

LiDAR 2.0: Hierarchical Curvy Waveguide Detailed Routing for Large-Scale Photonic Integrated Circuits

Hongjian Zhou, Haoyu Yang, *Member, IEEE*, Ziang Yin, Nicholas Gangi, Zhaoran (Rena) Huang, *Senior Member, IEEE*, Haoxing Ren, *Fellow, IEEE*, Joaquin Matres, Jiaqi Gu, *Member, IEEE*

Abstract— Driven by innovations in photonic computing and interconnects, photonic integrated circuit (PIC) designs advance and grow in complexity. Traditional manual physical design processes have become increasingly cumbersome. Available PIC layout tools are mostly schematic-driven, which has not alleviated the burden of manual waveguide planning and layout drawing. Previous research in PIC automated routing is largely adapted from electronic design, focusing on high-level planning and overlooking photonic-specific constraints such as curvy waveguides, bending, and port alignment. As a result, they fail to scale and cannot generate DRV-free layouts, highlighting the need for dedicated electronic-photonic design automation tools to streamline PIC physical design. In this work, we present **LiDAR**, the first automated PIC detailed router for large-scale designs. It features a grid-based, curvy-aware A* engine with adaptive crossing insertion, congestion-aware net ordering, and insertion-loss optimization. To enable routing in more compact and complex designs, we further extend our router to hierarchical routing as **LiDAR 2.0**. It introduces redundant-bend elimination, crossing space preservation, and routing order refinement for improved conflict resilience. We also develop and open-source a YAML-based PIC intermediate representation and diverse benchmarks, including **TeMPO**, **GWOR**, and **Bennes**, which feature hierarchical structures and high crossing densities. Evaluations across various benchmarks show that **LiDAR 2.0** produces nearly DRV-free layouts, achieving up to 16% lower insertion loss and 7.69× speedup over prior methods on spacious cases, and 9% lower insertion loss with 6.95× speedup over **LiDAR 1.0** on compact cases. Our codes are open-sourced at [link](#).

I. INTRODUCTION

In recent years, photonic integrated circuits have attracted considerable attention due to their advantages in high-speed data transmission and low power consumption. Notable advances include photonic tensor cores (PTCs) [1] developed for optical neural networks (ONNs) [2], as well as photonic network-on-chips (NoCs) [3] designed for high-bandwidth on-chip communication. Driven by these emerging applications, PIC designs are rapidly increasing in complexity. As shown in Fig. 1, the number of photonic components per chip is approaching the scale of 1000 and is expected to double approximately every 2.5 years [4]. This growth highlights the urgent need for advanced electronic-photonic design automation (EPDA) toolchains [5] to streamline layout, boost productivity, and ensure quality.

Traditionally, the physical design of PICs follows a schematic-driven approach [6], where components are placed

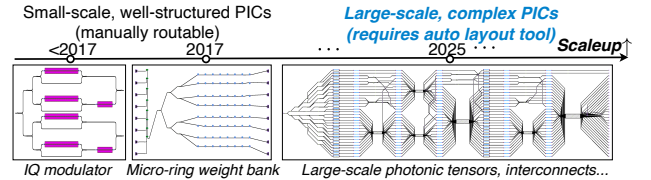


Fig. 1: Modern PIC scale and complexity require EPDA.

and connected based on the circuit topology and signal path defined in the schematic. This method seeks to minimize waveguide crossings, detours, and bends to reduce insertion loss and enhance signal integrity. In certain structured designs, routing can even be **manually managed**, especially when the circuit features a *well-organized, no-crossing topology*, as seen in crossbar arrays [7], triangular or rectangular meshes [8], or binary tree structures [9]. When such designs are *optimally placed with ample spacing and well-aligned ports*, devices can be directly abutted or connected via simple straight waveguides, resembling the standard cell-based layout methodology commonly used in SRAM array design.

However, as PIC designs grow in complexity, significant routing challenges emerge, making automated routing increasingly essential. These challenges typically arise when: ❶ the **circuit scale surpasses the limits of manual design**, involving hundreds or thousands of components and nets; ❷ the circuit features a **complex topology or sub-optimal placement**, resulting in port misalignments, excessive crossings, and severe routing congestion; ❸ the design must be **adaptable across different fabrication technologies or device variants**, each introducing variations in component size and behavior that impact waveguide routing; ❹ **frequent design iterations** are required, making manual updates time-consuming and error-prone, especially as schematic designers often lack full visibility into waveguide routing constraints or the necessity for inserting crossings. Such scenarios frequently lead to repeated back-and-forth between schematic and layout teams, particularly during layout exploration phases involving iterative placement and routing refinement.

Most existing studies primarily focus on global routing planning for PICs. A number of optical routing algorithms have been proposed for on-chip 3D system-on-package designs [10], [11], with the primary goal of optimizing signal loss and overall power consumption. Tools such as PROTON [12] and PLATON [13] offer automatic placement and routing, employing a modified Lee’s algorithm for waveguide routing. In [14], further improvements are achieved by optimizing device orientation, such as flipping and rotation, to reduce insertion loss through fewer crossings. While these global routing techniques are

The preliminary version has been accepted by the IEEE/ACM International Symposium on Physical Design (ISPD) in 2025.

H. Zhou and J. Gu are with the School of Electrical, Computer and Energy Engineering, Arizona State University, AZ, USA (e-mail: jiaqigu@asu.edu). N. Gangi and Z. Huang are with the Department of Electrical, Computer, and System Engineering, Rensselaer Polytechnic Institute, NY, USA. H. Yang and H. Ren are with NVIDIA, TX, USA. J. Matres is with GDSFactory, CA, USA.

effective in identifying low-loss paths, they often focus solely on logical path planning and tend to neglect physical realizability. As a result, they may suffer from routing congestion, infeasible bend or crossing insertions, ultimately leading to an invalid routing solution.

Some research efforts have also targeted the detailed routing stage in PIC layout design. The work [15] proposed a method using mixed integer programming for global routing, followed by Manhattan grid-based detailed routing with crossings as constraints. However, this grid-based method only supports 90 bends. A follow-up approach [16] introduced non-Manhattan channel routing to better accommodate curved waveguides. Despite this, modern PICs typically rely on a single optical waveguide layer, making crossings unavoidable and leaving limited flexibility for crossing optimization. To address these challenges, a fully automated PIC router is needed that is physically aware of waveguide and component instantiations and capable of performing design-rule-compliant routing with intelligent crossing insertion.

In this work, we propose LiDAR 2.0, a hierarchical PIC detailed routing tool that adopts a bottom-up routing strategy, supports non-Manhattan curvy waveguides, and enables adaptive crossing insertion. It addresses key limitations of existing methods by jointly optimizing insertion loss and enforcing layout constraints, with full awareness of waveguide, bend, and crossing geometries. Unlike schematic-driven approaches that require manual crossing insertion, LiDAR 2.0 adaptively inserts crossings during routing and produces nearly design-rule-violation (DRV)-free layouts within minutes, significantly reducing the need for post-routing fixes or iterative schematic updates. The main contributions are summarized as follows.

- 1) We propose a fully automated PIC detailed routing tool, LiDAR 2.0, that generates real GDSII layouts with low insertion loss, supporting curvy waveguide geometries and automatic crossing insertion for large-scale photonic circuits within minutes.
- 2) PIC Routing Benchmark: We introduce a hierarchical PIC intermediate representation (PIC IR) and open-source scalable benchmark generators for PICs, spanning diverse routing complexities, enabling realistic and challenging evaluation of PIC routing algorithms.
- 3) Curvy-Aware Non-Manhattan A* Routing: We develop a customized A* search algorithm with adaptive neighbors to efficiently handle curvy structures and non-Manhattan routing patterns.
- 4) Conflict-Resilient Routing: We boost routability through accessibility-enhanced port assignment, congestion and crossing space penalties, and group-based net ordering with congestion-penalized rip-up and reroute, effectively reducing access conflicts and unnecessary crossings.
- 5) Evaluations show that LiDAR 2.0 supports PIC routing with various crossing sizes and bend radii, achieving DRV-free layouts with 16% lower insertion loss (IL) and 7.69x speedup over prior methods on spacious benchmarks, and nearly DRV-free layouts with 9% lower IL and 6.95x speedup over LiDAR 1.0 on compact benchmarks.

TABLE I: Notations used in this paper.

