

ELITE: Efficient Lookup Table Assisted Routing Engine for Photonic Integrated Circuits

Xiaofei Yu^{1,2}, Yuchao Wu³, Haopeng Yan¹, Yeyu Tong^{1,2}, Yuzhe Ma^{1†}

¹Microelectronics Thrust, HKUST(GZ)

²Guangdong-Macao Joint Laboratory for Modular Chip Design and Testing, HKUST(GZ)

³Department of Electronic and Computer Engineering, HKUST

xyu082@connect.hkust-gz.edu.cn, yuzhema@hkust-gz.edu.cn

Abstract—Optical interconnects have emerged as a promising solution for rack-, board-scale, and even in-package communications, thanks to their high available optical bandwidth and minimal latency. However, optical waveguides are intrinsically different from traditional metal wires, especially in terms of phase-matching constraints, which pose new challenges for routing in photonic integrated circuit (PIC) design. In this paper, we present ELITE, a comprehensive and efficient lookup table (LUT)-assisted routing engine tailored for PICs with phase-matching constraints. We integrate a customized A*-based initial routing algorithm that accounts for potential waveguide crossings, a diffuse-based length-matching technique, and LUT-based strategies that leverage a pre-defined path library for bend reduction and strategic insertion to ensure precise phase alignment. The framework minimizes total insertion loss while adhering to all matching constraints. Experimental results based on real photonic designs show that our optical routing flow can reduce total insertion loss by 25% and maximum insertion loss by 46% compared to manual results, while satisfying matching constraints.

I. INTRODUCTION

As semiconductor technology advances toward the limit, the demand for reduced power consumption and diminished interconnection delay increases, thereby aggravating the challenges associated with transmission speed and energy efficiency. This predicament has sparked a growing interest in exploring optical interconnects as a promising alternative, which offers higher bandwidth, lower power consumption, and signal delay, compared to conventional electrical interconnects [1]–[4]. However, realizing these benefits at scale requires sophisticated photonics integrated circuit (PIC) design automation tools, which currently lag far behind mature electronic design automation. The PIC design flow often demands substantial expertise and manual effort, particularly for complex layouts and performance optimization [5], [6].

Within this landscape of design automation challenges, routing emerges as a critical bottleneck for achieving high-density, high-performance PICs. Critically, routing in the optical domain differs fundamentally from its electrical counterpart. Owing to the different natures of optical waveguides and electrical wires, PICs can be connected smoothly in any direction. In addition, the unique challenge of increased signal attenuation at sharp bends in photonic circuits necessitates the implementation of a minimum bend radius to maintain signal integrity. The optimization objective in photonic circuits also shifts from minimizing propagation length to reducing transmission loss. Furthermore, due to limitations in current fabrication technologies, optical routing is typically confined to a single dielectric layer, and waveguide crossings are allowed within the same layer to resolve topological conflicts [5].

Additional functional constraints are also imposed in optical routing. Many applications, including coherent communications, programmable photonic processors, and optical neural networks, necessitate that specific optical paths should be matched to ensure that optical waveguides have identical phase delays for the proper functionality of a circuit [7]. These factors considerably amplify the challenge of managing routing

constraints for waveguides on a chip while also striving to ensure minimal transmission loss. Consequently, traditional electrical routing algorithms can not be directly employed, and new methodologies are needed to tackle the unique routing challenges that minimize transmission loss while adhering to the phase-matching constraints.

Recent research has extensively focused on optical routing problems. Ding *et al.* [8] developed the first gridless optical routing framework based on integer linear programming (ILP) for low-power on-chip optical interconnections. They later optimized the framework by employing wavelength division multiplexing (WDM) to achieve high capacity and co-optimizing power and thermal reliability [9]. To improve performance further, Liu *et al.* [10] proposed a comprehensive routing flow for optical-electrical co-design and employed a Lagrangian relaxation (LR)-based algorithm to speed up the ILP solving. Notwithstanding the abundant studies, most existing studies only focus on external requirements like transmission loss and power while ignoring the internal limitations of matching constraints. Chuang *et al.* [11] first considered the matching constraints of waveguides and developed an ILP-based optical routing framework that optimizes the total transmission loss while satisfying all matching constraints. However, overly stringent requirements are imposed that all matched light paths be symmetrical or consistent, and ILP methods may not scale well.

Although PCB routing addresses length constraints through methods such as LR [12], LP-based grid allocation [13], and shape-based detouring [14], these approaches induce excessive bends via snaking patterns, making them inapplicable to optical routing. Moreover, minimizing bends is an NP-hard problem [15]. Current solutions, including the MILP formulation by Park *et al.* [15] and the edge labeling scheme by Lu *et al.* [16], face significant challenges regarding scalability and search efficiency due to the enlarged solution space. Since simultaneous optimization of path length and bend count is computationally prohibitive with these methods, an efficient bending minimization algorithm is urgently needed to mitigate performance degradation.

In this paper, we present a comprehensive and efficient optical routing engine, ELITE, that minimizes insertion loss while satisfying phase-matching constraints. Our approach begins with a global routing phase using a customized A* algorithm that directly optimizes insertion loss. To preserve routing space and reduce crossings, we incorporate a penalty term for potential crossings during this phase. For phase matching, we introduce a diffuse-based method that applies spiral detour patterns to phase-mismatched waveguides. This technique iteratively allocates sufficient area while ensuring detours achieve phase alignment with minimal additional bends, thereby controlling insertion loss. Finally, a lookup table (LUT)-based bend reduction techniques further reduce loss by minimizing the number of bends, with a strategic bend insertion on paths with fewer bends to align discrepancies across all nets. Our contributions are summarized as follows:

- We propose a comprehensive and efficient optical routing engine, ELITE, that minimizes insertion loss while satisfying phase-matching constraints.

