

Apollo: Automated Routing-Informed Placement for Large-Scale Photonic Integrated Circuits

Hongjian Zhou¹, Haoyu Yang³, Nicholas Gangi², Zhaoran (Rena) Huang², Haoxing Ren³, Jiaqi Gu^{1,†}

¹Arizona State University ²Rensselaer Polytechnic Institute ³NVIDIA Corporation

†jiaqigu@asu.edu

Abstract

As technology advances, photonic integrated circuits (PICs) are rapidly scaling in size and complexity, with modern designs integrating thousands of components to meet the demands of artificial intelligence (AI), high-performance computing, and chip-to-chip optical interconnects. However, the analog custom layout nature of photonics, the curvy waveguide structures, and single-layer routing resources impose stringent physical constraints, such as minimum bend radii and waveguide crossing penalties, which make manual layout the *de facto* standard. This manual process takes weeks to complete and is error-prone, which is fundamentally unscalable for large-scale PIC systems. Existing automation solutions have adopted force-directed placement on small benchmarks with tens of components, with limited routability and scalability. To fill this fundamental gap in the electronic-photonic design automation (EPDA) toolchain, we present Apollo, the first GPU-accelerated, routing-informed placement framework tailored for large-scale PICs. Apollo features an asymmetric bending-aware wirelength function with explicit modeling of waveguide routing congestion and crossings to preserve enough routing spacing for routability maximization. Meanwhile, conditional projection is employed to gradually enforce a variety of user-defined layout constraints, including alignment, spacing, etc. This constrained optimization is accelerated and stabilized by a custom blockwise adaptive Nesterov-accelerated optimizer, ensuring stable and high-quality convergence. To catalyze research in PIC layout automation, we also develop and open-source large-scale PIC benchmarks derived from real-world photonic tensor core designs. Compared to existing methods, Apollo can generate high-quality layouts for large-scale PICs with an average routing success rate of 94.79% across all benchmarks within minutes. By tightly coupling placement with physical-aware routing, Apollo establishes a new paradigm for automated PIC design—bringing intelligent, scalable layout synthesis to the forefront of next-generation EPDA. Our code is open-sourced at [link](https://github.com/ScopeX-ASU/Apollo)*

1 Introduction

Integrated photonics has emerged as a transformative platform for high-performance computing [17–19, 27] and communication systems [21, 24]. By manipulating and processing light signals on-chip through optical components, photonic integrated circuits (PICs) offer unparalleled advantages in speed, parallelism, and energy efficiency. These properties make PICs particularly attractive for a wide range of applications, from chip-to-chip optical interconnects [3, 20, 21, 24, 25] and data center networking, to artificial intelligence (AI) acceleration [17–19], quantum computing, and optical signal processing.

Despite their immense potential, the physical layout of PICs remains largely a manual and iterative process, even in state-of-the-art commercial and academic design flows [13]. Unlike digital VLSI, where placement and routing are decoupled and highly automated, PIC layout presents fundamentally different challenges due to the analog nature of light propagation and the unique physical constraints

needed to maintain signal integrity, performance, and fabrication yield. In PICs, we claim that waveguide routing must be tightly integrated into the placement process, far more so than in digital VLSI, because the feasibility and quality of routing are highly sensitive to component placement. At the core of these challenges lies the fact that photonic waveguides are directional, curvilinear, and highly space-consuming [28]. Unlike metal wires in VLSI circuits or on PCBs, waveguides must conform to minimum bend radii to avoid excessive propagation loss and cannot intersect with photonic devices. Waveguide crossings, analogous to vias in VLSI, must also be carefully planned to minimize crosstalk, insertion loss, and excessive detours, all of which are tightly influenced by placement. Crossings further introduce non-negligible area overhead and impose strict port orientation requirements, complicating the layout even more. Moreover, PICs are typically restricted to a single layer (or a few for advanced PICs), resulting in severe layout resource contention. These properties make waveguide routing a dominant constraint during placement. Furthermore, PICs often involve mixed-size components, ranging from compact $2 \times 2 \mu\text{m}^2$ devices (e.g., micro-disk modulators and y-branches) to large $100 \times 1000 \mu\text{m}^2$ devices (e.g., Mach-Zehnder modulators, MZM). Some devices exhibit extreme aspect ratios (e.g., 1:20), posing additional challenges for placement that must handle geometric diversity, routability, and photonic-specific layout rules.

Due to these complex requirements, current commercial PIC layout tools often rely on schematic-driven layout practices. Designers manually translate schematics into physical layouts, carefully placing components and routing waveguides to ensure correct port alignment, sufficient spacing between components to reduce crosstalk, and reserving enough room to accommodate bends and crossings. While this manual methodology can yield valid designs, it is highly time-consuming and cannot scale to large systems with thousands of photonic components.

Prior works have attempted to address PIC placement through force-directed approaches [4, 23]. These methods are fully automated and generate placements without leveraging human design expertise or schematic guidance, resulting in layouts that are unintuitive and difficult for engineers to interpret or validate. While these works primarily focus on minimizing the number of waveguide crossings, they fail to account for critical routability considerations such as bend radii, port accessibility, and the area overhead of crossings. As a result, their placement solutions frequently lead to routing failures and do not yield physically legal layouts.

In this work, we propose Apollo, an automated, GPU-accelerated, routing-informed placement engine for large-scale PICs. Our framework directly tackles the inherent limitations of prior solutions by integrating physical routing constraints and designer-specified layout rules into the placement phase. Apollo models waveguide routing as a first-class entity, enabling layout-aware placement that explicitly considers bend radii, crossing penalties, and port orientations. In contrast to prior methods that neglect practical layout constraints, Apollo ensures high-quality layouts that are physically routable.

*<https://github.com/ScopeX-ASU/Apollo>

- **Routing-Informed Placement Engine:** Co-optimizes bending-aware wirelength and routing congestion-driven component distribution using a unified differentiable framework.
- **Bending-Aware Wirelength and Spacing Estimation:** Accurately models waveguide geometries and local congestion to preserve layout feasibility.
- **Progressive Constraint Handling:** Handles spacing, alignment, symmetry, and uniformity using a conditional projected gradient descent strategy.
- **Blockwise Adaptive Nesterov Optimizer:** Ensures stable and high-quality convergence on mixed-size PIC designs via parameter group-wise gradient updates.
- **Superior Routability:** Compared to existing methods, our Apollo can generate high-quality layouts for large-scale PICs with an average routing success rate of 94.79% across all benchmarks within minutes.

2 Preliminaries

In this section, we review the background and the motivation.

2.1 Photonic Circuit Placement

Commercial PIC physical design toolflow has traditionally relied on schematic-driven layout methodology [13], in which devices are placed according to their logical positions and then manually abutted via waveguide ports, a time-consuming process that does not scale to large systems. To boost productivity, the work [8] introduced a semi-automatic Photonic CAD tool for visual place-and-route of on-chip photonic networks. Fully automatic frameworks such as Proton [1] and Platon [23] then focused on minimizing insertion loss, particularly at crossings, while another work [5] further reduced loss through device flipping and rotation. A recent research incorporated critical-path insertion-loss optimization directly into placement [4]. Another work [22] jointly optimized network topology and its physical realization. Thermal-aware approaches have also been proposed [10]. However, these efforts largely overlook the geometric cost of waveguide routing itself. In this work, we argue that a routing-informed placement engine, one that explicitly accounts for bends, crossings, and congestion during device placement, is essential for truly scalable, routable PIC layout.