Symbol	Description
N	The set of nets specified in the circuit netlist.
n_i	The i^{th} net in N , $1 \leq i \leq N $.
P	The set of all paths.
p_i	The i^{th} path in P , $1 \leq i \leq P $.
$IL(p_i)$	The insertion loss of the p_i .
IL_{\max}	The maximum insertion loss over all paths.
$IL_{wg}(p_i)$	The propagation loss of the path.
$IL_{cr}(p_i)$	The crossing loss of the path.
$IL_{bn}(p_i)$	The bending loss of the path.
w, c, b	Coefficient of $IL_{wg}(p_i)$, $IL_{cr}(p_i)$, and $IL_{bn}(p_i)$.
g_i	The i^{th} port group.
w_{g_i}	Check region of group-based congestion penalty.
w_{cr}	Check region of crossing space penalty.
c	The coefficient of congestion and crossing space penalty.
s	Routing grid size.

(a) (b) (c)
(d) (e) (f)

Fig. 2: Compare properties/rules of EIC and PIC routing.

II. PRELIMINARIES

This section first reviews related VLSI routing methods and outlines key differences from PIC routing by examining PIC-specific design rules. We then describe the conventional manual PIC routing flow, followed by routing evaluation metrics and associated challenges. Notations used in this paper are summarized in Table I.

A. VLSI Detailed Routing

VLSI detailed routing must address challenges such as complex design rules, pin accessibility, and limited routing resources [17]. Common routing approaches [18], [19], [20], [21] employ path-finding algorithms like A* search or maze routing, typically supported by a design rule checking (DRC) engine. A widely adopted strategy to resolve routing conflicts is negotiation-based routing [22], which iteratively applies rip-up and reroute techniques to eliminate routing failures. One of the key differences between VLSI and PIC routing lies in the routing geometry. While VLSI routing typically follows Manhattan or unidirectional styles, PIC routing requires curvilinear waveguides. To support more flexible layouts, octagonal routing and other non-Manhattan styles have been explored in analog [23], [24], PCB [25], [26], [27], and package routing [28], [29]. However, research specifically targeting curvy path routing remains limited.

B. Photonic Design Rules

PIC routing typically operates on a single waveguide layer, where all nets are 2-pin optical paths. We summarize key design rules and highlight photonic-specific considerations. 1) Waveguide Type: PICs often contain multiple types of waveguides, whose characteristics depend on factors such as wavelength, polarization mode, substrate type, and cross-section, as shown in Fig. 2(a). Waveguide connections must be

strictly matched in type or transitioned via tapers to minimize loss. Different waveguide types also have different spacing requirements to prevent unwanted crosstalk. For high-index contrast systems (e.g., silicon-on-insulator), relatively small spacings of 1–3 μm are typically sufficient.

2) Bend Radius: In contrast to the sharp bends used in VLSI metal routing, waveguides in PICs require smooth bends to minimize mode mismatch and radiation losses. To mitigate these losses, bends are typically implemented as circular arcs, Euler curves, or sine curves, depending on the routing scenario, as illustrated in Fig. 2(b).

The bend radius can vary significantly from a few microns to millimeters, depending on material properties and waveguide design. For example, silicon waveguides with high refractive index contrast support small bend radii of 5–10 μm, while silicon nitride waveguides, which have lower index contrast, typically require larger bend radii of 20–100 μm. Although larger bend radii reduce insertion loss, they also increase area consumption and may limit routing flexibility.

3) Waveguide Crossing: Unlike VLSI routing, which forbids wire crossings and uses vias for layer transitions, PICs permit waveguide crossings (CRs) on the same layer, which are often necessary for high-density designs. Each crossing introduces insertion loss, typically 0.1–1 dB, and occupies an area of approximately 5.5 μm².

The crossing angle is also critical: CRs should ideally intersect at 90° to minimize crosstalk. This constraint complicates routing in dense layouts, as parallel waveguides require extra space to adjust their direction via bending before forming perpendicular intersections, as shown in Fig. 2(c).

4) Port Connection and Alignment: Waveguides are connected via precise port abutment, which requires exact face-to-face alignment. When there is an offset between ports, additional bending is required to compensate for the misalignment as illustrated in Fig. 2(d). Misalignment or offset at the ports can break the optical path, making precise alignment essential for successful routing. An example of correct alignment is shown in Fig. 2(e).

5) Signal Integrity: Insertion loss is a critical metric for evaluating PIC routing quality, as it directly affects the laser power budget and signal integrity, including signal-to-noise ratio and crosstalk, as illustrated in Fig. 2(f). The primary evaluation metric is the maximum insertion loss along the critical path. Long waveguides and excessive crossings degrade signal integrity and should be minimized wherever possible.

C. Schematic-Driven PIC Layout

Traditional PIC layout workflows, including manual design and current EPDA tools, are schematic-driven [30]. In this approach, all structures, including crossings and waveguide segments, are instantiated explicitly. Designers must plan routing during the schematic stage and manually insert crossings, while waveguide connections rely on port abutment without explicit physical net definitions.

A major limitation is that waveguide paths and crossings must be predetermined by design experts during schematic creation, based largely on empirical estimates of the final layout. Once these paths are defined, it becomes difficult to

modify them during physical design, leading to a rigid and manually constrained routing topology. This inflexibility often results in repetitive revisions between the schematic and layout stages, which can be time-consuming and error-prone, and becomes impractical for large-scale designs.

To overcome these limitations, it is essential to adopt a new formulation of instances and nets that decouples schematic design from physical implementation. This will enable automated crossing insertion and routing flexibility, improving efficiency and scalability in PIC layout design.

D. PIC Routing Quality Metrics

In addition to standard routing metrics such as wirelength, design rule violations, and runtime, a key photonic-specific metric is the insertion loss (IL), which directly affects the link power budget and signal-to-noise ratio. IL is calculated based on the optical path, which represents the trajectory of light propagation through all cascaded components from the laser source to the photodetector.

Assume a path p_i consists of alternating instances and nets $(m_0 ! n_0 ! m_1 ! n_1 ! \dots)$. Some instances and nets may be shared across different paths. For simplicity and due to the lack of accurate port-specific IL data in available open-source PDKs, we assume uniform IL between any input-output port pair in multi-port photonic devices. Nevertheless, port-specific ILs can be incorporated into the same formulation when available. The total insertion loss $IL(p_i)$ of path p_i is defined as the sum of insertion losses from all devices IL_d and all routed waveguide segments IL_{wg} along the path.

The loss is measured in decibels (dB), following standard convention. For net-level IL, we consider three types of losses: crossing loss (IL_{cr}), bending loss (IL_{bn}), and propagation loss (IL_{wg}). Therefore, we have:

$$IL(p_i) = \sum_{m_j \in p_i} IL(m_j) + \sum_{n_j \in p_i} IL(n_j)$$

$$IL(n_j) = IL_{wg}(p_i) + IL_{cr}(p_i) + IL_{bn}(p_i)$$

$$IL_{wg}(p_i) = w \cdot WL_{p_i}; \quad IL_{cr}(p_i) = c \cdot \#CR_{p_i}; \quad IL_{bn}(p_i) = b \cdot \sum \Delta \theta_{p_i}; \quad (1)$$

where WL_{p_i} is the total waveguide length, $\#CR_{p_i}$ is the number of waveguide crossings, and $\sum \Delta \theta_{p_i}$ is the cumulative bend angle along the path. The coefficients w , c , and b represent the insertion loss per unit waveguide length, per crossing, and per unit bend angle, respectively, and are determined by the specific photonic technology and component structures. Minimizing insertion loss is crucial to achieving the desired optical performance and signal integrity for switching, modulation, or multiplexing applications.

The maximum insertion loss across all paths, denoted as IL_{max} , determines the worst-case optical budget and sets the minimum required laser power to ensure reliable detection at the outputs. Therefore, IL_{max} serves as the primary evaluation metric for PIC routing. The routing objective is expressed as:

$$IL_{max} = \max_{p_i \in P} IL(p_i) \quad (2)$$

Algorithm 1 Port Assignment Procedure

Input: Set of components with directional ports (C), crossing size (s), bend radius (r)
Output: Port assignment result and reserved regions

- 1: for each component c in C do
- 2: for each direction d in $\{0, 90, 180, 270\}$ do
- 3: for ports $p \in \mathcal{P}_d$ do
- 4: Sort p based on coordination
- 5: if p lie inside c 's bounding box then
- 6: Port Propagation
- 7: if p share the same routing grid then
- 8: Congested Port Spreading
- 9: Reserved_length = $f(\text{index}(p); r; s)$
- 10: Assign reserved region along p orientation

Fig. 3: Algorithm flow of our LiDAR framework.

E. Problem Formulation

We formulate the PIC detailed routing problem as follows:

Problem 1 (PIC Detailed Routing). Given a set of nets $N = \{n_i | 1 \leq i \leq |N|\}$ and a set of placed devices $M = \{m_j | 1 \leq j \leq |M|\}$, generate a routing solution for each net $n_i \in N$ such that all connections are completed without design rule violations, while minimizing I_{\max} .