[†]Corresponding author

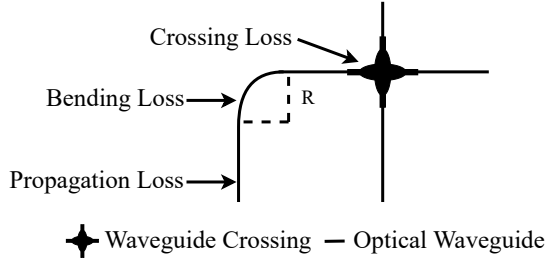


Fig. 1 Illustration of three major sources of loss for optical routing.

- We present a crossing-aware A* formulation that explicitly preserves routing space for each net, effectively reducing unnecessary net crossings and overall loss.
- We introduce a diffuse-based iterative length-matching method, effectively inserting detour patterns to ensure length matching.
- We propose a LUT-based bend reduction method, effectively reducing the extra bends to optimize the insertion loss.
- Experimental results based on real photonic designs show that our optical routing flow decreases 25% total insertion and 46% of max insertion loss while effectively satisfying matching constraints compared to manual results.

II. PRELIMINARY

In this section, we will first introduce the three major sources of insertion loss in optical routing addressed by this paper and then present our formulation of the optical routing problem.

A. Insertion Loss

The total insertion loss is primarily determined by three sources: propagation loss, bending loss, and crossing loss, as illustrated in Fig. 1. They can be defined as follows:

- 1) *Crossing Loss* (C_{loss}): The crossing loss occurs when two waveguides intersect. As shown in the experimental results of [17], the crossing loss can be minimized and remains constant when the angle between two waveguides is between 60 and 120 degrees.
- 2) *Bending Loss* (B_{loss}): The bending loss is induced when a waveguide is routed with a bend and can be calculated as follows:

$$B_{loss} = \sigma \cdot \theta \cdot r^{-\eta}, \quad (1)$$

where θ is the arc angle, r is the bending radius, and η and σ are two device specific parameters.

- 3) *Propagation Loss* (P_{loss}): The propagation loss occurs during waveguide transmission, typically on the order of 10^{-4} per micron. Due to its negligible scale, wire length is a trivial factor in optical routing, distinguishing it from the length-minimization objective in traditional electronic routing.

The total insertion loss (T_{loss}) in this paper is defined as follows:

$$T_{loss} = C_{loss} + B_{loss} + P_{loss}. \quad (2)$$

B. Problem Formulation

Here we formulate the optical routing problem under matching constraints as follows:

- Given the placement information, including the location and size of the components and the location of the pins on the components' boundaries, the design rules, and a netlist with matching net groups, the objective is to route all nets to minimize the total insertion loss without violating any design rules and to meet all matching constraints for all matching net groups.

III. PROPOSED METHOD

A. Overall Flow

Fig. 2 summarizes the flow of ELITE, which is divided into two main parts: (1) routing for matching net groups and (2) routing for general

nets without matching constraints. Each part is further subdivided into two stages, which are global routing and detailed routing.

For matching net groups, the global routing stage begins with square global grid construction to reduce the complexity of path searching and matching by restricting routes to Manhattan connections. An A*-based method is then applied to establish preliminary paths. Following the initial routing results, methods for length-matching and bend reduction are proposed to ensure compliance with matching constraints while minimizing insertion loss. In the detailed routing stage, refinement to the positions of pins and turning points while considering matching constraints is made to obtain the actual physical interconnections.

Similarly, for nets without matching constraints, the global routing stage starts with the rectangular global grid construction and initialization to set up the routing grid. Then, congestion-aware A*-based routing is conducted to construct paths with routability considerations, and a rip-up and reroute strategy is applied to nets with high loss. In the detailed routing stage, an extra A* search is performed on a finer grid defined by spacing and waveguide width, where nets are processed sequentially from the minimum to the maximum bounding boxes.

B. Routing for Matching Net Groups

1) A*-based Initial Routing

After constructing the global routing grid, we apply an A*-based routing algorithm to generate initial paths for matching nets with minimized loss. The routing cost is evaluated using total insertion loss as the objective function. As discussed in Section II-A, insertion loss depends on the waveguide length, as well as the number of bends and crossings. To estimate these components during routing, we use wire density as a proxy for predicting bends and potential crossings, while the minimum distance to the target is used to approximate path length. It can be defined as:

$$f(v) = g(v) + h(v), \quad (3a)$$

$$h(v) = C_{est}(v) + L_{est}(v), \quad (3b)$$

where v is the current position, $g(v)$ represents the current insertion loss of the path determined by Equation (2), $C_{est}(v)$ and $L_{est}(v)$ denote the estimated density and minimum length from v to the target, respectively.

We use a uniform function to estimate the wire density, thus speeding up the search process. This is computed as:

$$C_{est}(v) = \sum_{k=1}^N \frac{\alpha \cdot L_k(v, dst)}{Area(v, dst)}, \quad (4)$$

where N is the number of routed nets, $Area(v, dst)$ represents the area of the square constructed by the current point and target dst , and $L_k(v, dst)$ represents the length of net k in the area of the point and target. Fig. 3(a) illustrates the computation of the estimated routing density using a uniform density function. In this example, Net 3 (already routed) contributes to the estimated congestion for Net 2, which is currently being routed. At each grid cell, the algorithm constructs a bounding box between the current point (blue point) and the target of the net (shown by the dashed blue rectangle). The total length of routed segments from other nets (Net 3) that lie within this bounding box is used as an estimate of local density. This length is then normalized by the area of the bounding box to compute the density, under the assumption of uniform distribution of routing within the box. The resulting density is visualized using a color gradient: darker shades represent higher estimated density values.