2.2 Analytical Mixed-Size Placement

Analytical placement usually consists of three stages: global placement (GP), legalization (LG), and detailed placement (DP). Global placement spreads out instances in the layout; legalization removes the remaining overlaps between instances and aligns instances to placement sites; detailed placement performs incremental refinement to further improve the quality. Since the quality of the final placement solution largely depends on the global placement and legalization stage, we mainly focus on them in this work.

Global placement aims at minimizing the wirelength cost subject to density constraints. The density constraints are relaxed into a density penalty term, computed as the potential energy of an electrostatic system where cells are modeled as charges, as in ePlace [16]. The problem can be formulated as:

$$\min_{(x,y)} \sum_{e \in \mathcal{E}} w_e \cdot WL(e; x, y) + \lambda \cdot D(x, y), \quad (1)$$

where \mathcal{E} is the set of nets, (x, y) are the coordinates of all the instances, w_e is the weight of net e , $WL(\cdot)$ is a differentiable wirelength cost function, and $D(\cdot)$ denotes the density penalty that spreads instances out in the layout. The non-overlapping constraint can be satisfied by

gradually increasing the density weight λ , e.g., using the Lagrangian method. From an optimization perspective, this formulation can be extended to mixed-size placement as well, converting the analytical mixed-size global placement problem into an unconstrained optimization problem with a differentiable objective function. Our Apollo will use this analytical framework to co-optimize waveguide routing lengths and congestion while ensuring device non-overlapping and other user-defined layout constraints in a scalable way.

3 Apollo: Automated PIC Placement Framework

In this section, we present the details of our PIC placer Apollo.

3.1 Understanding the Placement Challenges of Photonic Circuits

Photonic IC layout presents fundamentally different challenges from digital VLSI due to the analog, custom-layout nature of photonics and stringent physical constraints. In PICs, placement and routing are tightly coupled. Layout feasibility and quality are highly sensitive to component positions, port orientations, and available spacing. This tight coupling arises from several critical factors:

① *Photonic waveguides are directional, curvy, and space-consuming.* Unlike flexible metal wires in VLSI, optical waveguides are of large width (500-1000 nm), require large minimum bend radii (5-10 μm) to avoid optical loss, and cannot intersect with other devices (unlike PCB routing), which are themselves optical waveguide structures. As a result, routing becomes a dominant constraint during placement.

② *PICs are limited to very few routing layers.* This leads to scarce routing resources and makes 90-degree waveguide crossings topologically inevitable in many designs, introducing insertion loss, crosstalk, and significant area overhead (as crossings take spaces like via in VLSI/PCB), further increasing congestion and routability complexity.

③ *Ports are orientation-sensitive and often densely packed.* Improper port orientation can introduce excessive bends and detours as shown in Fig. 1, which frequently result in unroutable or lossy connections, shrinking the feasible routing solution space and necessitating orientation-aware placement strategies. Moreover, photonic devices often feature high port density; for instance, a multimode interference (MMI) device or star coupler may include 16 input and 16 output ports tightly packed along its edge. These dense configurations implicitly require substantial surrounding space to accommodate waveguide escape routes and crossings, further complicating placement.

④ *Mixed-size PIC components are geometrically diverse.* Photonic components span from compact $2 \times 2 \mu\text{m}^2$ compact splitters/resonators to millimeter-scale high-speed modulators with extreme aspect ratios (e.g., 1:20). This diversity mirrors the challenges of mixed-size VLSI placement, long known to be combinatorial and ill-conditioned for joint optimization.

⑤ *Lack of high-quality PIC routers.* In PIC layout, placement quality is tightly coupled with the capabilities of the router. However, the current EPDA ecosystem lacks mature, high-quality routers for PICs. This practical limitation means that many routing complexities [14, 28], such as bend management, crossing avoidance, and port alignment, must be preemptively handled during placement. As a result, the burden of achieving a routable and high-performance layout is currently disproportionately shifted to the placement engine.

These challenges collectively demand a routing-informed placement framework that models photonic-specific constraints from the outset. Our Apollo framework addresses these challenges by incorporating routing-aware component spacing and bending-aware wirelength modeling, ensuring high-quality and physically routable layouts for large-scale PICs.

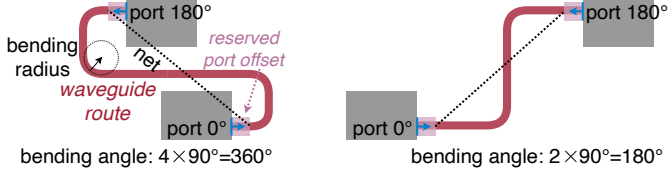


Figure 1: Solutions with the same HPWL give different waveguide routing and total bending angles.

3.2 Overview of Apollo Framework

Apollo consists of an analytical global placement engine with a simple legalization stage. It is built on an open-source GPU-accelerated VLSI placement framework, DREAMPlace, that enjoys fast PyTorch-based programming and automatic differentiation engines for gradient-based optimization. It features a LEF/DEF-inspired PIC benchmark suite in YAML format based on real large-scale photonic tensor core designs, which is intentionally designed to support seamless integration with open-source PIC layout tools GDSFactory and an open-source automated PIC router LiDAR for end-to-end evaluation. Different from the standard LEF/DEF definition, we extend it to incorporate the port orientation of photonic components with targeted waveguide width and crossing sizes. Different from schematic-driven layout methodology that manually determines a waveguide routing plan with crossings pre-inserted in the schematic, we **assume crossings are unknown during placement and only inserted during routing instead**. To enable a routing-informed global placement process, we will introduce our key innovations in the following section.

3.3 PIC-Specific Placement Objective Functions

3.3.1 Asymmetric Bending-Aware Wirelength. Weighted-average wirelength (WA-WL) is widely used in VLSI placement for wirelength cost, which is used to approximate the half-perimeter wirelength (HPWL). However, unlike VLSI routing, PICs only have two-pin nets to describe waveguides and use ports instead of pins to connect the waveguides. These ports are essentially open-ended waveguide segments that can only be accessed in a specific orientation. Therefore, HPWL is not an appropriate approximation for the PIC waveguide routes. The original WA-WL is symmetric along the x and y -axis. However, as shown in Fig. 1, PIC does not prefer sharp turns. Instead, we need to use a smooth, curvy waveguide for bending, which is area-consuming and thus harms routability.

To be aware of the *port orientation* introduced by inappropriate component locations, and *mitigate the routing resources consumption by curvy bending* in PICs, we propose a **asymmetric bending-aware wirelength function**, named as cosWA function, as follows,

$$\text{cosWA}_e = (1 + W_\theta) \cdot \left(\frac{\sum_{i \in e} x_i e^{\frac{x_i}{\gamma}} - \sum_{i \in e} x_i e^{-\frac{x_i}{\gamma}}}{\sum_{i \in e} e^{\frac{x_i}{\gamma}} - \sum_{i \in e} e^{-\frac{x_i}{\gamma}}} \right)^\alpha$$

$$W_\theta = [(c - \cos \theta_1)_+]^2 + [(c - \cos \theta_2)_+]^2 \quad (2)$$

$$\cos \theta_i = \frac{\mathbf{w} \cdot \mathbf{v}_i}{\|\mathbf{w}\| \|\mathbf{v}_i\|}, \quad i = 1, 2$$

$$\mathbf{w} = (dx, dy), \quad \mathbf{v} \in \{(1, 0), (0, 1), (-1, 0), (0, -1)\}$$