III. LiDAR: AUTOMATED PIC DETAILED ROUTING

With an input circuit netlist, as shown in Fig. 3, we present LiDAR, a detailed PIC routing framework based on customized grid-based A* search. It efficiently generates curvy waveguide paths with automatic crossing insertion, minimizing maximum insertion loss while ensuring design-rule compliance. The core routing flow has three phases: 1) Port Access Assignment: assigns ports by orientation and local density to ease routing and reduce congestion; 2) Iterative Curvy-Aware Routing: routes all nets with a curvy-aware A* search guided by group-based net ordering; 3) Crossing Optimization and Refinement: applies local rip-up and reroute to resolve conflicts and optimize crossings, then refines results to remove redundant bends and produce a clean GDS layout.

A. Accessibility-Enhanced Port Assignment

Port accessibility is one of the most critical and challenging subroutines in PIC detailed routing. Unlike unidirectional metal pins in VLSI, PICs use directional waveguide ports that require exact face-to-face orientation and precise alignment as shown in Fig. 4(a). Improperly oriented waveguides cannot legally connect to the target port, especially when insufficient space is available to reorient the waveguide using bends. This problem becomes even more pronounced when nearby waveguides obstruct the port region. This difficulty stems from the large area needed for curvy bends. To address this, we propose port access assignment techniques that consider both port orientation and spatial density to enhance accessibility. Some waveguide ports are inside the bounding box of a device. Since devices are treated as routing blockages, we propagate these internal ports outward to

device boundary based on their orientations, as illustrated in Fig. 4(a). This ensures they are reachable during routing. Congested Port Spreading Certain PIC devices feature dense clusters of ports that may overlap on the same routing grid, leading to access conflicts. To alleviate this, we apply an asymmetric spreading strategy, where high-density ports are distributed with a predefined spacing and extension length, as shown in Fig. 4(b). These new access ports are connected to their original locations via sine bends. We add 5 units of extension length for each grid shift, ensuring they occupy distinct routing tracks and satisfying the minimum bend radius. Staggered Port Access Region Reservation To prevent waveguides from blocking ports, a region in front of each port is reserved along its orientation (Fig. 4(a)). For multiport devices, all ports can still cause blockage, while closely spaced parallel waveguides hinder crossing insertion (Fig. 4(c)), since

(a)

(b)

(c)

Fig. 4: (a) Port propagation and reserved port region help port access. (b) Port spreading removes congested ports in the same grid. (c) group-based net order with access point offset enables channel planning and allows potential crossing.

a crossing requires a minimum footprint and cannot be placed consecutively without sufficient spacing. To address this, we introduce the staggered port access region strategy, ensuring adjacent access points are spaced larger than the crossing footprint, leaving sufficient room for accessibility. The reserved length depends on the port's order among ports of the same orientation, as well as crossing size and bend radius. Ports are grouped by orientation and sorted by coordinates along the perpendicular axis, each assigned an index p_i . The reserved length for port p is calculated by:

$$\left(\frac{p_{\text{num}}}{2} - p_i + \frac{p_{\text{num}}}{2} \right) s + r \quad (3)$$

where p_{num} is the number of ports in the group. This yields a staggered, mountain-shaped profile that prevents inner ports from being blocked and facilitates escape. If routing fails, the reserved length is adjusted according to the updated routing order in Section IV-B.

B. Port-Group-based Net Order

LiDAR is a sequential router that processes one net at a time, where the routing order has a significant impact on both quality and feasibility. To address this, we propose a port-group-based net ordering strategy that clusters ports on the same device by direction, denoted as g_1 , 0 and 180 ports form groups g and \bar{g} in Fig. 4(c). The routing proceeds group by group based on an inter-group routing order. Within each group, nets are routed in a defined intra-group order before moving to the next group. This strategy is motivated by the observation that most congestion and routing conflicts occur between nets in the same group. By routing nets group-wise with mutual awareness, intra-group conflicts are effectively reduced, improving overall routing quality.

The inter-group routing order of g_i is first determined by its group priority score $S_g = \min_{n_k \in g_i} \text{dist}_{n_k}$, where dist_{n_k} is the Euclidean distance of net n_k . A smaller S_g indicates a higher routing priority for that group. If $S_{g_1} = S_{g_2}$, the group that enters the priority queue earlier will be routed first. After that, we define the net routing order within the same group

Since ports within a group form a staggered access point region, they are more accessible and are designated as target ports, while the others serve as sources to reduce conflicts. However, the access order (i.e., intra-group routing order) is critical; an improper access sequence may lead to ports being blocked by waveguides or other port regions. Therefore, we determine the access sequence based on the source ports' spatial distribution. Details are provided in Section IV.

C. Non-Manhattan Waveguide Routing with Curvy-Aware A

Unlike Manhattan-style VLSI routing, PICs require non-Manhattan paths to support smooth curves that reduce bending and insertion loss. We propose an iterative waveguide routing algorithm based on A search, supporting both 45 and 90 degree turns with adaptive crossing insertion. Built on an existing VLSI-proven framework, the algorithm enables efficient multi-objective optimization for PICs.

Fig. 5: Parametric curvy-aware neighbors allow non-Manhattan curvy waveguide routing. Neighbors are automatically derived based on bending radius and grid size.

1) Spacing-Ensured A Routing Grid Size Setting: In LiDAR, the routing grid size s is set to be larger than the waveguide width. Since waveguides in PICs are generally wider than their ports, this setting improves routing efficiency and simplifies port access by reducing local congestion near narrow access points.

2) Parametric Curvy-Aware Neighbor Candidate Generation: To enable curvy-aware A search, we propose a parametric neighbor generation scheme that leverages bending geometry and performs comprehensive design rule checks to filter out illegal candidates.

While traditional VLSI and PCB routing often use sharp 90 or 45 degree turns, PIC routing requires smooth curves. We define each routing node by its spatial location and orientation, represented as a directional node $(x; y; \text{orientation})$. This directional representation is essential for maintaining alignment and orientation during port access. As illustrated in Fig. 5, neighbor candidates are determined by the node's current orientation and a user-defined bending radius r . Based on orientation, nodes are categorized into two states: Manhattan State (MS) and Non-Manhattan State (NMS). MS nodes, which align with the x- or y-axis, have five possible neighbors: one directly forward and four at angular offsets of 45 and 90 degrees. In contrast, NMS nodes, which follow diagonal trajectories, have three potential neighbors.

The exact positions of neighbor candidates are adaptively computed using the bend radius r and grid size s . Larger bending radii and smaller grids lead to larger step sizes. For example, for an MS node oriented at 0 degrees, its directly adjacent neighbor is 1 grid unit away, while the steps to reach the 90 and 45 degree neighbors are calculated as:

$$\begin{aligned} \text{step}_{90;x} &= \text{step}_{90;y} = dr = se; \\ \text{step}_{45;x} &= d \left(\frac{p}{2} - 1 \right) r = se; \quad \text{step}_{45;y} = d \left(1 - \frac{p}{2} \right) r = se; \end{aligned} \quad (4)$$

Here, $\text{step}_{90;x}$ and $\text{step}_{90;y}$ represent the horizontal and vertical grid distances required to reach a neighbor at a 90 degree turn. The ceiling function $\lceil \cdot \rceil$ guarantees that the bend radius constraint is met. Note that unlike 45 degree diagonal neighbors turns with adaptive crossing insertion. Built on an existing VLSI-proven framework, the algorithm enables efficient multi-objective optimization for PICs.

Fig. 6: Proposed adaptive waveguide crossing insertion.

Fig. 7: Left: group-based congestion penalty in Eq. (5). Right: crossing-space penalty allows consecutive crossing insertion.

3) Geometry-Aware Neighbor Legality Check: To ensure only feasible neighbor candidates are explored during A* search, each candidate must pass a geometry-aware legality check before being added to the priority queue. A neighbor is considered legal only if the corresponding waveguide segment does not violate any design rules upon geometric instantiation.

Hit No Obstacle: Geometry-Aware Spacing Check If the neighbor does not intersect any obstacle, we instantiate the actual geometry of the connecting waveguide segment and perform a spacing check to verify that the path complies with design rules and is free of violations (DRVs).

Hit Routed Nets: Predictive Crossing Insertion If the neighbor candidate intersects with an already routed waveguide (marked as an obstacle), we evaluate whether a waveguide crossing can be inserted to legally pass through it.

As shown in Fig. 6, several critical conditions must be satisfied to allow crossing insertion: Sufficient straight waveguide length: Crossings occupy a defined footprint and require perpendicular waveguide orientations. We ensure there is enough straight length and validate the orientation by checking the routing grid state at the potential crossing location. No overlap with blockages: The bounding box of the crossing must not intersect any existing obstacles to comply with the design rules. Port matching: For a valid crossing, the intersecting waveguides must align with the four ports of the crossing structure. This includes matching properties such as waveguide width, cross-section, and layer. By predictively checking all those legality conditions, our method adaptively enables crossing insertion when needed. This significantly reduces unnecessary detours and eliminates the rigidity of manually defined crossings in schematic-driven flows.