And α is a user-defined parameter that balances the trade-off between the current loss preserving routing flexibility for subsequent nets, and a large α means the paths tend to detour and ensure a large distance from the existing waveguides. Fig. 3(b)–(d) illustrate the impact of different α values on routing behavior. In Fig. 3(b), where $\alpha = 0$, the density function is disabled, and congestion awareness is not considered during

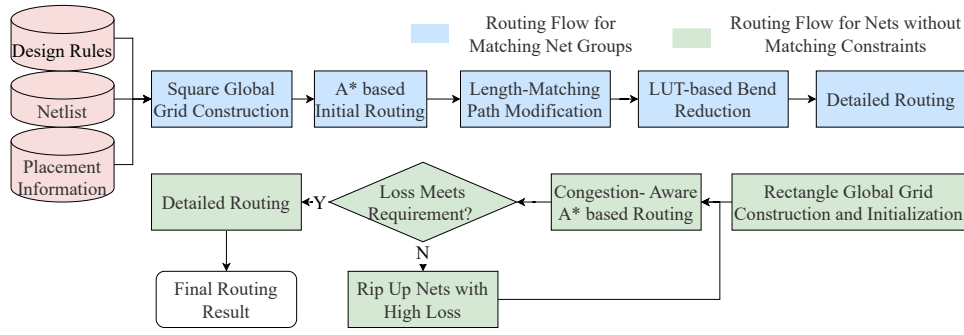


Fig. 2 The overview of our proposed routing flow.

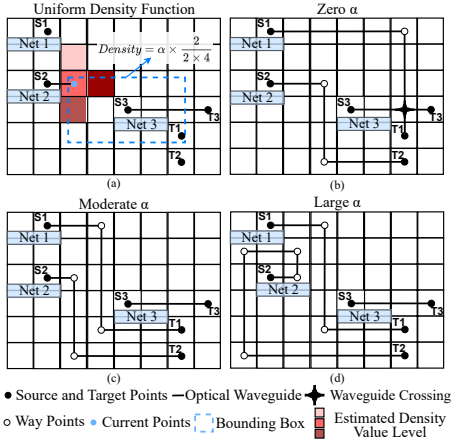


Fig. 3 Illustration of the uniform function and the effect of its weight. (a) The computation of the estimated density. (b)-(d) A* results with zero, medium, and large weight.

routing. As a result, Net 2 proceeds directly toward its target and closely follows the starting point of Net 3. This routing leaves insufficient space for Net 1, which then attempts to take the shortest path to its destination but becomes blocked by Net 2. Consequently, an unavoidable waveguide crossing occurs. In contrast, a moderate α value in Fig. 3(c) introduces enough deterrence to guide Net 2 slightly away from Net 3, leaving sufficient routing space for Net 1. This avoids the crossing entirely, without incurring additional path length. However, when α is set too high, as shown in Fig. 3(d), the algorithm becomes overly conservative. Net 1 is forced to route around the periphery of the layout to avoid moderately dense areas, resulting in a significant detour that increases the total waveguide length and, consequently, the insertion loss.

For routing order, nets with shorter half-perimeter wirelengths are prioritized. This heuristic is based on the observation that shorter nets generally have fewer detour options and are more sensitive to congestion. To ensure port alignment, routing starts in a fixed direction at the source while the target location is shifted by one grid cell. This adjustment guarantees that the final path properly connects both ports according to orientation constraints. Furthermore, explicit congestion modeling is omitted because bend radius and matching constraints already restrict each cell to one waveguide per direction.

2) Path Modification for Length-Matching

Following the A*-based initial routing, our algorithm addresses the length-matching challenges through a multi-step approach, which ensures efficient use of available routing resources and optimizes the detouring process to meet length-matching constraints.

Our algorithm begins by identifying the detour region within the initial routing result. The detour region is a closed area that satisfies the following conditions: (1) within this region, the paths maintain their relative order and there are no crossings; (2) all paths enter and exit through the same two sides, which, along with the outermost paths,

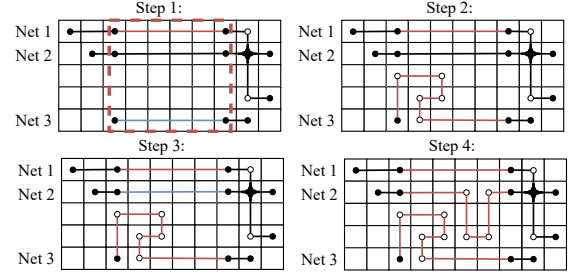


Fig. 4 Illustration of the path modification process for length-matching. The blue net represents the selected net for detouring, while the red nets indicate the fixed nets.

define the region's boundary. By traversing the paths, the maximum detour region is identified and serves as the basis for the subsequent routing process. This concept is illustrated in the Fig. 4, where the red dashed lines outline the detour region.

After assigning the detour region, we can assess the available area within the region to ensure sufficient space for the detour. This involves calculating the number of unused grids and comparing it to the required detouring space, which is determined by the difference between the current path length and the group's longest path, summed over all nets. To account for *wasted cells* as in [12], a slack variable δ is introduced. The space is considered sufficient if the available area is at least $(1 + \delta)$ times greater than the needed area. $\delta = 0.03$ is enough in our practice. If the initial assessment indicates insufficient detouring space, additional routing resources will be allocated by extending the longest edge on the boundary of the detour region.

With sufficient routing resources ensured, the algorithm proceeds with detouring process. If abundant space exists between adjacent paths, the process is straightforward. However, when the available area is limited, we employ the *diffuse-based iterative detour method* to maximize area utilization for detours and meet length-matching constraints. We begin by selecting the two outermost paths within the detour region based on the boundary and specifying one path as the candidate for detouring. Once a path is selected as a candidate, the remaining paths, excluding fixed paths and the candidate, are diffused along the outer boundary to maximize the available space for detouring. Fig. 4 demonstrates the procedure. In Step 1, we select the lower net (Net 3) as the candidate for detouring, while the other net (Net 2) is diffused along the upper boundary (Net 1). Then in Step 2, to minimize the number of bends required during detouring, we then apply a spiral detour pattern to the candidate (Net 3), where the routing segments extend from the source and bend back in a U-shape before continuing toward the destination. This pattern helps reduce complexity and potential signal degradation caused by excessive bending. The spiral detour pattern is applied iteratively to the longest segment of the selected net until the candidate reaches the desired length. Once the net length is matched, it is fixed as a boundary. Next, we repeat

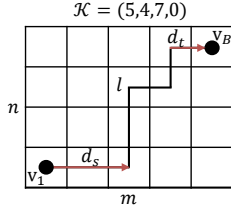


Fig. 5 Illustration of some notations.

the process for the other nets in the previously selected pair. In Step 3 and Step 4, other paths are diffused along the lower boundary (Net 3), and the spiral detour pattern is applied to the new candidate (Net 2). This process is performed iteratively until all nets are fixed in place, ensuring that all the nets have identical lengths.