Eq. (2) shows the horizontal wirelength of net e . \mathbf{w} is defined as the vector from the first pin to the second pin of the net, where dx and dy represent the horizontal and vertical distances between the two pins, respectively. \mathbf{v} is the unit vector indicating the port orientation. Then θ_i represents the angle between the port orientation and the straight line connection of two-pin nets. In this way, a large θ (e.g., 150-degree) will result in a large penalty factor W_θ , so that we can penalize the net

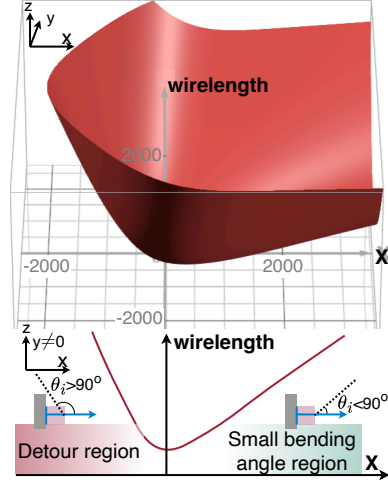


Figure 2: Proposed bending-aware wirelength function: detour region results in larger wirelength cost.

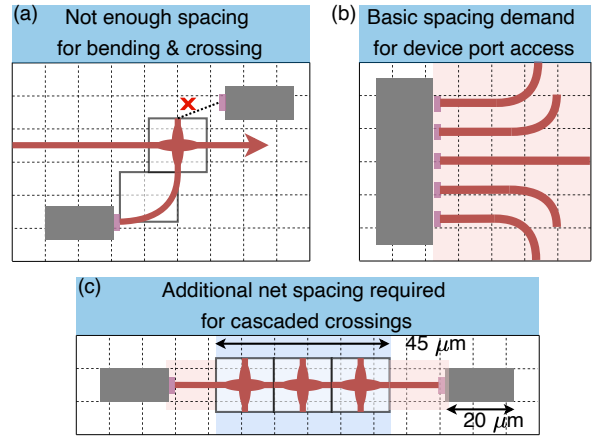


Figure 3: Illustration of the critical spacing requirements in PIC routing that should be considered during placement.

with a large bending angle as shown in Fig. 2. $(\cdot)_+$ is the ReLU function that casts a negative value to 0, which makes sure no extra penalty is added for acute angles. The parameter $c \in [0, 1]$ is used to control the angle margin. The W_θ will become 0 when both θ_1 and θ_2 are within the angle margin defined by c . The cosWA function is asymmetrical along the x -axis if the corresponding port has a horizontal orientation. Note that the y -axis cosWA wirelength in this case will degrade to a symmetric WA wirelength. Similarly, it simply flips to x -symmetry and y -asymmetry for vertical ports.

Another key variable here is the exponent α that makes this function not an approximation of linear HPWL, which leads to very long wires and is unable to encourage symmetric layouts, as many placement solutions can have the same HPWL. To borrow the *long-wire penalty property* of quadratic wirelength without overestimating long waveguides and ignoring short waveguides, we add this $\alpha \in [1, 2]$ exponent to make it smoother. Empirically, we set α to 1.4, which is later identified as the optimal setting in the ablation study section.

3.3.2 Routing-Informed Net Spacing Model. A major root cause of routing failure in PICs is **spacing deficiency**. Unlike VLSI, where multiple metal layers are available, PICs are typically limited to a single optical routing layer. As a result, routing resources are scarce, and

waveguide crossings and curved bends consume significant area. Without enough spacing, routing is likely to fail (see Fig. 3(a)). Hence, it is essential to integrate *routing-informed spacing estimation* into placement. Accurate modeling of the spatial demands from port density, bend radii, and anticipated crossings can preempt routing congestion and ensure sufficient space for successful waveguide routing.

Informed by empirical analysis of the routing behavior for the open-source PIC router LiDAR, the **main routing failure comes from the port accessibility and crossing insertion**, we propose the routing-informed net spacing model S_i based on the port density P_{dens} and routing congestion R_{cong} . For each net, we define a *spacing demand* based on two factors: (1) the **local port density**, which captures how many co-directional ports are clustered, and (2) **estimated routing congestion**, which accounts for potential waveguide crossings. Specially, for a 2-pin net i connecting port p_1 (on node v_1) and port p_2 (on node v_2), the net spacing demand S_i is estimated as

$$\begin{aligned} S_i &= \max(S_i(v_1, p_1), S_i(v_2, p_2)), \\ S_i(v, p) &= P_{\text{dens}}(v, p) + R_{\text{cong}} \\ P_{\text{dens}}(v, p) &= r_{\text{bend}} + \frac{1}{2}P_{\text{num}}(v, p) \times S_{\text{crs}}, \\ P_{\text{num}}(v, p) &= |\{p_j \in \text{Ports of } v \mid \text{Dir}(p_j) = \text{Dir}(p)\}|. \end{aligned} \quad (3)$$

where the local port density P_{dens} represents the basic spacing needed for feasible port access, as shown in Fig. 3(b). The r_{bend} is the bending radius, e.g., $5 \mu\text{m}$. $P_{\text{num}}(v, p)$ is the number of ports on node v that have the same orientation as port p connected to net i . S_{crs} is the waveguide crossing height/width (typically a 4-port square-shaped device), e.g., $10 \mu\text{m}$. Since a net connects two ports of different components, which gives different port density spacing for the net, we choose the one with the largest required spacing.

Apart from the basic spacing P_{dens} , net crossings also cost a large amount of routing resources that further cause congestion, which should be considered in the spacing model. Once the positions of all components are roughly stabilized, we estimate the potential crossings to calculate the additional spacing required. We estimate the crossings by calculating the line segment intersections between nets during placement. If a crossing occurs, we approximate the additional spacing needed for cascaded crossings (see Fig. 3(c)) by multiplying the number of estimated crossings $\#CR_{\text{net}}$ by the size of a crossing S_{crs} ,

$$R_{\text{cong}} = \#CR_{\text{net}} \times S_{\text{crs}}$$

Since net crossings $\#CR_{\text{net}}$ evolve as components move, the congestion estimate is updated periodically every 100 iterations starting from the 100th iteration to improve accuracy. Finally, we define the net spacing penalty for a net e as,

$$NS_e(x, y) = [(v_x \cdot dx - S_i)_+]^2 + [(v_y \cdot dy - S_i)_+]^2, \quad (4)$$

where (v_x, v_y) is the port's direction vector, which determines the port's orientation. Then we calculate the spacing needed for the x-axis and y-axis using a quadratic function. $(\cdot)_+$ is $\max(\cdot, 0)$ that makes sure we only penalize when spacing is below the threshold. This penalty term effectively encourages the placer to reserve sufficient space for waveguide routing throughout the layout process.

3.3.3 Modified Augmented Lagrangian for Electrostatic Density. We follow the density model of DREAMPlace3.0 [6] to remove the overlap among components defined as $D(x, y)$, which models components as electric charges, density as potential energy, and density gradient as electric field. The electric potential and field distribution are obtained by solving Poisson's equation from the charge density distribution via spectral methods with a two-dimensional fast Fourier transform (FFT).

To accelerate the cell spreading and avoid complete cell overlapping that can hardly be separated by this discretized electrical field, we employ a modified augmented Lagrangian method with a quadratic density term modulated by the density weight λ_D to *avoid overly fast spreading in the early stages* that is critical to wavelength optimization,

$$\mathcal{D}(x, y) = \lambda_D \cdot (D(x, y) + \frac{1}{2}\rho D^2(x, y)), \quad (5)$$

where ρ is the quadratic penalty coefficient, empirically set to 2,000. The scheduling of λ_D follows the overflow-based schedule [15].