4) Insertion Loss-Aware A* Search Cost: LiDAR adopts a customized A* search cost function that incorporates insertion-loss considerations to improve both routing quality and algorithm efficiency. The total cost of a node n in A* search is given by the standard formulation: $f(n) = g(n) + h(n)$, where $g(n)$ denotes the accumulated cost from the source node to the current node n , and $h(n)$ is the heuristic estimate of the

Fig. 8: Represent routed waveguides in oriented grid map.

cost from node n to the target node t . The cost $g(n)$ includes an insertion-loss-inspired term $g_{IL}(n)$, adapted from Eq. (1) with a tunable weight and a group-based congestion penalty (GCP), denoted as $g_c(n; g_i)$ to save routing resources for a group of nets. By adjusting the weight parameters, users can flexibly balance between waveguide length and the number of crossings to meet specific design objectives:

$$g(n) = g_{IL}(n) + g_c(n; g_i); \quad g_c(n; g_i) = c_{\#grids} w_{g_i}; \quad (5)$$

where $c_{\#grids}$ is a penalty coefficient that discourages routing too close to blockages or previously routed waveguides, preserving routing resources for unrouted nets within the same routing group, and $\#grids(w_{g_i})$ is the number of grids that occupied by others in the check region w_{g_i} as shown in Fig. 7. The check region w_{g_i} is determined by the number of unrouted nets in its port group. As more nets are routed, w_{g_i} decreases to avoid consuming extra space.

Since our method allows non-Manhattan routing, the shortest path is often diagonal. We define a customized heuristic $h(n)$ as shown in Eq. (6).

$$\begin{aligned} d_{min} &= \min(|j_n x - t_x|; |j_n y - t_y|); \\ d_{max} &= \max(|j_n x - t_x|; |j_n y - t_y|); \\ h(n) &= \begin{cases} d_{max} - d_{min} + \frac{\beta}{2} d_{min} + IL_{bd;45}; & (6) \\ 1; & \text{if } d_{min} > 0 \text{ and } d_{max} > 0; \\ 0; & \text{others} \end{cases} \end{aligned}$$

where d_{min} and d_{max} represent the minimum and maximum differences in the x and y coordinates between the current node n and the target node t , respectively. When $\beta = 0$, we have $d_{min} = 0$, and $h(n)$ reduces to the standard Manhattan case.

When $\beta = 0$, it indicates that the subsequent path search must contain at least one bend with an angle no smaller than 45 degrees. Therefore, we incorporate the loss of a bend into $h(n)$, which does not overestimate the cost and thus the A* router remains admissible, and penalizes the bend and encourages paths that end in orientations suited for port connection.

5) Waveguide Instantiation: A key distinction of LiDAR compared to prior global routing methods is its geometry-awareness. After determining a routing path, we instantiate the actual curvy waveguide geometry using the extrude function from GDSFactory [31], and record it on an overlapped, orientation-aware routing grid map, as illustrated in Fig. 8.

This enables the A* search engine to treat existing waveguides as obstacles and efficiently perform waveguide spacing checks and determine valid locations for crossing insertion.

Fig. 10: Waveguide re nement to remove bending.

Fig. 9: LRR check after nding a routing solution.

6) Violated Net Removal: When the router fails to access the target port with the correct orientation, LiDAR applies a rip-up-and-reroute (RR) strategy. DRC is temporarily relaxed to allow routing progress, and any nets that violate design rules or conflict with existing paths are recorded. These violating nets are then removed and rerouted in subsequent iterations. To prevent repeated routing failures and reduce congestion, a history cost [32] is maintained and updated in a global history map before each net is ripped up. This history-based negotiation mechanism discourages routing over previously congested regions and helps distribute paths more evenly across the layout. In practice, this approach effectively resolves routing failures and improves overall success by balancing routing demands across all nets.

D. Crossing-Waveguide Optimization

Waveguide crossings are sometimes inserted to bypass congested regions, circumvent obstructions, or avoid excessive detours. However, they not only introduce insertion loss, phase shifts, and occupy chip area, but they also cause crosstalk and potentially impact subsequent net routing. Consequently, we propose a local rip-up-and-reroute (LRR) strategy to eliminate unnecessary crossings and optimize the overall routing. As shown in Fig. 9, LRR is triggered after a solution is found. If the initial routing solution (RS) contains no crossings, it is accepted directly. If crossings exist, possible causes include: (1) a blockage caused by another waveguide requiring a crossing, (2) a crossing chosen to bypass congestion regions, or (3) high propagation loss for non-crossing paths. To verify these, we rerun the routing with crossings disabled (NCS). If NCS succeeds, its result is compared against RS by insertion loss, and the lower-loss path is selected. If NCS fails, it indicates the net is blocked. We then check the blocking waveguide. If the blocking waveguide has never been ripped up before, it will be ripped up, as this operation will not affect CR re-insertions in subsequent iterations. This LRR process empirically optimizes the routing by balancing long-range waveguides and CRs to reduce overall loss.

E. Routed Waveguide Re nement

Since the grid-based routing may cause slight misalignment between the port center and the grid center, a small offset can occur between the routed path and the port, as illustrated in Fig. 10. To resolve this, we adjust the initial and final segments of the waveguide to precisely align with the port, without altering the bend radius along the path. If direct adjustment

is not feasible, a sine bend is introduced at the port to ensure proper alignment.

IV. EXTENDED CONFLICT-RESILIENT HIERARCHICAL PIC ROUTING SCHEME

To enable a more compact layout and ensure successful routing under limited routing resources, it is essential not only to minimize bending and crossing but also to reserve sufficient space for them, as they are area-consuming. This becomes even more challenging in compact layouts. To address this, we further extend our framework to LiDAR 2.0, enhancing its conflict resolution capability and enabling successful routing even under tight layout constraints.

A. Hierarchical Netlist Tree Construction

PIC netlists often exhibit inherent hierarchical structure, where complex systems are composed of repeated functional modules, as shown in Fig. 11. Leveraging this structure is crucial for scalable physical design. To exploit such hierarchy, we construct a Hierarchical Tree by parsing the nested module instantiations defined in the YAML-format netlist, which is similar in style to a SPICE netlist and inherently preserves hierarchical relationships. In this tree, the root node corresponds to the top-level module of the circuit, the leaf nodes represent primitive photonic components, and the internal nodes represent intermediate-level submodules. The resulting tree has arbitrary fanout and depth, depending on the circuit topology. The construction has linear time complexity of $O(n)$, where n is the total number of unique modules and components. This hierarchical representation enables global routing planning at multiple levels.

We then perform bottom-up routing for each level of modules progressively. Within each level, the priority of each routing group is determined according to the group priority

(a)

(a) (b)

Fig. 12: (a) Add offset neighbor to remove redundant bending caused by port misalignment. (b) Preserve crossing space for the crossing nets.

score defined in Subsection III-B. This multi-level routing approach helps avoid conflicts between different modules, as routing is localized within each submodule before higher-level integration. Similar to standard-cell design in VLSI, the internal routing resources of a module are treated as private and should not be accessed by external nets.

To further improve efficiency, we reuse routing results of identical module types via translation, rotation, and flipping. Before reuse, we apply design rule checking at the target location to ensure legality. If a reused result violates constraints, we fall back to re-routing that specific instance from scratch. This reuse mechanism significantly accelerates layout generation for large-scale, hierarchical PIC designs.

B. Routing Conflict Reduction via Spatial-Aware Optimization

A? Offset Neighbor. Although the A algorithm discourages excessive bends via its bend-cost heuristic, it may still add unnecessary bends when ports are slightly misaligned. In tight spaces, such misalignment can cause multiple bends, raising insertion loss and risking routing failure (Fig. 12(a)).

To address this, we add offset neighbors under specific conditions: when the current node is within four bending radii of the target and their orientations differ by 180 degrees.

These neighbors correct small misalignments (below the bend radius) between source and target ports. Their locations are computed analytically from the bend radius and grid size, ensuring smooth, fabricable sine-bend waveguides. From an algorithmic perspective, offset neighbors simplify port access, cut redundant detours, and improve search efficiency.

Preserved Crossing Space A key source of routing conflict in PICs is the limited space for inserting crossings. Unlike VLSI vias, which can be inserted independently of other nets, photonic waveguide crossings must be placed directly on top of the crossed waveguides and require significant area, making insertion difficult in dense regions. As shown in Fig. 12(b), closely spaced waveguides may block required consecutive crossings, causing routing failure.

