CPONoC: Critical-Path-Aware Physical Implementation for Optical Networks-on-Chip

Yan-Ting Chen[1], Zhidan Zheng[2], Shao-Yun Fang[1], Tsun-Ming Tseng[2], and Ulf Schlichtmann[2*]
[1]Department of Electrical Engineering, National Taiwan University of Science and Technology, Taipei, Taiwan
[2]Department of Electronic Design Automation, Technical University of Munich, Munich, Germany
{M11107418,syfang}@mail.ntust.edu.tw

ABSTRACT

Optical networks-on-chips (ONoCs), which adopt optical waveguides, microring resonators (MRRs), and the wavelength division multiplexing (WDM) scheme to transmit optical signals, serve as promising solutions for integrating multi- and many-core systems to provide high-bandwidth, low-latency, and low-power on-chip communication. To minimize the insertion loss of a wavelength-routed ONoC (WRONoC) during physical implementation, existing studies either adopt conventional standard cell placement techniques or maximally avoid waveguide crossings; however, all of them ignore the fact that the critical path suffering from the maximum insertion loss dominates the overall power efficiency and system performance. In this work, we propose CPONoC, a critical-path-aware physical implementation tool for WRONoCs. Different from existing studies, CPONoC focuses on minimizing the insertion loss of the critical path using an iterative crossing-aware force-directed method, and it is compatible with different representative logic schemes and input configurations. Compared to the state-of-the-art design automation tools, CPONoC achieves an average reduction of 9.6% in maximum insertion loss.

1 INTRODUCTION

As technology advances, the density of integrated transistors continues to increase, which contributes to the development of multi- and many-core systems. To facilitate on-chip communications among cores and memory components, optical networks-on-chip (ONoC) has been proposed as an emerging interconnect solution for high communication bandwidth, low power, low latency.

There are two types of ONoCs: control-network-based ONoCs and wavelength-routed ONoCs (WRONoCs). The control-network-based ONoC, referred to as *active ONoCs*, use a control network to modulate the refractive index of microring resonators (MRRs), dynamically tuning their resonance to control signal routing. In contrast, the WRONoCs, also known as *passive ONoCs*, set up collision-free signal paths for all master-slave pairs in advance by allocating specific wavelengths to each communication path. WRONoCs are usually preferred over active ONoCs especially in high-performance computing (HPC) pursuing extremely high performance, high bandwidth, and low power.

ONoCs apply wavelength-division multiplexing (WDM) to accommodate signals of different wavelengths in a single waveguide. Additionally, signals on the same waveguide can be directed to different destinations by employing microring resonators (MRRs) or photonic switching elements (PSEs) composed of MRRs. An MRR is a circular waveguide coupled to straight waveguides and operates in two states: in the on-state, matching wavelengths resonate within the ring and drop into the output waveguide; in the off-state, non-matching wavelengths pass straight through the input waveguide without coupling, as shown in Fig. 1(a). In Fig. 1(b), a 2×2 PSE is illustrated, consisting of a pair of MRRs. When the wavelength of the input signal is λ_i and

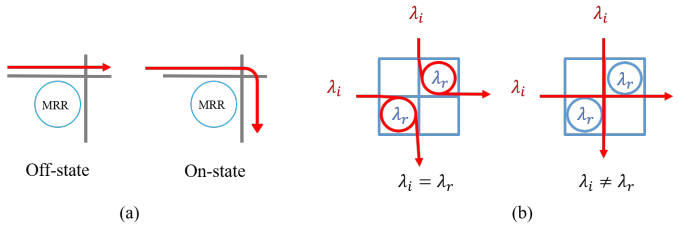


Figure 1: Microring resonators (MRRs) and Photonic switching elements (PSEs). (a) When an optical signal resonates with an MRR (on-state), it drops; otherwise (off-state), it passes. (b) Whether an input signal passes or drops in a PSE is determined by its resonance with the MRRs.

matches the resonant wavelength λ_r of the MRR, the signal is coupled and undergoes a 90-degree deflection, known as a *drop* operation. Conversely, when λ_i does not match λ_r, the input signal passes directly through the PSE, known as a *cross* operation.