Filler Size Setting. Fillers are dummy cells introduced to occupy whitespace and maintain electrical equilibrium. The size of fillers directly influences the granularity of the placement grid and thus affects overall placement precision. In DREAMPlace, filler sizes are estimated as the average standard cell dimensions. However, this assumption breaks down in mixed-size PIC designs, where component geometries vary significantly. To better suit the photonic context, we compute the filler aspect ratio based on the average aspect ratio of movable photonic components and clamp it within a predefined range (e.g., [0.2, 5.0]) to avoid extreme shapes. Additionally, we flip the filler orientation *opposite to the predominant optical signal flow* to further improve placement adaptability. In our benchmarks, optical signals primarily propagate from left to right, making placement accuracy in the x direction especially critical. Therefore, we adopt *tall and narrow fillers* to increase resolution along the x-axis, enabling fine-grained control over component placement.

3.3.4 Overall Placement Objective Function. In Apollo, the overall combined objective function is as follows,

$$\mathcal{L}(x, y) = \sum_{e \in E} \cosWA_e(x, y) + \lambda_{NS} \sum_{e \in E} NS_e(x, y) + \mathcal{D}(x, y), \quad (6)$$

where E denotes the set of all nets, and λ_{NS} is the weight of the spacing penalty, which is set to 1 in the later experiment. Initially, all components are randomly placed at the layout center, overlapping each other. Guided by wavelength minimization and density control, components progressively move toward optimized positions, while the spacing penalty pushes apart those in congested areas to meet spacing constraints.

3.4 Designer-Constrained Placement via Conditional Projection and Cell Inflation

Design constraints provided by photonic circuit designers often capture domain-specific knowledge that is crucial for ensuring functional and robust layouts. For example, some thermo-optic phase shifters should have enough spacing around them to avoid circuit performance degradation caused by the thermal crosstalk. Some components or nets should have an almost identical routing topology to achieve phase matching. However, manually enforcing these constraints is tedious and error-prone, especially under tight geometric constraints. Our placement engine supports these constraints automatically using two complementary mechanisms: (1) cell inflation for spacing-driven constraints, and (2) projected gradient descent for alignment and uniformity constraints.

Case Study 1: Alignment and Regularity via Conditional Projected Gradient Descent. Many coherent photonic circuits require phase-matched routing or geometric symmetry for robustness considerations. A common practice is to align such components in rows or grids, ensuring identical optical path lengths and reducing bending complexity. Additionally, uniformly distributed components help reserve adequate spacing between waveguides to minimize unwanted coupling and crosstalk. Moreover, a regular, evenly spaced layout

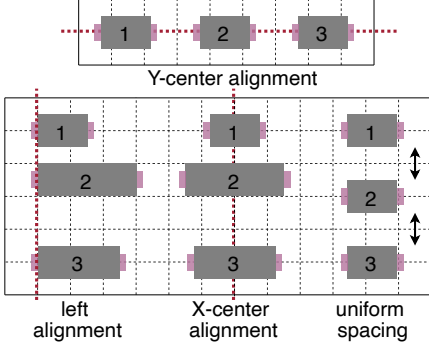


Figure 4: Common physical design constraints in PICs.

is inherently more robust to fabrication variations, boosting overall yield and making it far easier to scale designs to larger arrays without sacrificing performance.

We enforce such constraints using a conditional projected gradient descent scheme. In our PIC benchmark suite, we extend the LEF/DEF syntax to support group-based constraints, e.g., `{alignment: 'left', n_1, n_2, n_3 }` describes left edge alignment among a group of three nodes. Figure 4 illustrates different constraints, i.e., left, X-center, and Y-center alignment and uniform spacing.

During placement, nodes in each constraint group are progressively projected toward their feasible locations to honor the constraints. To avoid overly hard projection to the feasible set that hinders optimization, we relax the constraints and progressively enforce them using conditional projected gradient descent (CPGD). We handle these constraints using a projection-based scheme integrated into the gradient descent optimization. Specifically, we adopt a *projection scheduler* that controls the *projection sharpness* of a set of compound projection operators. At each iteration t (out of a total T steps), the updated position of each component x_i is computed as a smooth interpolation between its current location and the projected target $x_{new,i}$:

$$x_i = (1 - s_t) \cdot x_i + s_t \cdot x_{new,i}, \quad (7)$$

where $s_t \in [0, 1]$ is an iteration-dependent sharpness value that increases over the course of optimization. Here, we use a cosine function to gradually increase projection strength:

$$s_t = s_0 + (s_T - s_0) \cdot \frac{1 - \cos(\pi t/T)}{2}, \quad (8)$$

where s_0 and s_T denote the initial and final sharpness, respectively. For each constraint group G with component positions $\{x_i\}_{i \in G}$, the projected target position $x_{new,i}$ is computed based on the type of constraint. For *alignment constraints*, all components are projected to their group centroid:

$$x_{new,i} = \frac{1}{|G|} \sum_{j \in G} x_j. \quad (9)$$

For *quantization (uniform spacing)* constraints, each component is snapped to a regular grid within the group’s bounding box range:

$$x_{new,i} = x_{\min} + s_x \cdot \text{round}\left(\frac{x_i - x_{\min}}{s_x}\right), \quad s_x = \frac{x_{\max} - x_{\min}}{|G| - 1}. \quad (10)$$

As optimization progresses, the increasing sharpness s_t ensures that constraint satisfaction is gradually enforced, allowing initial placement flexibility while converging smoothly to the final aligned or evenly spaced configuration. Note that this projection is not performed after the optimizer descent step. Otherwise, it is not aware of this projection. The projection is applied after the tentative descent for step size estimation in the Nesterov optimizer (see Alg. 1 Line 18).

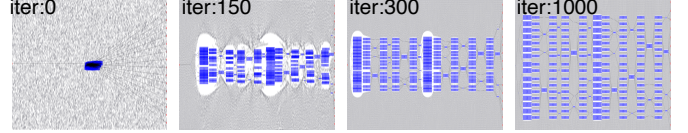


Figure 5: Placement animation of Apollo on ADEPT_16x16.

Case Study 2: Spacing Constraint via Cell Inflation. A common constraint is to protect thermo-optic phase shifters from causing thermal crosstalk by enforcing minimum spacings from other cells, e.g., $50 \mu\text{m}$. This can be converted to an overlap removal problem and elegantly solved by density minimization once the protected cell is inflated with a $50 \mu\text{m}$ *halo* (as defined in the LEF/DEF syntax).

3.5 Stabilizing Mixed-Size PIC Placement via Blockwise Adaptive Nesterov Optimizer

As emphasized in Section 3.1, the significant heterogeneity in component sizes presents a major challenge for stability and convergence in PIC placement. To address this, we introduce a *Blockwise-Adaptive Barzilai–Borwein (BBB)* step size scheme in Nesterov-accelerated gradient descent optimizer in Alg. 1, named BNAG, specifically tailored for mixed-size PIC layouts. In our formulation, all placement variables are partitioned into 4 logical blocks: movable instances and dummy fillers in both the x and y dimensions. An independent Barzilai–Borwein [2] step size is computed for each block, and the resulting values are truncated to prevent instability. This blockwise treatment allows us to stabilize the motion of large nodes by decoupling their updates from smaller cells and encourages faster convergence of filler cells to surround movable instances and fill their spacing. While the Barzilai–Borwein (or inverse Lipschitz) step size is effective for acceleration, it often overestimates the appropriate step size, especially when certain cells are *already near-optimal*. To mitigate this and ensure convergence, we apply a global *cosine annealing schedule* to gradually reduce the effective step size over time. This helps maintain robustness in the presence of highly uneven cell dimensions and density. Additionally, we project v_{k+1} onto the feasible placement region to correct for potential extrapolation beyond legal bounds in Nesterov’s momentum. This guarantees that gradient evaluations occur at valid locations, preserving placement legality and improving stability.