To mitigate the crossing space conflict, we introduce a crossing space penalty $g_c(n)$ during the A search to encourage the router to preserve sufficient spacing around the nets that are likely to be crossed, as inferred from the netlist topology. The $g_c(n)$ penalty is defined as:

$$g_c(n) = c_c \cdot \#grids(w_{cr}); \quad (7)$$

(b)

Fig. 13: (a) Routing conflicts due to inappropriate routing order. (b) Route based on the Manhattan distance between the target port and the mean of the source port locations.

Fig. 14: Rip-up blocking waveguides and update the routing order based on conflict precedence, while resizing target port regions to prevent access blockage for subsequent nets.

where c_c is a user-defined penalty and w_{cr} is the minimum width required to insert a valid crossing. The penalty is applied over a w_{cr} -wide region around the net segment, discouraging other nets from occupying this reserved space.

C. Reduce Routing Conflict via Routing Order Optimization

Although port regions allow staggered access points to reduce conflicts, an improper order may still block target ports with routed waveguides or other regions. For multi-port components, the intra-group routing order is thus crucial. As shown in Fig. 13(a), both orientation and spatial location of ports strongly influence order, as they determine accessibility and potential conflicts.

To address this, we derive the order from the spatial relation between source and target ports (Fig. 13(b)). Let the source port set be $P_s = \{p_1; p_2; \dots; p_n\}$ with positions $(x_i; y_i)$. Their mean coordinates are:

$$(x_s; y_s) = \left(\frac{1}{n} \sum_{i=1}^n x_i; \frac{1}{n} \sum_{i=1}^n y_i \right) \quad (8)$$

Routing order is then based on Manhattan distance from each target port to $(x_s; y_s)$, with closer ports given higher priority.

This method effectively resolves most conflicts, especially in multiport components.

Routing Order Refinement. It is difficult to guarantee an optimal routing order. When nets are routed independently without group information, the method above is inapplicable,

TABLE II: Spacious PIC benchmark statistics.

Benchmarks	# Components	# Nets	# Total CR	Die Size (m ²)	
Clements_8_8	52	79	0	4800	1600
Clements_16_16	168	287	0	8000	3200
ADEPT_8_8	82	111	33	4400	1600
ADEPT_16_16	162	223	55	6900	3200
ADEPT_32_32	318	446	121	13000	6400
Light_(a,b,c,d)	9	16	-	10000	10000

TABLE III: Compact PIC benchmark statistics.

Benchmarks	# Components	# Nets	# Total CR	Die Size (m ²)	
Clements_8_8_C	60	87	0	3500	1000
Clements_16_16_C	184	303	0	6800	2000
ADEPT_8_8_C	104	127	30	4400	1000
ADEPT_16_16_C	227	319	80	6200	2000
ADEPT_32_32_C	529	767	160	7800	4000
TeMPO_8_8_C	452	515	49	1700	1500
TeMPO_16_16_C	1796	2051	225	3000	3400
TeMPO_32_32_C	7172	8195	961	5700	6500
GWOR_16_16_C	17	32	-	4000	4000
GWOR_32_32_C	33	64	-	6000	6000
Benes_16_16_C	224	416	88	2200	1300
Benes_32_32_C	576	1024	416	3500	2500

making dynamic order adjustment necessary. An improper order may block ports with earlier waveguides, causing escape or access failures. As shown in Fig. 14, routing net1, net2, and net3 independently leads to net2's failure, as net1's port region and net3's waveguide block its path.

To address routing failures, we enhance rip-up and re-route by removing not only nets directly conflicting with the failed path but also those in front of the failed net's ports to ease congestion. As shown in Fig. 14, we scan along the port orientation, identify conflict order and conflicting nets (starting from θ_0), and use them to adjust routing order. For instance, if net3 is blocked by net2, and net2 by net1, we merge them and uniformly update the routing order from 3 to 1. When net3 and net2 are both blocked by net1 and their order is indistinguishable, we sort them by net names. Although higher-priority nets are routed first, they may still block others. To prevent this, we dynamically extend high-priority target ports. Specifically, the port length is extended by $jN \cdot i/r$, where N is the number of merged nets, i the order, and r the bend radius, shifting access points and reducing blocking risk. Our extended conflict-resilient hierarchical routing scheme can effectively enhance routability even under tight layout constraints and help accelerate the routing process.

V. PIC ROUTING BENCHMARK

PIC Intermediate Representation Inspired by the LEF/DEF formats widely used in electronic design automation, we introduce a YAML-based photonic intermediate representation (PIC IR) to describe the connectivity and types of photonic components. This IR supports hierarchical module definitions, enabling modular reuse and streamlined management of complex circuits. To bridge this abstract representation with physical layouts, we develop a translation tool that interfaces with GDSFactory, allowing for both visualization and generation of valid GDSII layouts from the IR.

Benchmark Suites Built upon the PIC IR, we further develop and open-source a suite of parameterized benchmark generators. These scripts enable users to generate benchmarks with configurable scales, such as the number of components, number of nets, and overall circuit dimensions. They also

support ne-grained control over routing complexity by adjusting parameters including component pitch (i.e., component density), port density, port misalignment, and the number of topological crossings, allowing for the creation of benchmarks with varying levels of routing difficulty. The benchmark suite includes both photonic tensor core and optical switch, providing a diverse set of testcases to evaluate the router in realistic PIC settings. Users can also define their own PIC benchmarks easily by using our interface.

For the photonic tensor core, we provide the following benchmarks, each available at multiple scales:

Clements [33]: A highly structured Mach-Zehnder interferometer (MZI) array with a regular topology and no topological crossings. It is relatively straightforward to route when sufficient layout space is available.

ADEPT [34]: An auto-searched photonic tensor core that includes numerous multiport components and a large number of inherent crossings, resulting in high routing congestion and increased difficulty.

TeMPO [35]: A time-multiplexed dynamic photonic tensor accelerator with a hierarchical crossbar structure. It contains multiple computing nodes and large fan-out MMI components, along with a significant number of crossings, making routing particularly challenging.

For the optical switch category, we use benchmarks below:

Light [36]: Based on macro-ring resonators (MRRs), it features an unstructured topology with a centralized MRR macro. We define four layout variants according to memory controller and hub placement to evaluate routing under different traffic patterns, focusing on minimizing crossings and insertion loss.

GWOR [37]: Also MRR-based, it splits the central switch into multiple distributed parts, with I/O ports placed uniformly along the chip periphery and no group-level planning. This benchmark emphasizes routing conflict resolution via pure spatial reasoning.

Benes [38]: Based on Mach-Zehnder interferometers, it adopts a structured, hierarchical Benes topology. It exhibits extremely high crossing density, where a single net may require tens of crossings, challenging both crossing reservation and congestion-aware routing strategies.

Tables II and III summarize the benchmark statistics. Total #CR refers to the number of topological crossings inherent in the circuit and resolved during routing. For Light and GWOR, internal switch crossings are excluded, as we focus solely on the routing between the switch and the I/O interfaces in these benchmarks. Table II (ISPD version) includes spacious, moderately complex layouts, while Table III features more compact benchmarks with higher component density, smaller die sizes, and more crossings, posing greater routing challenges.

For the Light and GWOR benchmarks, the bending radius, waveguide width, and grid resolution are set to 60 μ m, 2 μ m, and 50 μ m, respectively, to emulate the larger bend radii of the SiN platform. Since these circuits provide abundant routing resources, a larger grid size is adopted to accelerate the search process. For the other benchmarks, these values are set to

5 μm , 0.5 μm , and 2 μm , respectively, reflecting the tighter bends of the SiPh platform. By default, the crossing size is fixed at 10 μm \times 10 μm .

VI. EVALUATION RESULTS

The proposed photonic detailed routing framework is implemented in Python based on GDSFactory 8 [31] libraries. All evaluations are conducted on a Linux server with a 128-core AMD EPYC 7763 CPU and 512-GB memory. Note that runtime and results may slightly differ from the ISPD version due to platform and GDSFactory version variations.

The placement solutions of all benchmark circuits are verified with simulation using GDSFactory and KLayout. Device insertion losses used for critical path IL evaluation are listed in Table V, with bending loss proportional to angle.

A. Search Parameters Referring to the parameters in Table V and scaling them to a unified micrometer unit, the base propagation, bending, and crossing weights are set to 1.5, 50 and 5000, respectively. On top of these, we further adjust the weights according to the characteristics of each benchmark. For the photonic tensor core, where crossings are inherent and unavoidable, we set the crossing weight equal to the bending weight to speed up search. For the optical switch, the crossing weight remains 5000, while the bending weight is also set to 5000 to reduce bends; in long waveguides with large radii, shorter wirelength can offset added bend loss. The group congestion penalty (GCP) parameter is determined by the detour distance, as saving routing resources through GCP may conflict with wirelength minimization. We set this value to 500, corresponding to reserving approximately 200 μm of spacing. The crossing spacing coefficient is set similarly, but its checking range is limited to a single crossing.