C. Path Library for Bend Reduction

While the initial routing phase establishes the basic connectivity and length matching, the resulting paths often exhibit extra bends that increase insertion loss. To address this, we propose a LUT-based path library. This library enables efficient refinement by replacing existing path segments with pre-calculated, bend-optimized alternatives without altering the total path length.

1) Path Library Definition

We define the path library $\mathcal{L}$ as a collection of key-value pairs $\langle \mathcal{K}_i, \mathbf{V}_i \rangle$. The key $\mathcal{K}_i$ encodes the geometric characteristics of the initial path in the i -th scenario, mapping to a value $\mathbf{V}_i$ that contains a set of optimized paths $\{\mathbf{P}_a\}_{a=1}^A$ sorted by ascending bend count. Each $\mathbf{P}_a$ is defined as a vertex sequence $\{v_1, \dots, v_{B_a}\}$ of length B_a . For any vertex $v_b \in \mathbf{P}_a$, the component $v_b = (x_b, y_b) \in \mathbb{R}^2$ denotes the coordinates of the b -th vertex in the sequence.

The key $\mathcal{K}$ is formally defined as $\langle v_1, v_B, m, n, l, d_s, d_t \rangle$. Here, v_1 and v_B denote the start and end points of the path to be optimized. The parameters m and n represent the dimensions of the minimum bounding box spanning v_1 and v_B calculated as $m = |x_1 - x_B|$ and $n = |y_1 - y_B|$. The variable l corresponds to the total Manhattan length of the path, given by $l = \sum_{k=1}^{B-1} (|x_k - x_{k+1}| + |y_k - y_{k+1}|)$. Furthermore, $d_s = d(1)$ and $d_t = d(B-1)$ specify the waveguide orientations at the source and terminal points, respectively, where the direction d between two nodes v_i and v_{i+1} is encoded as 0, 1, 2, or 3, representing Right, Up, Left, and Down, respectively.

Note that to break geometric symmetry and reduce library redundancy, we transform each path into a canonical form by normalizing the coordinates such that the source v_1 is at $(0, 0)$ and the target v_B is at (m, n) , while rotating the path so the source direction d_s is always 0. Consequently, the geometric signature simplifies to the four-tuple:

$$\mathcal{K} = (m, n, l, d_t). \quad (5)$$

Associated with the geometric feature $\mathcal{K}_i$ is the path set $\mathbf{V}_i = \{\mathbf{P}_1, \dots, \mathbf{P}_A\}$. We partition and sort this set ascendingly into distinct clusters according to the number of bends in each path.

We refer to this set as a candidate bucket. Each bucket provides a pre-computed mapping from the geometric signature to its corresponding alternatives. This indexing scheme allows the path insertion to perform an $O(1)$ query to retrieve a set of alternative paths that satisfy identical geometric and length constraints while offering a reduced bend count.

2) Path Library Generation

Due to the exponential growth of paths in obstacle-free regions, exhaustive enumeration for LUT construction is computationally prohibitive. Inspired by Line Probe Routing, we propose a randomized search algorithm that alternately samples actions from $\pi_h = \{\leftarrow, \rightarrow\} \times \{1, \dots, m\}$ horizontally and $\pi_v = \{\uparrow, \downarrow\} \times \{1, \dots, n\}$ vertically sets as Fig. 6(a). The search depth is set to $D = BC + 1$, iteratively incrementing the bend count BC . By incorporating pruning

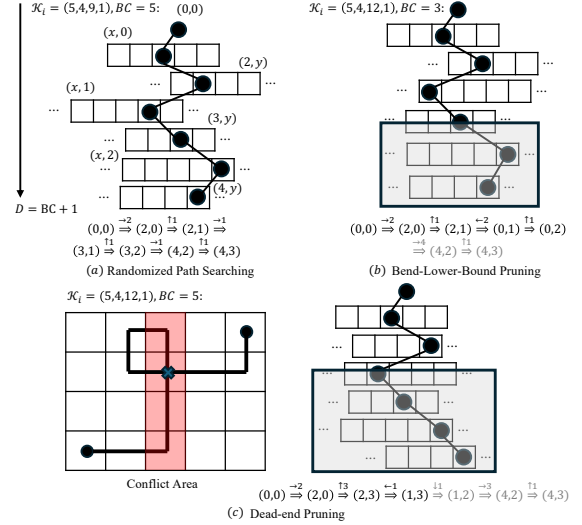


Fig. 6 Overview of LUT Generation. (a) Randomly sample the possible paths according to $\mathcal{K}_i$ and bend count BC with alternate $\pi_h = \{\leftarrow, \rightarrow\} \times \{1, \dots, m\}$ and $\pi_v = \{\uparrow, \downarrow\} \times \{1, \dots, n\}$ action. (b) Bend-Lower-Bound Pruning. (c) Dead-end Pruning.

strategies illustrated in Fig. 6(b) and Fig. 6(c), valid paths satisfying $\mathcal{K}_i$ are efficiently collected into $\mathbf{V}_i$, balancing solution diversity with computational cost. Bend-Lower-Bound Pruning terminates a branch if the current bend count, plus the mandatory turns predicted to reach T (e.g., if $x \neq m$ and $y \neq n$), exceeds BC . Simultaneously, Dead-end Pruning utilizes a visited bitmask to detect connectivity. If the current path forms a self-intersecting loop or isolates the target T within the $m \times n$ bounding box, the search backtracks immediately. Beyond search efficiency, the utility of the LUT is constrained by the trade-off between memory overhead and topological diversity. Considering this, we introduce a length threshold to discard paths exceeding twice the Manhattan distance ($l > 2(m + n)$).