3.6 Legalization

Since the components in the PICs have a relatively large feature size, we directly perform macro legalization [15] after the global placement. The greedy Macro Legalization operator legalizes movable macros in two stages: first, a coarse Hanan-grid pass for small clusters and blocked macros, and then an LP/graph-based refinement, alternating until total and weighted displacement are minimized and the best legal layout is retained.

We show an **animation of the placement process** of Apollo on a representative benchmark to visualize its behavior in Fig. 5.

4 Evaluation Results

The Apollo framework builds upon DREAMplace, with PIC-specific optimization. All the experiments are conducted on an AMD EPYC 7763 Linux server with a 2.9GHz CPU and NVIDIA RTX A6000 GPU. **Benchmarks.** In our customized realistic benchmark suites, we use the Clements-style Mach-Zehnder Interferometer (MZI) [17] arrays and an auto-searched photonic tensor core design (ADEPT) [7] at two different scales. For each circuit, we generate two chip configurations: S size with a very compact die size with a $5 \mu\text{m}$ bending radius and

Algorithm 1 Blockwise Adaptive Nesterov-accelerated Gradient Descent (BNAG) Optimizer (one step).

Input: a_k (optimization parameter), u_k (major solution), v_k (reference solution), v_{k-1} , $\nabla f(v_k)$, $\nabla f(v_{k-1})$, $\{B_1, \dots, B_m\}$

Output: u_{k+1} (updated sol.), v_{k+1} , a_{k+1}

```

1: for  $j = 1$  to  $m$  do
2:    $g^{(k-1)} \leftarrow \nabla f(v_{k-1})[B_j]$ 
3:    $g^{(k)} \leftarrow \nabla f(v_k)[B_j]$ 
4:    $s^{(k-1)} \leftarrow v_k[B_j] - v_{k-1}[B_j]$ 
5:    $y^{(k-1)} \leftarrow g^{(k)} - g^{(k-1)}$ 
6:    $\alpha_{bb}^{(j)} \leftarrow \frac{(s^{(k-1)})^T y^{(k-1)}}{(y^{(k-1)})^T y^{(k-1)}}$ 
7:    $\alpha_{lip}^{(j)} \leftarrow \frac{\|s^{(k-1)}\|}{\|y^{(k-1)}\|}$ 
8:   if  $\alpha_{bb}^{(j)} > 0$  then  $\alpha_k^{(j)} \leftarrow \alpha_{bb}^{(j)}$ 
9:   else  $\alpha_k^{(j)} \leftarrow \min(\alpha_{lip}^{(j)}, \alpha_{k-1}^{(j)})$ 
10:   $\alpha_k^{(j)} \leftarrow \alpha_k^{(j)} \cdot \eta_k$ 
11: end for
12: for each block  $j = 1, \dots, m$  and each  $i \in B_j$  do
13:   $u_{k+1}[i] \leftarrow v_k[i] - \alpha_k^{(j)} \cdot \nabla f(v_k)[i]$ 
14: end for
15:  $a_{k+1} \leftarrow \frac{1 + \sqrt{4a_k^2 + 1}}{2}$ 
16:  $v_{k+1} \leftarrow u_{k+1} + \frac{a_k - 1}{a_{k+1}}(u_{k+1} - u_k)$ 
17:  $\eta_{k+1} \leftarrow \eta_{\min} + \frac{1}{2}(\eta_0 - \eta_{\min}) \left(1 + \cos\left(\frac{\pi k}{K_{\max}}\right)\right)$ 
18: project( $v_{k+1}$ )  $\triangleright$  enforce constraints
19: return  $u_{k+1}$ ,  $v_{k+1}$ ,  $a_{k+1}$ 

```

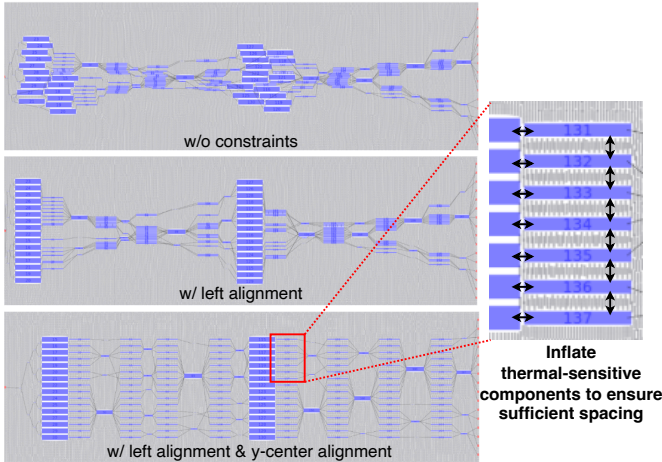


Figure 6: Proper placement constraints ensure high PIC layout quality on ADEPT_16x16 circuit.

L size with a more relaxed die size with a $10 \mu m$ bending radius, and use a $10 \mu m \times 10 \mu m$ crossing size.

Baselines. We compare Apollo against two baselines: DREAMPlace+RO, DREAMPlace augmented with routability optimization via node inflation based on the RUDY congestion map, and a PCB placement tool named Cypress [26] that minimizes net crossings to improve routability.

Evaluation Metrics. We evaluate the placement solution quality using several key metrics: **routability**, **number of crossings (#CR)**, and **runtime**. Routability is defined as the ratio of successfully routed

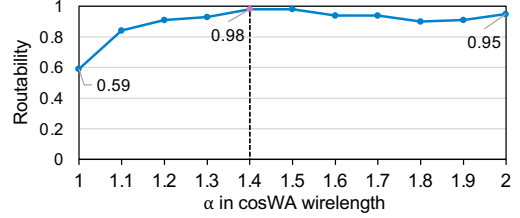


Figure 7: Impact of the exponent α in our cosWA wirelength on routability on ADEPT_16x16.

nets to the total number of nets. To evaluate routability, we perform exact waveguide routing on the legalized placement using an open-source PIC router LiDAR [28]. Crossing counts, as an indirect routability indicator, are computed directly from the placement by detecting intersections of the straight-line segments representing each two-pin net. Insertion loss IL_{max} is a critical metric related to waveguide bending angles, routing length, and crossings that implies the optical signal integrity. Note that without a fully DRV-free layout, it is not meaningful to evaluate post-routing insertion loss. Instead, we estimate the **pre-routing insertion loss** IL_{max} after the placement stage from the estimated routing length and port orientations, predicting bending angles and applying an analytical waveguide insertion loss model [28].