Baselines We compare LiDAR 2.0 with PROTON [12], which performs path planning with adaptive crossing penalties but lacks valid waveguide geometry generation. For fair comparison, we adapt PROTON with reserved port regions and global rip-up and reroute support. If no feasible solution is found due to congestion, DRC constraints are temporarily relaxed for specific failed nets for meaningful comparison of #DRVs. LiDAR 1.0 is also included to demonstrate the improvements in conflict handling, crossing insertion, and runtime efficiency in LiDAR 2.0.

A. Results on Spacious Benchmarks

We first evaluate the routing algorithm under spacious layout conditions. Table IV presents a comparison of routing metrics on the spacious benchmarks. In this setting, LiDAR 1.0 and LiDAR 2.0 achieve comparable performance and successfully generate DRV-free layouts across all benchmarks. Compared to PROTON, LiDAR 2.0 achieves an average 16% lower critical path IL and 7.69 speedup.

Analysis of PTC Results Despite the Clements circuit having a regular topology with no inherent crossings and relatively spacious layouts, insufficient planning and lack of reserved access space lead to unnecessary crossings and DRVs in PROTON. These routing conflicts become more severe in ADEPT benchmarks, which feature dense multi-port MMIs and a large number of crossings. By incorporating Accessibility Enhanced Port Assignment and Port-Group-Based Net solutions remain available at 15 μm despite minor DRVs,

Ordering, LiDAR 2.0 effectively mitigates these conflicts and reduces unnecessary rip-up and reroute. As circuit scale increases, PROTON suffers from escalating #DRVs and runtime, while LiDAR 2.0 consistently produces DRV-free, low-IL layouts.

Analysis of Optical Switch Results Light benchmark provides abundant routing resources and is primarily used to evaluate the router's ability to optimize insertion loss and crossings. Notably, there exists a counter-intuitive trade-off between #CR and WL; minimizing #CR may lead to long detours that increase propagation loss, resulting in a higher overall IL_{max} . Except for the Light_b case, LiDAR achieves the lowest IL_{max} across all other cases while maintaining crossing-optimal (#CR=0), DRV-free layouts.

B. Results on Compact Benchmarks

We further evaluate routing on compact benchmarks with higher component density and more crossings. Only one rip-up and reroute round is applied, as limited space makes further iterations ineffective. Figure 15 shows the final layouts by LiDAR 2.0.

As in Table VI, LiDAR 2.0 delivers nearly DRV-free results with an average 16% IL reduction, 5.16 speedup over PROTON, and 6.95 speedup with 9% lower IL vs. LiDAR 1.0. On TeMPO, PROTON runs faster only because complete port blocking forces early termination. For compact cases, LiDAR 1.0 shows more DRVs and runtime, especially on new benchmarks. In contrast, LiDAR 2.0's hierarchical routing reuses subcircuits (TeMPO, Benes), greatly improving runtime. Intra-group ordering and proactive crossing-space reservation further reduce conflicts, enhance quality, and avoid redundant search. Thus, LiDAR 2.0 achieves DRV-free routing even on high-crossing Benes. For GWOR, lacking group data, LiDAR 2.0 still succeeds by dynamically reordering and port regions during reroute.

Figure 16 shows the runtime breakdown on Benes_32 32. The speedup of LiDAR 2.0 comes from reducing conflicts, eliminating redundant search and RR iterations, and hierarchically reusing results.

C. Results on Different Crossing Sizes and Bend Radii

Different PIC technologies impose varying constraints on crossing sizes and bending radii. Table VII and Table VIII show the routing results of LiDAR 2.0 under different crossing sizes and bend radii. We synthetically stretch the device only to increase the routing difficulty and test our router's capability, regardless of its actual IL value. Benchmarks with an L suffix indicate larger component pitch. As expected, larger crossing sizes and bend radii increase area usage and routing difficulty. Still, LiDAR 2.0 consistently generates DRV-free layouts across compact and spacious benchmarks, with only minor DRVs in ADEPT cases. LiDAR 2.0 supports the insertion of various crossing sizes and remains robust under different bend radii. It successfully routes GWOR_32 32 even with large bend radii up to 80 μm , which is typical in silicon nitride platforms. For silicon photonics, where the bending radius is typically 5–10 μm , LiDAR 2.0 performs reliably. Feasible solutions remain available at 15 μm despite minor DRVs,

TABLE IV: Comparisons of the maximum insertion loss IL_{max} (dB), the path length with IL_{max} (WL (mm)), the number of crossings passed by the signal with IL_{max} , total design rule violations (#DRV), and runtime (s) for spacious benchmarks.

Benchmarks	PROTON [12] (Adaptive crossing penalty)					LiDAR 1.0 [39]					LiDAR 2.0				
	#CR	WL	IL_{max} #	#DRV #	Time # (s)	#CR	WL	IL_{max} #	#DRV #	Time # (s)	#CR	WL	IL_{max} #	#DRV #	Time # (s)
Clements_8 8	0	3.39	16.99	0	112	0	2.94	16.38	0	29	0	2.89	15.98	0	7
Clements_16 16	5	5.06	29.31	12	527	0	4.38	26.74	0	144	0	4.07	26.03	0	61
ADEPT_8 8	16	4.70	17.12	26	194	18	4.10	18.00	0	71	18	3.99	17.63	0	65
ADEPT_16 16	28	7.84	24.07	98	1395	16	7.38	17.80	0	243	16	6.95	17.20	0	243
ADEPT_32 32	66	16.13	44.57	355	10894	50	15.04	36.34	0	1348	50	14.70	36.06	0	1038
Light_a	12	32.98	11.09	0	39	0	31.11	7.78	0	101	0	29.99	7.61	0	103
Light_b	6	18.71	5.89	0	8	0	21.55	6.31	0	44	0	21.12	6.24	0	47
Light_c	14	20.81	10.23	1	52	0	35.29	8.40	0	72	0	35.92	8.57	0	74
Light_d	13	28.49	10.94	1	53	0	33.52	8.14	0	80	0	32.82	8.09	0	83
Geo-mean	-	15.35	18.91	-	1475	-	17.26	16.21	-	237	-	16.94	15.93	-	191
Ratio	-	1	1	-	1	-	1.12	0.86	-	0.16	-	1.1	0.84	-	0.13

(a) (b) (c)
(d) (e)

Fig. 15: Layout of (a) Benes_16 16 (b) TeMPO_16 16 (c) GWOR_32 32 (d) Clements_16 16_C (e) ADEPT_16 16_C generated by LiDAR 2.0.

TABLE V: Device IL parameters used in IL_{max} evaluation.

Propagation w	90 Bend b	CR c	Y-branch	MZI	MMI
1.5 dB/cm [40]	0.005 dB [40]	0.52 dB [12]	0.3 dB [41]	1.2 dB [42]	0.1 dB [43]

Fig. 17: Layout of Light_a of different crossing loss.

Fig. 16: Runtime breakdown of PROTON [12], LiDAR 1.0 [39], and LiDAR 2.0 in percentages on Benes_32 32 with total runtime marked in the center.

except for the TeMPO and Benes benchmarks, where no valid routing solution exists due to insufficient spacing.

D. Discussion

Manual Routing Result. We conducted a manual routing comparison on small circuits, Light_a and ADEPT_4 4, as shown in TABLE IX. It can be observed that the routing quality of LiDAR 2.0 is comparable to that of manual routing, and even achieves lower insertion loss on Light_a. However, manual routing required approximately two hours to complete for these small circuits, and as the circuit scale increases, manual efforts become impractical.

Spacing Reservation We propose several techniques for reserving spacing, including staggered offset port region, re-

served crossing spacing, group congestion penalty, and group-based net ordering to mitigate routing conflicts in multiport components. TABLE XI highlights their impact on representative circuits. It can be observed that the staggered offset port region is most critical: removing it causes many routing failures. Without the crossing space penalty, the Benes circuit suffers heavily due to consecutive crossings, while TEMPO is less affected since its crossings are independent. Disabling group-based net ordering, however, triggers numerous DRVs in TEMPO because of high-fanout components. We also assess the proposed GCP in guiding crossing optimization under different crossing weights (Table X). With larger c , the router avoids crossings to reduce maximum insertion loss (IL_{max}); with smaller c , it favors shorter paths even with extra crossings, further improving IL_{max} . Without GCP, more crossings arise as c increases, since

TABLE VI: Comparisons of the maximum insertion loss IL_{max} (dB), the path length with IL_{max} (WL (mm)), the number of crossings passed by the signal with IL_{max} , total design rule violations (#DRV), and runtime (s) for compact benchmarks.

