to the following factors: 1) Early-stage routing resource planning, including net reordering and via assignment, effectively allocates routing resources and minimizes detours. 2) The proposed TGV insertion strategy successfully alleviates routing congestion, thereby enhancing routability. 3) The detailed routing method optimizes wirelength across the entire layout, rather than being restricted to local tile-based pathfinding as in [8]. Moreover, the algorithm runtime is largely reduced due to the adoption of binary indexed trees in detailed routing.

5 CONCLUSIONS

To the best of our knowledge, this is the first paper in the literature to consider the 5.5D redistribution layer routing problem in glass interposer-based packaging. We proposed a three-stage algorithm. Firstly, preprocessing is done to allocate routing resources. Secondly, we find the routing guide by global routing. Finally, the routing result is generated via detailed routing. Experimental results have shown the potential of 5.5D ICs and efficiency of our routing algorithm to completely solve the 5.5D RDL routing problem.

Acknowledgments

This research was partially conducted by ACCESS – AI Chip Center for Emerging Smart Systems, supported by the InnoHK initiative of the Innovation and Technology Commission of the Hong Kong Special Administrative Region Government. This work is jointly supported by the Research Grants Council of Hong Kong SAR No. CUHK14211324 and No. T46-415/25-R.

REFERENCES

[1] L. T. Su, S. Naffziger, and M. Papermaster, "Multi-chip technologies to unleash computing performance gains over the next decade," in *Proceedings of 2017 IEEE International Electron Devices Meeting (IEDM)*, 2017, pp. 1.1.1–1.1.8.

[2] C.-H. Chien, H. Yu, C.-K. Lee, Y.-M. Lin, R.-S. Cheng, C.-J. Zhan, P.-S. Chen, C.-C. Liu, C.-K. Hsu, H.-H. Chang, H.-C. Fu, Y.-C. Lee, W.-W. Shen, C.-T. Ko, W.-C. Lo, and Y. J. Lu, "Performance and process characteristic of glass interposer with through-glass-via(TGV)," in *Proceedinsg of 2013 IEEE International 3D Systems Integration Conference (3DIC)*, 2013, pp. 1–7.

[3] B.-Q. Lin, T.-C. Lin, and Y.-W. Chang, "Redistribution layer routing for integrated fan-out wafer-level chip-scale packages," in *Proceedinsg of 2016 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2016, pp. 1–8.

[4] H.-T. Wen, Y.-J. Cai, Y. Hsu, and Y.-W. Chang, "Via-based Redistribution Layer Routing for InFO Packages with Irregular Pad Structures," in *Proceedings of ACM/IEEE Design Automation Conference (DAC)*, 2020, pp. 1–6.

[5] Y.-J. Cai, Y. Hsu, and Y.-W. Chang, "Simultaneous Pre- and Free-assignment Routing for Multiple Redistribution Layers with Irregular Vias," in *Proceedings of 2021 58th ACM/IEEE Design Automation Conference (DAC)*, 2021, pp. 1147–1152.

[6] Y.-T. Chen and Y.-W. Chang, "Obstacle-Avoiding Multiple Redistribution Layer Routing with Irregular Structures," in *Proceedings of 2022 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, 2022, pp. 1–6.

[7] M.-H. Chung, J.-W. Chuang, and Y.-W. Chang, "Any-Angle Routing for Redistribution Layers in 2.5D IC Packages," in *Proceedings of 2023 60th ACM/IEEE Design Automation Conference (DAC)*, 2023, pp. 1–6.

[8] H. Xu, X. Huang, Z. Zhuang, Z. Yu, B. Guo, K.-Y. Chao, B. Yu, T.-Y. Ho, and M. D. F. Wong, "Hierarchical Partitioning-Based Interchip Redistribution Layer Routing for Fan-Out Wafer-Level Packaging," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 44, no. 11, pp. 4367–4380, 2025.

[9] Z. Zhuang, W. Hung, M. A. Kabir, Y. Peng, and T.-Y. Ho, "Adaptive Redistribution Layer Routing for Chiplet-Package Co-Design in 2.5D System," *ACM Transaction on Dessign Automation and Electronic Systems (TDAES)*, 2025.

[10] P. Vanna-Iampikul, L. Zhu, S. Erdogan, M. Kathaperumal, R. Agarwal, R. Gupta, K. Rinebold, and S. K. Lim, "Glass Interposer Integration of Logic and Memory Chiplets: PPA and Power/Signal Integrity Benefits," in *Proceedings of ACM/IEEE Design Automation Conference (DAC)*, 2025, p. 1–6.

[11] T. Yan and M. D. F. Wong, "Correctly Modeling the Diagonal Capacity in Escape Routing," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (ICCAD)*, vol. 31, no. 2, pp. 285–293, 2012.

[12] J. Bresenham, "A linear algorithm for incremental digital display of circular arcs," *Communication of ACM*, vol. 20, no. 2, p. 100–106, 1977.

[13] B. Y. Ryabko, "A fast on-line adaptive code," *IEEE Transaction on Information Theory*, vol. 38, pp. 1400–1404, 1992.

[14] "The boost c++ libraries." [Online]. Available: https://www.boost.org/

[15] Gurobi Optimization, LLC, "Gurobi Optimizer Reference Manual," 2024. [Online]. Available: https://www.gurobi.com

[16] H. Park and M. S. Bakir, "Ultra-thin chiplet embedding in glass-core package redistribution layers," in *2025 IEEE 75th Electronic Components and Technology Conference (ECTC)*, 2025, pp. 2224–2229.