4.1 Handling of Design Constraints

Constraints are absolutely critical in PIC placement. Due to the complex port alignments in PIC circuits, unreasonable device placement can severely degrade routability, make it impossible to produce a legal layout, induce thermal crosstalk, and exacerbate manufacturing variability. Therefore, it is essential to impose layout constraints from the very beginning. Figure 6 illustrates how placement evolves as left-alignment and y-center-alignment constraints are gradually applied. With no constraints, components are placed in a highly irregular fashion, leading to severe port misalignment and an entirely unroutable solution. As constraints accumulate, the circuit becomes increasingly regular: grouping certain components for left alignment arranges them into neat columns, though the groups remain too close and can introduce crosstalk; adding y-center alignment then transforms the layout into a true grid, perfectly aligning ports and markedly enhancing routability. To ensure a fair comparison in our experiments, we apply the same constraint set to all baseline methods so that each can produce a valid, reasonable placement.

4.2 Main Results

For the parameters in the proposed cosWA function, we first performed a parameter sweep on the ADEPT_16x16 benchmark. As shown in Fig. 7, we varied α from 1.0 to 2.0 in increments of 0.1, ran placement followed by full routing at each setting, and recorded the resulting routability. We then selected $\alpha = 1.4$, the value that produced the highest routability for all subsequent experiments. As for the margin parameter c in cosWA, because we employ Manhattan routing in later stages, we set $c=0$; that is, once the ports achieve a 90-degree orientation, no further bending-angle penalty is applied.

Moreover, in all subsequent experiments, we initialize every component at a random center location. As Table 2 shows, even when compared to manually placing components in favorable starting positions, our method still converges to high-quality solutions under random center initialization, demonstrating the robustness of the proposed approach.

Table 1: Comparison of the routability, the estimated number of crossings after placement, and runtime (s).

Benchmark	DREAMPlace /w routability optimization [15]			Cypress [26]			Apollo		
	#CR	Routability	Runtime (s)	#CR	Routability	Runtime (s)	#CR	Routability	Runtime (s)
Clements_8x8_S	7	43.68%	30.15	0	72.41%	15.80	0	98.85%	21.13
Clements_8x8_L	8	50.42%	31.64	0	77.73%	16.48	0	98.85%	23.07
Clements_16x16_S	20	63.70%	43.67	13	41.91%	22.33	2	95.05%	31.89
Clements_16x16_L	28	59.43%	45.19	17	48.64%	23.01	2	96.04%	32.64
ADEPT_8x8_S	38	48.82%	39.64	61	42.73%	21.58	28	90.05%	22.79
ADEPT_8x8_L	49	70.63%	40.92	53	61.85%	22.63	33	95.04%	23.11
ADEPT_16x16_S	83	47.96%	61.57	105	35.42%	34.36	79	98.12%	35.88
ADEPT_16x16_L	124	40.87%	63.24	113	33.79%	35.62	87	90.60%	39.62
ADEPT_32x32_S	261	44.20%	117.41	209	62.71%	64.43	206	96.74%	65.99
ADEPT_32x32_L	263	48.37%	124.73	253	46.74%	69.89	237	93.38%	71.62
ADEPT_64x64_S	381	43.89%	235.31	358	41.82%	133.39	314	97.10%	105.96
ADEPT_64x64_L	303	48.17%	238.47	297	50.83%	136.71	209	87.62%	110.93
Geo-mean	-	50.85%	89.33	-	51.38%	49.69	-	94.79%	48.72
Ratio	-	0.53	1.83	-	0.54	1.02	-	1	1

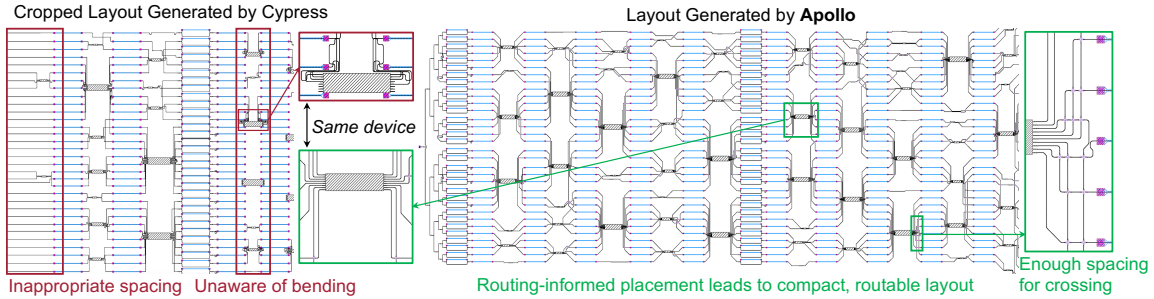


Figure 8: ADEPT_32x32 visual comparison of post-routing layout (viewed in KLayout) generated by Cypress and our Apollo.

Table 2: Random initialization vs. manual initial placement. Apollo is robust and does not rely on good initial placement.

Benchmarks	Random initialization		Manual initialization	
	#CR	Routability \uparrow	#CR	Routability \uparrow
Clements_16x16_S	2	95.05%	2	95.05%
ADEPT_32x32_S	206	96.74%	183	95.36%
ADEPT_64x64_S	314	97.10%	304	96.32%

Table 1 shows that Apollo consistently outperforms two baselines, four Clements and four ADEPT benchmarks, under both compact and relaxed die sizes. Apollo achieves a geometric mean routability of 94.79%, versus 50.85% for DREAMPlace and 51.38% for Cypress. In all S-size runs, Apollo exceeds 98.5% routability, drives net crossings down to zero on Clements_8x8_S and Clements_16x16_S, and finishes in under 20s on average, roughly 1.8x faster than DREAMPlace and on par with Cypress. Even on the largest ADEPT_64x64_L circuit, Apollo achieves 87.6% routability, significantly outperforming prior methods (~48%/51%), and reduces waveguide crossings by ~30%. Despite the L benchmark offering a larger area, its 10 μm bending radius makes routing more challenging than in the S benchmark. This illustrates that a larger die size does not suffice to resolve local congestion issues.

Unlike the baselines, which optimize a generic VLSI wirelength objective, Apollo is designed specifically for PIC placement and explicitly accounts for the spatial footprint of curved waveguide bends and crossings. This routing-aware formulation yields post-placement layouts that are inherently highly routable. Even though DREAMPlace+RO performs node inflation to create extra port spacing based on congestion maps, it does not capture the area overhead of curvy bends, leaving many nets difficult to access and route.

4.3 Ablation Studies

4.3.1 Wirelength Function Comparison. To evaluate the impact of different wirelength models on photonic layout quality, we compare weighted-average wirelength (WA-WL) [9], log-sum-exponential (LSE) [11], quadratic wirelength, and our proposed cosine-weighted wirelength (cosWA). As shown in Table 3, cosWA achieves the lowest maximum insertion loss (IL_{max}), outperforming WA-WL, LSE, and quadratic by 1.10x, 1.13x, and 1.05x, respectively. It also produces the smallest total bending angle improvements of roughly 1.18x, 1.38x, and 1.21x over WA-WL, LSE, and quadratic wirelength. These reductions in loss and detour translate directly into *compact layouts, reduced local congestion, and ultimately significantly improved routability.*

4.3.2 Spacing Model Comparison. Although cosWA wirelength substantially reduces unnecessary waveguide detours, regions traversed by multiple waveguides can still become heavily congested. This is especially true for multi-port components, where servicing or escaping several ports in a tight area entails numerous bends and crossings—scenarios that conventional VLSI node-inflation cannot adequately resolve. To address this, we borrow from automated waveguide routing practices and introduce explicit spacing models to mitigate congestion arising from both bends and crossings. We conduct an ablation study comparing: (1) no spacing mode, (2) port-density inflation (PI): adding Halo to cells proportional to the product of number of ports and bending radius for the cell, our proposed basic net spacing schemes, including two variants: (3) bend-radius spacing (r_{bend}) and (4) the port-count-crossing product S_{CRS} , and (5) our final net spacing model with basic spacing and congestion cores ($P_{\text{dens}} + R_{\text{cong}}$).