3) Path Matching and Insertion

Following the construction of the path library, the subsequent challenge is to apply these pre-calculated paths to optimize paths with excessive bend counts. However, direct lookup is often hindered by the substantial dimensions of paths, which exceed the pre-defined grid scales of the path library. To bridge this gap, we propose a multi-level bend reduction strategy, as outlined in Algorithm 1.

In each iteration, the algorithm selects the path with the most bends $\mathbf{p} \in \mathbf{R}$ (Step 4) and searches for a replaceable subpath $\mathbf{s}$ (Step 5), where $\mathbf{R}$ is the set of routed paths on the design. Specifically, a subpath is considered replaceable only if it is *box-contained*, meaning that its trajectory stays within the bounding box induced by its two endpoints. To cover both local and moderately long patterns, we enumerate candidate subpaths across multiple levels up to Z . A level-1 subpath $\mathbf{s}$ is defined as a maximal segment along $\mathbf{p}$, obtained by expanding from a starting vertex to the furthest possible vertex such that the box-contained property is preserved. Higher-level candidates are then constructed through concatenation, where a level- z subpath is formed by merging z consecutive level-1 segments, provided the resulting trajectory remains box-contained.

For each candidate $\mathbf{s}$, we normalize it into a canonical key (Step 6) by translating its start point to the origin, rotating it to ensure $d_s = 0$, and applying a mirror reflection if the terminal y -coordinate is negative. This transformation yields a standardized key (m, n, l, d_t) to query $\mathcal{L}$ for a set of alternatives $\mathbf{C}$ with fewer bends (Step 7). Each alternative is then denormalized (Step 9) by reversing these geometric operations to obtain a concrete replacement $\mathbf{q}'$. To ensure feasibility, NONCONFLICTING

Algorithm 1: Imp-LUT($\mathbf{R}, \mathcal{L}, Z$)**Input:** Path set $\mathbf{R}$, LUT $\mathcal{L}$, max level Z **Output:** Optimized path set $\mathbf{R}$

```

1 updated  $\leftarrow$  true;
2 while updated do
3   updated  $\leftarrow$  false;
4    $\mathbf{p} \leftarrow \text{SELECTWORST}(\mathbf{R})$ ;
5   foreach  $\mathbf{s} \in \text{ENUMERATESUBPATHS}(\mathbf{p}, Z)$  do
6     key  $\leftarrow \text{NORMALIZETOKEY}(\mathbf{s})$ ;
7      $\mathbf{C} \leftarrow \text{QUERYBETTER}(\mathcal{L}, \textit{key}, \text{BENDS}(\mathbf{s}))$ ;
8     foreach  $\mathbf{q} \in \mathbf{C}$  do
9        $\mathbf{q}' \leftarrow \text{DENORMALIZE}(\mathbf{s}, \mathbf{q})$ ;
10      if  $\text{NONCONFLICTING}(\mathbf{R}, \mathbf{p}, \mathbf{s}, \mathbf{q}')$  then
11         $\mathbf{p}' \leftarrow \mathbf{p}[\mathbf{s} \rightarrow \mathbf{q}']$ ;
12         $\mathbf{R} \leftarrow (\mathbf{R} \setminus \{\mathbf{p}\}) \cup \{\mathbf{p}'\}$ ;
13        updated  $\leftarrow$  true;
14        break;
15    if updated then
16      break;
17 return  $\mathbf{R}$ ;

```

(Step 10) temporarily clears the occupancy of $\mathbf{s}$ and verifies that $\mathbf{q}'$ does not collide with the rest of the path set, which comprises all other paths in $\mathbf{R}$ and the segments of $\mathbf{p}$ not covered by $\mathbf{s}$. If $\mathbf{q}'$ is feasible, $\mathbf{s}$ is replaced and $\mathbf{R}$ is updated (Steps 11–12). The process repeats until no further improvements are found.

D. General Nets Routing**1) Congestion-Aware A*-based Routing**

Unlike the approach in Section III-B, which only permitted Manhattan connections, diagonal connections are allowed in this section. The capacity of each edge is set to be the width divided by the minimum waveguide spacing plus the waveguide width. For Manhattan connections, the routing demand is set to 1 per net. For non-Manhattan connections, we distribute the routing demand across the four intersecting edges evenly, with each edge allocated one-quarter of the demand.

Building upon the previous A* formulation used for matching net groups routing, we introduce a congestion-aware A* formulation to the routing for general nets without constraints. This formulation incorporates a congestion penalty term p_e into the cost function to account for potential overflow, which helps to distribute the routing demand more evenly and avoid congestion. Different from the previous approaches in Section III-B that restricted diagonal movement, the current formulation allows for diagonal routing by considering eight directions in each A* search. The cost function can be defined as:

$$f(\mathbf{v}) = c(\mathbf{v}) + g(\mathbf{v}') + h(\mathbf{v}), \quad (6)$$

where $\mathbf{v}'$ represents the previous node of $\mathbf{v}$ and $c(\mathbf{v})$ is the cost of the current node, which can be computed as:

$$c(\mathbf{v}) = (1 + p_e) \times (g(\mathbf{v}) - g(\mathbf{v}')). \quad (7)$$

Assuming C_e is an edge's capacity, D_e is an edge's demand. We define ω_e as the availability of the edge: $\omega_e = (C_e - D_e)/C_e$. The congestion penalty term p_e is computed as follows:

$$p_e = \begin{cases} \frac{\beta}{1 + e^{t \cdot \omega_e}} \times (1 - \omega_e), & \text{if } \omega_e > 0 \\ \infty, & \text{otherwise} \end{cases} \quad (8)$$

where β and t are user-defined parameters. β determines the influence of congestion, and t represents the sensitivity to congestion. In this paper, β and t are set to be 1.5 and $\ln 5$, respectively. We choose the logistic function [18] to ensure that the congestion penalty is near zero

when resources are abundant. As the available resources decrease, the penalty increases rapidly, reaching its highest slope at the capacity point. By multiplying the demand term, we ensure the penalty is zero when no demand exists. Note that in current mainstream silicon photonic designs, only one layer is available and the overflow is strictly prohibited, so we set a very large constant as the penalty to avoid this situation.