Benchmarks	PROTON [12] (Adaptive crossing penalty)					LiDAR 1.0 [39]					LiDAR 2.0				
	#CR	WL	IL_{max} #	#DRV #	Time # (s)	#CR	WL	IL_{max} #	#DRV #	Time # (s)	#CR	WL	IL_{max} #	#DRV #	Time # (s)
Clements_8_8_C	0	2.12	16.28	2	11	0	1.91	16.23	2	11	0	1.91	16.23	0	9
Clements_16_16_C	1	3.91	27.13	5	93	0	3.56	26.56	5	101	0	3.56	26.56	0	89
ADEPT_8_8_C	14	4.53	16.26	11	96	14	3.71	16.12	2	82	14	3.71	16.12	0	68
ADEPT_16_16_C	34	8.24	27.60	42	515	21	8.63	21.09	7	304	21	8.35	21.02	0	236
ADEPT_32_32_C	49	10.78	36.20	171	1496	31	10.58	27.07	30	735	30	9.83	26.48	3	568
TeMPO_8_8_C	12	1.73	18.48	14	95	10	1.90	17.33	5	147	7	2.83	15.86	0	44
TeMPO_16_16_C	27	3.47	34.58	35	251	23	3.86	32.27	24	1571	15	5.48	28.31	0	169
TeMPO_32_32_C	49	5.91	62.57	113	1125	46	6.11	60.45	57	8057	31	10.96	53.27	0	425
GWOR_16_16_C	30	3.09	16.72	35	19	27	2.78	15.02	26	15	21	2.80	12.03	0	7
GWOR_32_32_C	62	7.22	35.26	126	60	56	6.81	31.26	44	45	44	7.82	25.40	0	11
Benes_16_16_C	33	4.42	22.69	76	721	26	4.28	18.84	3	175	23	3.57	17.15	0	33
Benes_32_32_C	56	9.24	36.76	365	5117	68	9.08	42.39	78	1690	58	6.51	36.56	0	251
Geo-mean	-	5.39	29.21	-	800	-	5.27	27.05	-	1078	-	5.61	24.58	-	155
Ratio	-	1	1	-	1	-	0.97	0.92	-	1.33	-	1.04	0.84	-	0.19

TABLE VII: LiDAR 2.0 can handle various sizes of waveguide crossings in PICs with different layout densities.

Benchmarks	Crossing size = 10 10 m ²					Crossing size = 15 15 m ²					Crossing size = 20 20 m ²				
	#CR	WL	IL_{max} #	#DRV #	Time # (s)	#CR	WL	IL_{max} #	#DRV #	Time # (s)	#CR	WL	IL_{max} #	#DRV #	Time # (s)
ADEPT_16_16_C	21	8.35	21.02	0	236	21	8.54	21.08	0	299	21	9.08	20.96	3	379
ADEPT_16_16_L	21	8.84	21.05	0	339	21	8.80	21.06	0	347	21	9.18	21.13	1	373
TeMPO_16_16_C	15	5.48	28.31	0	119	15	5.41	28.30	0	127	15	5.33	28.29	0	131
TeMPO_16_16_L	15	4.86	28.22	0	132	15	4.79	28.21	0	136	15	4.71	28.19	0	139
Benes_16_16_C	23	3.57	17.15	0	33	23	3.45	17.13	0	34	23	3.34	17.15	0	34
Benes_16_16_L	22	3.70	16.68	0	34	22	3.57	16.62	0	28	23	3.49	17.23	0	35

TABLE VIII: LiDAR 2.0 can flexibly support PIC routing under various user-specified bend radii r (m). Different radii are shown according to the chip area and layout compactness of each benchmark.

Metrics	GWOR_32_32					Clements_16_16_C		ADEPT_16_16_C		Benes_16_16_C		TeMPO_16_16_C	
	$r=10$	$r=20$	$r=40$	$r=80$	$r=100$	$r=10$	$r=15$	$r=10$	$r=15$	$r=5$	$r=10$	$r=5$	$r=10$
WL	7.77	7.77	7.79	7.85	8.06	3.49	3.33	8.55	8.52	3.57	3.76	5.48	5.41
#CR	44	44	44	44	44	0	3	21	22	23	23	15	15
IL_{max}	26.40	25.40	25.40	25.41	25.44	26.54	26.25	21.06	21.62	17.15	17.32	28.32	27.90
#DRV	0	0	0	0	3	0	3	0	2	0	0	0	0
Time (s)	10	10	10	14	16	96	111	298	339	33	33	169	172

TABLE IX: Compare to manual routing on small PIC circuits. TABLE XI: Ablation results on representative benchmarks. Removing staggered port offset, group-based net ordering, or crossing space leads to more #DRV.

Metrics	Light_a		ADEPT_4_4	
	Manual	LiDAR 2.0	Manual	LiDAR 2.0
#CR	0	0	10	10
WL (mm)	31.89	29.99	2.30	2.33
IL_{max} #	7.85	7.61	12.95	13.14
DRV	0	0	0	0
Time	2h	103s	2h	26s

Benchmarks	w/o Staggered offset		w/o crossing space		w/o group-based net order	
	IL_{max}	#DRV	IL_{max}	#DRV	IL_{max}	#DRV
ADEPT_16_16_C	18.95	37	19.97	6	21.63	4
ADEPT_16_16_L	19.13	33	20.34	3	21.95	3
TeMPO_16_16_C	28.49	27	28.31	0	31.87	16
TeMPO_16_16_L	28.72	26	28.22	0	32.64	16
Benes_16_16_C	14.62	21	16.24	32	16.83	2
Benes_16_16_L	14.37	21	16.93	30	17.21	4

TABLE X: Congestion penalty with different crossing costs.

Metrics	High Crossing Cost $c_c = 1$		Low Crossing Cost $c_c = 0.3$	
	w/o GCP	LiDAR 2.0	w/o GCP	LiDAR 2.0
#CR	6	0	5	5
WL (mm)	20.72	29.99	25.11	26.04
IL_{max} #	15.21	10.61	7.18	7.31
DRV	0	0	1	0
Time (s)	129	103	261	197

TABLE XII: Proposed offset neighbors reduce #DRV, IL_{max} , and runtime on TeMPO. C+ means more compact layout where x/y pitch drops from 25/50 to 20/15 m.

Benchmarks	w/o Offset Neighbor			w/ Offset Neighbor		
	IL_{max}	#DRV	Time (s)	IL_{max}	#DRV	Time (s)
TeMPO_8_8_C+	18.00	22	96	15.86	0	44
TeMPO_8_8_C	16.02	0	67	15.14	0	40
TeMPO_16_16_C+	34.12	71	470	28.03	0	144
TeMPO_16_16_C	28.58	0	215	28.31	0	119
TeMPO_32_32_C+	58.57	157	1713	52.28	0	494
TeMPO_32_32_C	53.94	0	673	53.27	0	425

earlier nets congest low-crossing paths. With GCP, routing conflicts drop notably, allowing efficient paths with fewer crossings and lower IL. In Fig. 17, c flexibly balances waveguide length and crossings to meet performance needs. Crossing-Waveguide Optimization As shown in Table IV (PROTON [12] vs. LiDAR 2.0), our proposed crossing optimization strategy introduces an extra runtime penalty on the high crossing penalty. Light benchmark, but it leads to higher solution quality as a? Offset Neighbor. Offset neighbors help eliminate unrec-

essary bending detours caused by port misalignment, thereby reducing routing conflicts and search overhead. We evaluate them on a compact TeMPO_C+ variant with x/y pitch reduced from 25/50 m to 20/15 m (Table XII). In such constrained layouts, routing without offset neighbors often causes rule violations. Even in spacious cases, enabling them yields consistently faster runtime.

VII. CONCLUSION

In this work, we present Li DAR, an open-source detailed router for PICs. It features a non-Manhattan curvy-aware A engine with enhanced port assignment, adaptive crossing insertion, congestion-aware group-based net ordering, and crossing optimization to address unique PIC constraints while minimizing critical path insertion loss. We further extend it into a scalable, conflict-resilient framework, Li DAR 2.0, by adding hierarchical routing reuse, intra-group ordering, offset-neighbor search, and crossing-space preservation. These enhancements markedly improve routability under tight layouts and high crossing density. Li DAR 2.0 achieves nearly DRV-free layouts on challenging benchmarks, with up to **9% lower IL** and **6.95 speedup** over Li DAR 1.0, advancing automation for large-scale PIC design.