As Table 4 shows, port-density-based cell inflation does improve routability over *no spacing model* (geometric-mean success rate rises from 67.87% to 73.12%), but it still fails on ADEPT circuits with heavy multi-port crossings. Introducing a basic bend-radius clearance (r_{bend}) further boosts the geo-mean to 76.06%, yet it still overlooks congestion

Table 3: Comparison of total bending angle BA_{tot} and max insertion loss IL_{max} for different wirelength models. (↓): lower is better.

Benchmark	WA-WL		LSE		Quadratic		Proposed cosWA	
	BA_{tot} ↓	IL_{max} ↓	BA_{tot} ↓	IL_{max} ↓	BA_{tot} ↓	IL_{max} ↓	BA_{tot} ↓	IL_{max} ↓
Clements_8×8_S	1.64E+04	0.32	1.57E+04	0.30	1.66E+04	0.33	1.57E+04	0.31
Clements_16×16_S	5.61E+04	0.59	8.24E+04	0.73	5.63E+04	0.60	5.37E+04	0.60
ADEPT_8×8_S	2.44E+04	0.61	3.09E+04	0.67	2.40E+04	0.60	2.06E+04	0.57
ADEPT_16×16_S	5.96E+04	1.31	7.49E+04	1.37	5.62E+04	1.28	5.17E+04	1.25
ADEPT_32×32_S	1.63E+05	2.58	1.74E+05	2.46	1.60E+05	2.44	1.17E+05	2.32
ADEPT_64×64_S	2.74E+05	3.30	3.17E+05	3.43	2.96E+05	3.04	2.44E+05	2.89
Geo-mean	9.90E+04	1.45	1.16E+05	1.49	1.01E+05	1.38	8.37E+04	1.32
Ratio	1.18	1.10	1.38	1.23	1.19	1.05	1.00	1.00

Table 4: Routing success rates (higher the better) under different placement spacing models (w/o spacing, port-density-based component inflation (PI) and our spacing model with different configuration) on various benchmarks (S is small chip space and L is large chip space). Our net spacing model, based on joint waveguide, bending, and crossing modeling (last column), leads to the best routing success rate. PI : cell inflation based on pin density.

Benchmark	w/o spacing	PI	Only basic spacing (P_{dens})		$P_{dens} + R_{cong}$
			r_{bend}	$P_{num} \times S_{crs}$	
Clements_8×8_S	94.25%	95.40%	95.40%	94.25%	98.85%
Clements_8×8_L	93.01%	96.43%	95.54%	97.70%	98.85%
Clements_16×16_S	82.51%	85.81%	85.48%	87.13%	95.05%
Clements_16×16_L	74.59%	85.15%	87.13%	93.07%	96.04%
ADEPT_8×8_S	76.38%	78.74%	70.87%	87.40%	90.05%
ADEPT_8×8_L	65.80%	70.08%	76.38%	87.74%	95.04%
ADEPT_16×16_S	60.50%	61.76%	54.23%	95.61%	98.12%
ADEPT_16×16_L	63.95%	69.91%	84.01%	86.21%	90.60%
ADEPT_32×32_S	49.15%	51.89%	53.85%	96.35%	96.74%
ADEPT_32×32_L	62.40%	63.31%	72.62%	83.96%	93.38%
ADEPT_64×64_S	40.87%	52.65%	66.44%	91.29%	97.10%
ADEPT_64×64_L	50.98%	66.32%	70.77%	76.94%	87.62%
Geo-mean	67.87%	73.12%	76.06%	89.80%	94.79%
Ratio	0.72	0.77	0.80	0.95	1.00

Table 5: Compare our proposed BNAG optimizer to other optimizers used in DREAMPlace. NA means the diverged placement has very low quality and cannot be evaluated for total bending angle (BA_{tot}) and maximum insertion loss (IL_{max}). BNAG achieves the most stable convergence and the highest placement quality. (↓) means lower is better.

Benchmark	Proposed BNAG			NAG+BB [2]			NAG [15]			Adam [12]		
	Status	BA_{tot} ↓	IL_{max} ↓	Status	BA_{tot} ↓	IL_{max} ↓	Status	BA_{tot} ↓	IL_{max} ↓	Status	BA_{tot} ↓	IL_{max} ↓
Clements_8×8_S	Success	1.56E+04	0.294	Success	1.59E+04	0.357	Diverge	1.61E+04	0.307	Diverge	1.53E+04	0.285
Clements_16×16_S	Success	5.75E+04	0.684	Success	5.35E+04	0.655	Diverge	5.54E+04	0.666	Success	5.96E+04	0.690
ADEPT_8×8_S	Success	1.78E+04	0.464	Success	1.81E+04	0.467	Success	1.95E+04	0.724	Diverge	NA	NA
ADEPT_16×16_S	Success	4.18E+04	1.097	Success	4.59E+04	1.610	Success	4.62E+03	1.540	Success	4.42E+04	1.363
ADEPT_32×32_S	Success	1.03E+05	1.497	Success	1.20E+05	2.958	Diverge	1.15E+05	1.932	Diverge	NA	NA
ADEPT_64×64_S	Success	2.48E+05	1.901	Success	2.52E+05	2.619	Success	2.69E+05	3.779	Success	2.93E+05	2.706
Geo-mean	-	8.06E+04	0.99	-	8.43E+04	1.44	-	7.99E+04	1.49	-	-	-
Ratio	-	1.00	1.00	-	1.05	1.46	-	0.99	1.51	-	-	-

at port crossings. When we explicitly account for crossing-induced spacing, the routability jumps to 89.80%. Finally, our full net spacing model, combining port-density and crossing-aware spacing, delivers **an average success rate of 94.79% across all benchmarks**.

4.3.3 Optimizer Comparison. Table 5 compares our proposed BNAG optimizer with NAG+BB [2], NAG [15], and Adam [12] over six benchmarks under the same number of iterations. While NAG and Adam optimizers diverge on several test cases, both BNAG and NAG+BB converge across all benchmarks. Moreover, BNAG achieves an improvement of approximately 1.04× in total bending cost and 1.29× in maximum insertion loss.

5 Conclusion

We introduce Apollo, the first routing-informed, constraint-aware, GPU-accelerated placement framework for large-scale photonic ICs. By modeling waveguide constraints, port orientations, and physical

spacing requirements as first-class citizens during placement, Apollo bridges the longstanding gap in photonic physical design automation toolflow. Through a synergy of bending-aware wirelength modeling, congestion-driven spacing, progressive constraint projection, and blockwise adaptive optimization, Apollo achieves highly routable layouts within minutes, even for PICs with thousands of components and a stringent chip area budget and layout constraints. In addition to the placement engine, we contribute a suite of realistic, large-scale PIC benchmarks derived from real photonic tensor core designs, fostering reproducibility and future research in layout automation. We show that Apollo consistently outperforms prior approaches in routability, layout quality, and runtime. As photonics becomes central to future computing and interconnects, Apollo offers a powerful foundation for next-generation electronic-photonic design automation.