Taking into account the minimum bending radius constraints, we introduce a bending factor γ which is defined as follows:

$$\gamma = \begin{cases} 1/(\epsilon(\text{dist}(\mathbf{v}_i, \mathbf{v}_{i+1}) - r_{\min})), & \text{if } i = 1, \\ 1/(\epsilon(\text{dist}(\mathbf{v}_i, \mathbf{v}_{i+1}) - 2 \times r_{\min})), & \text{otherwise,} \end{cases} \quad (9)$$

where i is the sequence of the turning point in the path from the source, $\text{dist}(\mathbf{v}_i, \mathbf{v}_{i+1})$ is the distance between the turning point $\mathbf{v}_i$ and $\mathbf{v}_{i+1}$, and $\epsilon(x)$ is the Heaviside function, which is 1 if $x \geq 0$ and 0 otherwise. If the bending radius satisfies the minimum bending radius constraint, this factor will be 1 and will not affect the $g(\mathbf{v})$. Otherwise, it will be infinite, effectively avoiding small bending radius.

To handle the loss dynamically, we apply a rip-up and re-route (RRR) strategy. First, we check if the total and maximum losses meet our requirements. If the total loss is too high, we identify the grid with the maximum loss, which typically corresponds to the region with the most crossings. We then rip up the nets passing through the identified high-loss region and reroute them. If the maximum loss requirement for the nets is not met, we rip up the net with the maximum loss and the intersecting nets. The order of rerouting is arbitrary, but nets with a higher frequency of rip-ups are prioritized in the rerouting process.

A negotiated congestion algorithm [21] is applied to manage the over-lossy grid in each iteration. The cost for the current grid $c(\mathbf{v})$ is modified as:

$$c(\mathbf{v}) = (1 + h(\mathbf{v}) \times o(\mathbf{v})) \times (g(\mathbf{v}) - g(\mathbf{v}')), \quad (10)$$

where $h(\mathbf{v})$ is the added cost reflecting the history over-lossy, and $o(\mathbf{v})$ is the penalty for the lossy grid. Similar to the Equation (8), we replace the relative resource of the edge ω_e with the over-lossy ratio ω_v , which is defined as the ratio of excess to the maximum threshold:

$$\omega_v = \begin{cases} (L_{\text{total}} - L_{th})/L_{th}, & \text{if total loss is unqualified,} \\ (L_{\text{max_net}} - L_{th})/L_{th}, & \text{if max loss is unqualified.} \end{cases} \quad (11)$$

And $o(\mathbf{v})$ can be modified as follows:

$$o(\mathbf{v}) = \frac{\lambda}{1 + e^{t \cdot \omega_v}}, \quad (12)$$

in which we omitted the demand term here because the ω_v is always larger than 0, and we do not want the penalty to increase linearly but rather stabilize when the loss is much higher than the threshold. The parameter λ is a scaling factor that is set to 0.02 in our paper. The history cost $h(\mathbf{v})$ is defined as follows:

$$h(\mathbf{v})^{k+1} = \begin{cases} h(\mathbf{v})^k + h_{inc}, & \text{if loss is unqualified,} \\ h(\mathbf{v})^k, & \text{Otherwise,} \end{cases} \quad (13)$$

where $h_v^0 = 0$, and h_{inc} is set to 1 to affect the convergence speed.

IV. EXPERIMENTAL RESULTS

In this section, we present our experimental results. We implement ELITE in C++ and export the final results into GDS file in Python with Nazca library. All the experiments are conducted on a computer with an AMD Ryzen 9 8945HS CPU and 32GB of memory. The performance metrics of circuit layouts produced by our optical routing framework are evaluated in comparison with the other three baselines, including the BSG-route [13], ILP [11], and the designs from experienced designers, with all methods executed under a time limitation of 20000 seconds. For a fair comparison, we implemented the BSG-route using the same A*-based initial routing topology. To evaluate the effectiveness

TABLE I Benchmark information of the die size(μm^2), net number, and routing grid number. Comparison of the total insertion loss TL(dB), max insertion loss ML(dB), max length difference MLD(μm), max bends difference MBD, and runtime(s).

Benchmark				Manual				BSG_route [13]					ILP [11]					ELITE				
Name	Size(μm^2)	# Net	# Grids	TL	ML	MLD	# MBD	TL	ML	MLD	# MBD	Runtime	TL	ML	MLD	# MBD	Runtime	TL	ML	MLD	# MBD	Runtime
Optical_Phase_Array_4x4 [19]	250x4300	4	750	0.948	0.237	0	0	1.056	0.354	0	8	0.62	0.690	0.1725	0	0	1512.59	0.690	0.1725	0	0	0.03
Optical_Phase_Array_9x9 [19]	450x350	9	1575	7.452	0.828	0	0	5.358	0.822	0	20	1.63	-	-	-	-	-	3.078	0.342	0	0	0.14
Optical_Phase_Array_16x16 [19]	800x1300	16	10400	21.504	1.344	0	0	19.998	1.908	0	52	9.82	-	-	-	-	-	8.448	0.528	0	0	6.06
Optical_Delay_Lines_6x6	100x150	6	150	1.380	0.390	0	12	1.560	0.390	0	12	0.03	1.674	0.279	0	0	16243.95	1.914	0.339	0	4	0.02
Optical_Delay_Lines_8x8	1450x250	8	3625	3.612	0.669	0	16	5.538	0.838	0	20	4.83	-	-	-	-	-	2.760	0.345	0	0	4.06
Waveguide_Crossing_Array_8x8	600x600	8	3600	5.163	1.833	240	6	7.116	1.842	0	52	3.51	-	-	-	-	-	4.596	0.582	0	2	1.12
Waveguide_Crossing_Array_16x16	1000x2000	16	20000	22.118	4.377	1001.075	16	41.148	4.045	0	91	87.72	-	-	-	-	-	22.488	1.435	0	2	42.15
Bundled_Bending_Array_8x8_A [20]	400x1250	8	5000	3.360	0.420	0	0	4.860	0.9525	0	26	1.74	-	-	-	-	-	2.340	0.2925	0	0	0.20
Bundled_Bending_Array_8x8_B	1800x1450	8	26100	5.772	0.879	0	0	13.512	2.319	0	62	58.76	-	-	-	-	-	5.592	0.699	0	0	53.97
Bundled_Bending_Array_14x14 [20]	1600x700	14	11200	10.332	0.738	0	0	15.423	2.5845	0	78	5.33	-	-	-	-	-	7.623	0.5445	0	0	0.55
Geo-mean				5.30	0.83	-	-	7.02	1.23	-	-	3.50	-	-	-	-	-	3.97	0.45	-	-	0.95
Ratio				1.33	1.84	-	-	1.77	2.73	-	-	3.69	-	-	-	-	-	1.00	1.00	-	-	1.00