The design of WRONoCs is typically achieved with two main steps. The first is topological design, which focuses on determining the configuration of waveguides and MRRs, allocating wavelengths between different signals, planning paths, and interconnecting network components. The second step is physical implementation, which emphasizes the placement of network components and the routing of the waveguides. So far, several topological designs have been proposed. The λ-router [1], built by cascading 2×2 PSEs, stands out thanks to its efficient handling of data traffic without causing congestion. The GWOR [2] and Light [3], on the other hand, propose to construct larger topologies using basic building blocks. For GWOR [2], the positioning of its interfaces aligns well with the input and output ports of memory controllers and hubs, resulting in excellent performance. On the other hand, Light [3] significantly reduces the number of MRRs, which not only lowers power consumption but also minimizes the impact of crosstalk, providing distinct advantages.

Concerning physical implementation, multiple design automation approaches have been developed to automate the implementation process. PROTON [4] is the first automated placement and routing tool for 3-dimensional (3D) ONoC. It adopts conventional nonlinear cell placement and maze routing techniques, approximates waveguide crossings during PSE placement, and allows waveguide crossings during routing with additional costs. PROTON+ [5] is an extension of PROTON, which improves the crossing approximation function during the placement stage and introduces a new net order during the routing stage, resulting in outstanding performances in terms of insertion loss. Platon [6] applies a force-directed placement method to enhance the overall efficiency, albeit with slight performance degradation. PlanarONoC [7] aims to find a crossing-free physical implementation result to prevent any insertion loss caused by waveguide crossings. The approach relies on a Hamiltonian path-finding process, which limits its generality considering that the derived layout strongly relies on the quality of the found Hamiltonian path. ToPro [8] utilizes a dynamic pushing mechanism to eliminate additional crossings and strategically chooses shorter paths when detours occur, leading to the currently best results for some specific topologies.

Despite the presence of excellent design automation approaches, the physical implementation still faces numerous challenges. First of all, some of these approaches are only applicable to specific topologies or specific positions of components with fixed port arrangements. For

*This work was partially supported by TSMC, Synopsys, and NSTC of Taiwan under Grant No's NSTC 113-2927-I-011-502, 113-2640-E-002-001, 113-2640-E-006-001, and 113-2222-E-011-005-MY3.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
ASPDAC '25, January 20–23, 2025, Tokyo, Japan
© 2025 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-0635-6/25/01…$15.00
https://doi.org/10.1145/3658617.3697727

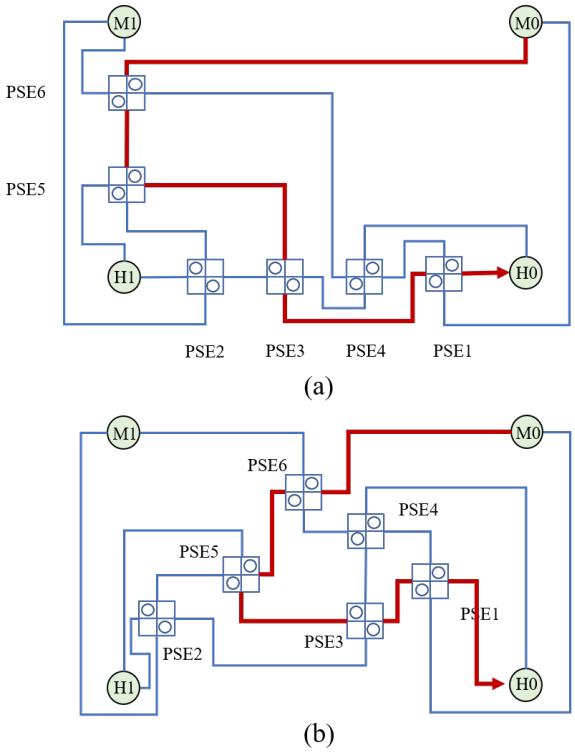


Figure 2: An example motivating this work. (a) A layout generated by PlanarONoC where the worst-case path is routed around the boundary. (b) A better PSE placement and waveguide routing result with a short worst-case path.

example, ToPro is not applicable to λ-routers, and PlanarONoC can hardly handle communication nodes at irregular positions. Moreover, existing approaches often fail to strike a good trade-off between waveguide crossing number and waveguide length. For example, to achieve crossing-free layouts, one may need to tolerate extremely long waveguide detours that aggravate the insertion loss, while the wave-guide might be significantly shortened when a few crossings can be allowed. More importantly, existing approaches overlook certain worst-case scenarios in their optimization framework. For instance, as a part of the strategy to eliminate waveguide crossings, PlanarONoC arranges components sequentially around the perimeter of the placement area, as shown in Fig. 2(a), which can lead to a long critical path that spans almost the entire boundary, while other PSE arrangements shown in Fig. 2(b) could significantly shorten the critical path.