REFERENCES

- [1] S. Hua, E. Divita, S. Yu *et al.*, “An integrated large-scale photonic accelerator with ultralow latency,” *Nature*, 2025.
- [2] S.R. Ahmed, R. Baghdadi, M. Bernadskiy *et al.*, “Universal photonic artificial intelligence acceleration,” *Nature*, vol. 640, no. 8058, pp. 368–374, 2025, epub 2025 Apr 9.
- [3] E. Taheri, S. Pasricha, and M. Nikdast, “ReSIPI: A Reconfigurable Silicon-Photonic 2.5D Chiplet Network with PCMs for Energy-Efficient Interposer Communication,” in *Proc. ICCAD*, 2022.
- [4] L. Thylén, S. He, L. Wosinski *et al.*, “The moore’s law for photonic integrated circuits,” *Journal of Zhejiang University-SCIENCE A*, vol. 7, pp. 1961–1967, 2006.
- [5] J. Pond, X. Wang, Z. Lu *et al.*, “Latest advancements to the industry-leading EPDA design flow for silicon photonics,” in *Proc. ICCAD*. IEEE, 2019, pp. 1–6.
- [6] T. Korthorst, W. Bogaerts, D. Boning *et al.*, “Photonic integrated circuit design methods and tools,” in *Integrated Photonics for Data Communication Applications*. Elsevier, 2023, pp. 335–367.
- [7] J. Feldmann, N. Youngblood, M. Karpov *et al.*, “Parallel convolutional processing using an integrated photonic tensor core,” *Nature*, 2021.
- [8] Y. Shen, N.C. Harris, S. Skirlo *et al.*, “Deep learning with coherent nanophotonic circuits,” *Nature Photonics*, 2017.
- [9] H. Hashemi, “A review of silicon photonics lidar,” in *Proc. CICC*. IEEE, 2022, pp. 1–8.
- [10] J.R. Minz, S. Thyagara, and S.K. Lim, “Optical routing for 3-d system-on-package,” *IEEE Transactions on Components and Packaging Technologies*, vol. 30, no. 4, pp. 805–812, 2007.
- [11] D. Ding and D.Z. Pan, “Oil: A nano-photonics optical interconnect library for a new photonic networks-on-chip architecture,” in *Proceedings of the 11th international workshop on System level interconnect prediction*, 2009, pp. 11–18.
- [12] A. Boos, L. Ramini, U. Schlichtmann *et al.*, “PROTON: An automatic place-and-route tool for optical networks-on-chip,” in *Proc. ICCAD*. IEEE, 2013, pp. 138–145.
- [13] A. von Beuningen and U. Schlichtmann, “PLATON: A force-directed placement algorithm for 3d optical networks-on-chip,” in *Proc. ISPD*, 2016, pp. 27–34.
- [14] Y.K. Chuang, K.J. Chen, K.L. Lin *et al.*, “Planaronoc: concurrent placement and routing considering crossing minimization for optical networks-on-chip,” in *Proc. DAC*, 2018, pp. 1–6.
- [15] C. Condrat, P. Kalla, and S. Blair, “A methodology for physical design automation for integrated optics,” in *2012 IEEE 55th International Midwest Symposium on Circuits and Systems (MWSCAS)*. IEEE, 2012.
- [16] C. Condrat, P. Kalla, and S. Blair, “Channel routing for integrated optics,” in *2013 ACM/IEEE International Workshop on System Level Interconnect Prediction (SLIP)*. IEEE, 2013, pp. 1–8.
- [17] G. Posser, E.F. Young, S. Held *et al.*, “Challenges and approaches in vlsi routing,” in *Proc. ISPD*, 2022, pp. 185–192.
- [18] A.B. Kahng, L. Wang, and B. Xu, “Tritonroute-wxl: The open-source router with integrated drc engine,” *IEEE TCAD*, vol. 41, no. 4, pp. 1076–1089, 2022.
- [19] M. Gester, D. Müller, T. Nieberg *et al.*, “Algorithms and data structures for fast and good vlsi routing,” in *Proc. DAC*, 2012, pp. 459–464.
- [20] H. Li, G. Chen, B. Jiang *et al.*, “Dr. CU 2.0: A Scalable Detailed Routing Framework with Correct-by-Construction Design Rule Satisfaction,” in *Proc. ICCAD*, 2019.
- [21] F.K. Sun, H. Chen, C.Y. Chen *et al.*, “A multithreaded initial detailed routing algorithm considering global routing guides,” in *Proc. ICCAD*, 2018.
- [22] L. McMurchie and C. Ebeling, “PathFinder: A Negotiation-Based Performance-Driven Router for FPGAs,” in *Proc. FPGA*, 1995.
- [23] R.M.F. Martins and N.C.C. Lourenço, “Analog integrated circuit routing techniques: An extensive review,” *IEEE Access*, vol. 11, pp. 35965–35983, 2023.
- [24] H.C. Ou, H.C.C. Chien, and Y.W. Chang, “Nonuniform multilevel analog routing with matching constraints,” *IEEE TCAD*, vol. 33, no. 12, pp. 1942–1954, 2014.
- [25] T. Yan and M.D.F. Wong, “Recent research development in pcb layout,” in *Proc. ICCAD*, 2010, pp. 398–403.
- [26] Q. Liu, Q. Tang, J. Chen *et al.*, “Disjoint-path and golden-pin based irregular pcb routing with complex constraints,” in *Proc. DAC*, 2023.
- [27] S.T. Lin, H.H. Wang, C.Y. Kuo *et al.*, “A complete pcb routing methodology with concurrent hierarchical routing,” in *Proc. DAC*, 2021.
- [28] M.H. Chung, J.W. Chuang, and Y.W. Chang, “Any-angle routing for redistribution layers in 2.5d ic packages,” in *Proc. DAC*, 2023.
- [29] C.A. Lee, W.H. Liu, G. Lin *et al.*, “Delay-matching routing for advanced packages,” in *Proc. ICCAD*, 2023.
- [30] L. Chrostowski, Z. Lu, J. Flückiger *et al.*, “Schematic driven silicon photonics design,” in *Smart Photonic and Optoelectronic Integrated Circuits XVIII*, vol. 9751. SPIE, 2016, pp. 9–22.
- [31] J. Matres *et al.*, “Gdsfactory,” <https://github.com/gdsfactory>, 2024.
- [32] W.H. Liu, W.C. Kao, Y.L. Li *et al.*, “Nctu-gr 2.0: Multithreaded collision-aware global routing with bounded-length maze routing,” *IEEE TCAD*, vol. 32, no. 5, pp. 709–722, 2013.
- [33] W.R. Clements, P.C. Humphreys, B.J. Metcalf *et al.*, “Optimal Design for Universal Multiport Interferometers,” *Optica*, 2018.
- [34] J. Gu, H. Zhu, C. Feng *et al.*, “ADEPT: Automatic Differentiable Design of Photonic Tensor Cores,” in *Proc. DAC*, 2022.
- [35] M. Zhang, D. Yin, N. Gangi *et al.*, “TeMPO: Efficient time-multiplexed dynamic photonic tensor core for edge AI with compact slow-light electro-optic modulator,” *J. Appl. Phys.*, vol. 135, no. 22, 2024.
- [36] Z. Zheng, M. Li, T.M. Tseng *et al.*, “Topro: A topology projector and waveguide router for wavelength-routed optical networks-on-chip,” in *Proc. ICCAD*. IEEE, 2021, pp. 1–9.
- [37] X. Tan, M. Yang, L. Zhang *et al.*, “On a scalable, non-blocking optical router for photonic networks-on-chip designs,” in *2011 Symposium on Photonics and Optoelectronics (SOPO)*. IEEE, 2011, pp. 1–4.
- [38] L. Qiao, W. Tang, and T. Chu, “32 × 32 silicon electro-optic switch with built-in monitors and balanced-status units,” *Scientific Reports*, vol. 7, no. 1, p. 42306, 2017.
- [39] H. Zhou, K. Zhu, and J. Gu, “LiDAR: Automated Curvy Waveguide Detailed Routing for Large-Scale Photonic Integrated Circuits,” in *Proc. ISPD*, 2025.
- [40] J. Chan, G. Hendry, A. Biberman *et al.*, “Architectural exploration of chip-scale photonic interconnection network designs using physical-layer analysis,” *Journal of Lightwave Technology*, vol. 28, no. 9, pp. 1305–1315, 2010.
- [41] D.P. Nair and M. Ménard, “A compact low-loss broadband polarization independent silicon 50/50 splitter,” *IEEE Photonics Journal*, vol. 13, no. 4, pp. 1–7, 2021.
- [42] S. Akiyama, T. Baba, M. Imai *et al.*, “12.5-Gb/s operation with 0.29-V-cm V()L using silicon Mach-Zehnder modulator based on forward-biased pin diode,” *Optics express*, vol. 20, no. 3, pp. 2911–2923, 2012.
- [43] M. Rakowski, C. Meagher, K. Nummy *et al.*, “45nm cmos-silicon photonics monolithic technology (45c1o) for next-generation, low power and high speed optical interconnects,” in *Optical Fiber Communication Conference*. Optica Publishing Group, 2020, pp. T3H–3.