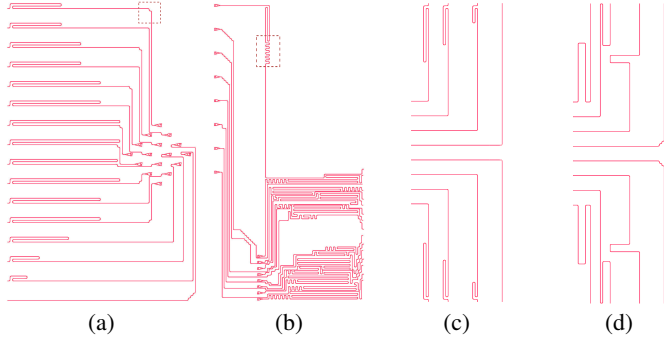


Fig. 7 Part of results. (a) Optical_Phase_Array_16x16, (b) Waveguide_Crossing_Array_16x16, (c) Bundled_Bending_Array_8x8_A without path library, (d) Bundled_Bending_Array_8x8_A with path library.

and scalability of ELITE, we selected ten real photonic designs with different sizes and numbers of nets, as illustrated in TABLE I.

In these benchmarks, the insertion loss settings are: 0.52 dB per crossing, 1.5 dB per centimeter, 0.03 dB per 90-degree bend, and 0.015 dB per 45-degree bend, based on a PDK from an industrial foundry. Due to limitations in the PDK, which only provides 90-degree waveguide crossing devices, we adopt a square grid for all routing. If additional crossing angles are supported, our method can be easily extended by adjusting the grid geometry or aspect ratio accordingly, thereby enabling a broader and more flexible design space.

For the settings of ELITE, we empirically set the parameter α in Equation (4) to 0.3 and Z in Algorithm 1 to 10. The path library is pre-generated by exploring the configuration space (m, n, D) , where the grid dimensions $m, n \in [0, 60]$ and the search depth $D \in \{1, 2, \dots, 7\}$. For each (m, n, D) , 500 sampling iterations are conducted. This pre-computation process requires approximately 30.7 hours, producing a raw LUT of 286 MB. After redundancy removal, the final LUT size is reduced to 195 MB.

TABLE I shows the comparison between routing results. As shown in the table, ELITE satisfies all the length-matching constraints and only exhibits a few mismatches in the number of bends. Compared to manual designs, our results achieve a 25% reduction in total loss and a 46% reduction in maximum loss. Additionally, BSG-route exhibits the highest insertion loss and a significant discrepancy in maximum bends due to its tendency to generate excessive bends, whereas our method achieves a 73% reduction in computation time. In contrast, ILP achieves optimal maximum loss but demonstrates poor scalability and the highest computation times, completing only a limited set of small cases. This is evident when the net count scales from 4 to 6: despite a reduced grid size, runtime increases from 1512s to 16243s.

The resulting layout is shown in Fig. 7. To satisfy the matching constraints, a heuristic strategy is applied after the bend reduction

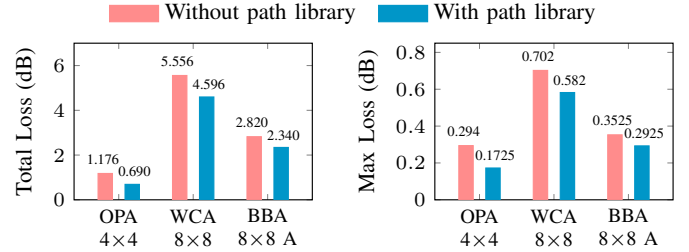


Fig. 8 Ablation study of path library. OPA is Optical Phase Array, WCA is Waveguide Crossing Array, BBA is Bundled Bending Array.

stage. For nets with fewer bends, bends are inserted while maintaining a constant path length. Specifically, as shown in Fig. 7(a), bends are inserted into existing L-shaped segments. In cases with detour patterns, such as in Fig. 7(b), U-shaped bends are used to increase the bend count. This approach ensures that nets in a matching group achieve identical bend counts without violating the length-matching constraint.

Additionally, we evaluate the advantages of the proposed path library. As illustrated in Fig. 8, its integration significantly reduces both total and maximum insertion losses, achieving an average 25% improvement. For Optical_Phase_Array_4x4, our method with the path library yields results identical to those of the ILP method, underscoring its effectiveness. Fig. 7(c) and (d) further compare the layouts of Bundled_Bending_Array_8x8_A without and with the path library, respectively. The detour pattern in (c) is optimized in (d), reducing the number of bends from 7 to 5 while maintaining the same path length. Note that all improvements are achieved with minimal computational overhead, as the library query time remains under 1 ms in all cases.

V. CONCLUSION

In this paper, we proposed ELITE, a comprehensive and efficient LUT-assisted routing engine tailored for PICs with phase-matching constraints. The framework integrates a crossing-aware A*-based routing algorithm that accounts for insertion loss and potential waveguide crossings, enabling high-quality initial path generation. The proposed diffuse-based length-matching method effectively maximizes area utilization for our spiral detour pattern insertion, ensuring length-matching constraints are met with minimal bend overhead. The LUT-based bend reduction method efficiently reduces the number of bends without changing the path length. Experimental results on real photonic designs show that our method decreases total insertion loss by 25% and maximum insertion loss by 46% while effectively satisfying matching constraints compared to manual results and completing the routing in seconds.