References

- [1] Anja Boos, Luca Ramini, Ulf Schlichtmann, and Davide Bertozzi. 2013. PROTON: An automatic place-and-route tool for optical networks-on-chip. In *2013 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 138–145.
- [2] Yifan Chen, Zaiwen Wen, Yun Liang, and Yibo Lin. 2023. Stronger Mixed-Size Placement Backbone Considering Second-Order Information. In *Proc. ICCAD*. 1–9.
- [3] Yan-Lin Chen, Wei-Che Tseng, Wei-Yao Kao, and Yao-Wen Chang. 2023. A General Wavelength-Routed Optical Networks-on-Chip Model with Applications to Provably Good Customized and Fault-Tolerant Topology Designs. In *2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*. 1–7. doi:10.1109/ICCAD57390.2023.10323627
- [4] Yan-Ting Chen, Zhidan Zheng, Shao-Yun Fang, Tsun-Ming Tseng, and Ulf Schlichtmann. 2025. CPONoC: Critical Path-aware Physical Implementation for Optical Networks-on-Chip. In *Proceedings of the 30th Asia and South Pacific Design Automation Conference*. 1251–1256.
- [5] Yu-Kai Chuang, Kuan-Jung Chen, Kun-Lin Lin, Shao-Yun Fang, Bing Li, and Ulf Schlichtmann. 2018. PlanarONoC: concurrent placement and routing considering crossing minimization for optical networks-on-chip. In *Proceedings of the 55th Annual Design Automation Conference*. 1–6.
- [6] Jiaqi Gu, Zixuan Jiang, Yibo Lin, and David Z. Pan. 2020. DREAMPlace 3.0: multi-electrostatics based robust VLSI placement with region constraints. In *Proc. ICCAD*.
- [7] Jiaqi Gu, Hanqing Zhu, Chenghao Feng, Zixuan Jiang, Mingjie Liu, Shuhan Zhang, Ray T. Chen, and David Z. Pan. 2022. ADEPT: Automatic Differentiable DEsign of Photonic Tensor Cores. In *Proc. DAC*.
- [8] Gilbert Hendry, Johnnie Chan, Luca P Carloni, and Keren Bergman. 2011. VANDAL: A tool for the design specification of nanophotonic networks. In *2011 Design, Automation & Test in Europe*. IEEE, 1–6.
- [9] Meng-Kai Hsu, Valeriy Balabanov, and Yao-Wen Chang. 2013. TSV-aware analytical placement for 3-D IC designs based on a novel weighted-average wirelength model. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 32, 4 (2013), 497–509.
- [10] Fengxian Jiao, Sheqin Dong, Bei Yu, Bing Li, and Ulf Schlichtmann. 2018. Thermal-aware placement and routing for 3d optical networks-on-chips. In *Proc. ISCAS*.
- [11] Andrew B Kahng and Qinke Wang. 2006. A faster implementation of APlace. In *Proceedings of the 2006 international symposium on Physical design*. 218–220.
- [12] D. Kingma and J. Ba. 2015. Adam: A Method for Stochastic Optimization. In *Proc. ICLR*.
- [13] Twan Korthorst, Wim Bogaerts, Duane Boning, Mitch Heins, and Barton Bergman. 2023. Photonic integrated circuit design methods and tools. In *Integrated Photonics for Data Communication Applications*. Elsevier, 335–367.
- [14] Jun-Wei Liang and Iris Hui-Ru Jiang. 2025. TriHOT: Triangular and Hexagonal Norm Based Timing-Driven Optical Routing with Wavelength Division Multiplexing. *ACM Trans. Des. Autom. Electron. Syst.* (March 2025). doi:10.1145/3725888 Just Accepted.
- [15] Yibo Lin, Shounak Dhar, Wuxi Li, Haoxing Ren, Bruce Khailany, and David Z. Pan. 2020. DREAMPlace: Deep Learning Toolkit-Enabled GPU Acceleration for Modern VLSI Placement. *IEEE TCAD* (2020).
- [16] Jingwei Lu, Pengwen Chen, Chin-Chih Chang, Lu Sha, Dennis Jen-Hsin Huang, Chin-Chi Teng, and Chung-Kuan Cheng. 2015. ePlace: Electrostatics-based placement using fast fourier transform and Nesterov’s method. *ACM TODAES* 20, 2 (2015), 17.
- [17] Yichen Shen, Nicholas C. Harris, Scott Skirlo, et al. 2017. Deep Learning with Coherent Nanophotonic Circuits. *Nature Photonics* (2017).
- [18] Shanshan Yu Shiyue Hua, Erwan Divita et al. 2025. An integrated large-scale photonic accelerator with ultralow latency. *Nature* (2025).
- [19] Mikhail Bernadskiy Sufi R. Ahmed, Reza Baghdadi et al. 2025. Universal photonic artificial intelligence acceleration. *Nature* (2025).
- [20] Ebadollah Taheri, Sudeep Pasricha, and Mahdi Nikdast. 2022. ReSiPI: A Reconfigurable Silicon-Photonic 2.5D Chiplet Network with PCMs for Energy-Efficient Interposer Communication. In *Proc. ICCAD*.
- [21] Yvain Thonnart, Stéphane Bernabé, Jean Charbonnier, et al. 2020. POPSTAR: a robust modular optical NoC architecture for chiplet-based 3D integrated systems. In *Proc. DATE*.
- [22] Alexandre Truppel, Tsun-Ming Tseng, Davide Bertozzi, José Carlos Alves, and Ulf Schlichtmann. 2019. PSION: Combining logical topology and physical layout optimization for Wavelength-Routed ONoCs. In *Proceedings of the 2019 International Symposium on Physical Design*. 49–56.
- [23] Anja von Beuningen and Ulf Schlichtmann. 2016. PLATON: A force-directed placement algorithm for 3d optical networks-on-chip. In *Proceedings of the 2016 on International Symposium on Physical Design*. 27–34.
- [24] Yuyang Wang, Songli Wang, Robert Parsons, Asher Novick, Vignesh Gopal, Kaylx Jang, Anthony Rizzo, Chia-Pin Chiu, Kaveh Hosseini, Tim Tri Hoang, Sergey Shumarayev, and Keren Bergman. 2024. Silicon Photonics Chip I/O for Ultra High-Bandwidth and Energy-Efficient Die-to-Die Connectivity. In *Proc. CICC*. 1–8.
- [25] Yaoyao Ye, Jiang Xu, Baihan Huang, Xiaowen Wu, Wei Zhang, Xuan Wang, Mahdi Nikdast, Zhehui Wang, Weichen Liu, and Zhe Wang. 2013. 3-D Mesh-Based Optical Network-on-Chip for Multiprocessor System-on-Chip. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 32, 4 (2013), 584–596. doi:10.1109/TCAD.2012.2228739
- [26] Niansong Zhang, Anthony Agnesina, Noor Shbat, Yuval Leader, Zhiru Zhang, and Haoxing Ren. 2025. Cypress: VLSI-Inspired PCB Placement with GPU Acceleration. In *Proc. ISPD*.
- [27] Hailong Zhou, Jianji Dong, Junwei Cheng, Wenchan Dong, Chaoran Huang, Yichen Shen, Qiming Zhang, Min Gu, Chao Qian, Hongsheng Chen, et al. 2022. Photonic matrix multiplication lights up photonic accelerator and beyond. *Light: Science & Applications* 11, 1 (2022), 30.
- [28] Hongjian Zhou, Keren Zhu, and Jiaqi Gu. 2025. LiDAR: Automated Curvy Waveguide Detailed Routing for Large-Scale Photonic Integrated Circuits. In *Proc. ISPD*.