ACKNOWLEDGMENT

This work is supported in part by the Nansha District Key Area S&T Scheme (No. 2024ZD007) and the Guangdong Science and Technology Department (No. 2025B1212150003).

REFERENCES

- [1] C. Sun, M. T. Wade, Y. Lee, J. S. Orcutt, L. Alloatti, M. S. Georgas, A. S. Waterman, J. M. Shainline, R. R. Avizienis, S. Lin *et al.*, “Single-chip microprocessor that communicates directly using light,” *Nature*, vol. 528, no. 7583, pp. 534–538, 2015.
- [2] K.-i. Kitayama, M. Notomi, M. Naruse, K. Inoue, S. Kawakami, and A. Uchida, “Novel frontier of photonics for data processing—photonic accelerator,” *Apl Photonics*, vol. 4, no. 9, 2019.
- [3] K. Lu, Z. Chen, H. Chen, W. Zhou, Z. Zhang, H. K. Tsang, and Y. Tong, “Empowering high-dimensional optical fiber communications with integrated photonic processors,” *Nature Communications*, vol. 15, no. 1, p. 3515, 2024.
- [4] Z. Xu, T. Zhou, M. Ma, C. Deng, Q. Dai, and L. Fang, “Large-scale photonic chiplet taichi empowers 160-tops/w artificial general intelligence,” *Science*, vol. 384, no. 6692, pp. 202–209, 2024.
- [5] W. Bogaerts and L. Chrostowski, “Silicon photonics circuit design: methods, tools and challenges,” *Laser & Photonics Reviews*, vol. 12, no. 4, p. 1700237, 2018.
- [6] Y. Wu, X. Yu, H. Chen, Y. Luo, Y. Tong, and Y. Ma, “PICBench: Benchmarking LLMs for photonic integrated circuits design,” *2025 Design, Automation & Test in Europe Conference (DATE)*, pp. 1–6, 2025.
- [7] G. Zuo, W. Li, Z. Yang, S. Li, R. Qi, Y. Huang, and L. Xia, “Double phase matching in mzi with antiresonant effect for optical fiber sensor application,” *Journal of Lightwave Technology*, vol. 39, no. 2, pp. 660–666, 2021.
- [8] D. Ding, Y. Zhang, H. Huang, R. T. Chen, and D. Z. Pan, “O-router: an optical routing framework for low power on-chip silicon nano-photonic integration,” in *Proceedings of the 46th annual design automation conference*, 2009, pp. 264–269.
- [9] D. Ding, B. Yu, and D. Z. Pan, “Glow: A global router for low-power thermal-reliable interconnect synthesis using photonic wavelength multiplexing,” in *17th Asia and South Pacific Design Automation Conference*. IEEE, 2012, pp. 621–626.
- [10] D. Liu, Z. Zhao, Z. Wang, Z. Ying, R. T. Chen, and D. Z. Pan, “Operon: optical-electrical power-efficient route synthesis for on-chip signals,” in *Proceedings of the 55th Annual Design Automation Conference*, 2018, pp. 1–6.
- [11] F.-Y. Chuang and Y.-W. Chang, “On-chip optical routing with waveguide matching constraints,” in *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*. IEEE, 2021, pp. 1–6.
- [12] M. M. Ozdal and M. D. Wong, “A length-matching routing algorithm for high-performance printed circuit boards,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, no. 12, pp. 2784–2794, 2006.
- [13] T. Yan and M. D. Wong, “Bsg-route: A length-matching router for general topology,” in *2008 IEEE/ACM International Conference on Computer-Aided Design*. IEEE, 2008, pp. 499–505.
- [14] C.-A. Lee, W.-H. Liu, G. Lin, and T.-Y. Ho, “Delay-matching routing for advanced packages,” in *2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*. IEEE, 2023, pp. 1–8.
- [15] H. Park, B.-I. Kim, D. G. Choi, H. Kyu Kim, T. Wook Kang, and Y. Jung Lee, “Layer and length-deviation limit aware interposer routing for bend and wirelength minimization,” *IEEE Transactions on Components, Packaging and Manufacturing Technology*, vol. 14, no. 6, pp. 993–1006, 2024.
- [16] J. Lu and C.-W. Sham, “Lmgr: A low-memory global router with dynamic topology update and bending-aware optimum path search,” in *International Symposium on Quality Electronic Design (ISQED)*, 2013, pp. 231–238.
- [17] P. Sanchis, J. Galan, A. Griol, J. Marti, M. Piqueras, and J. Perdigues, “Low-crosstalk in silicon-on-insulator waveguide crossings with optimized-angle,” *IEEE Photonics Technology Letters*, vol. 19, no. 20, pp. 1583–1585, 2007.
- [18] M. Pan and C. Chu, “Fastroute: A step to integrate global routing into placement,” in *Proceedings of the 2006 IEEE/ACM international conference on Computer-aided design*, 2006, pp. 464–471.
- [19] J. Büttow, J. S. Eismann, V. Sharma, D. Brandmüller, and P. Banzer, “Generating free-space structured light with programmable integrated photonics,” *Nature Photonics*, vol. 18, no. 3, pp. 243–249, 2024.
- [20] J. Huang, J. Mao, X. Li, J. Yuan, Y. Zheng, C. Zhai, T. Dai, Z. Fu, J. Bao, Y. Yang *et al.*, “Integrated optical entangled quantum vortex emitters,” *Nature Photonics*, pp. 1–8, 2025.
- [21] L. McMurchie and C. Ebeling, “Pathfinder: A negotiation-based performance-driven router for fpgas,” in *Proceedings of the 1995 ACM third international symposium on Field-programmable gate arrays*, 1995, pp. 111–117.