In order to tackle the issues mentioned above, this paper introduces a physical implementation tool, *CPONoC*, which is designed to handle various types of topologies and take worst-case scenarios into account during optical component placement and routing for power optimization. The main features and contributions of CPONoC are summarized as follows:

- This work proposes the first automatic placement and routing approach considering insertion loss minimization for critical paths based on a force-directed algorithm.
- The approach has outstanding generality, making it applicable to various topologies that already exist and will emerge in the future.
- Experimental results show that the proposed method can averagely reduce the maximum insertion loss by 9.6% compared to the state of the art.

The rest of the paper is organized as follows: Section 2 introduces the background of this work. Section 3 presents the proposed placement and routing algorithms. Section 4 discusses the experiment results, and a brief conclusion is given in Section 5.

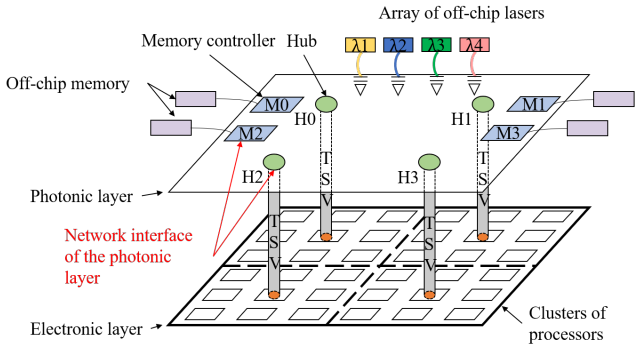


Figure 3: A 3-D stacked architecture for a multi-core system.

2 PRELIMINARIES

This section introduces the preliminaries of this work, covering the 3-D stacked architecture in Section 2.1, WRONoC design constraints in Section 2.2, and power evaluation metrics in Section 2.3.

2.1 3-D Stacked Architecture

A 3-D stacked architecture for multi- or many-core systems is shown in Fig. 3, which comprises an optical layer for high-speed data communication and an electronic layer for data processing and computing. These layers are interconnected using Through-Silicon Vias (TSVs), which enable efficient vertical communication and integration between the two layers. The electrical signals transmitted from a cluster of processors on the electronic layer have a dedicated gateway to a hub on the optical layer, and an optical signal is transmitted between two hubs or between a hub and a memory controller of an off-chip memory, e.g., a dual in-line memory module (DIMM). Note that for a given 3-D architecture, the positions of hubs and controllers are usually specified and cannot be changed during the physical implementation.

2.2 Topological and Physical Design of WRONoCs

In a WRONoC, the topology refers to how optical paths are organized and interconnected to realize data transmission. WRONOC typically uses MRRs for optical signal routing. During the design phase, MRRs are configured to determine the path and connectivity of each optical link. The topology design in WRONOC aims to maximize system performance and minimize power consumption, supporting high-speed and efficient optical communication.

The physical implementation of WRONoCs determines the positions of PSEs or MRRs during placement and routes the waveguides to minimize the maximum insertion loss required among all the signal paths. Typtically, two constraints need to be satisfied. First, the positions of the memory controllers and hubs are fixed and cannot be changed. Second, the input and output ports of the same communication node, i.e., a hub or a memory controller, should be placed next to each other during the physical implementation.

2.3 Power Consumption

The insertion loss of each path p in an ONoC is mainly composed of (1) propagation loss, (2) crossing loss, (3) drop loss, and (4) bend loss. According to [9], the insertion loss of p, $il(p)$, can be computed as follows:

$$il(p) = 1.5\frac{db}{cm} \times L + 0.15db \times C + 0.5db \times D + 0.005db \times B, \quad (1)$$

where L, C, D, B separately represent the waveguide length, the number of waveguide crossings, the number of MRR drops, and the number of waveguide bends in p. Various processes today affect the calculation of insertion loss, primarily influencing its parameters. The maximum insertion loss il_{max} of a WRONoC is then computed as follows:

$$il_{max} = \max_{p \in P} il(p), \quad (2)$$

where P is the set of all signal paths. We refer to the path with il_{max} as the critical path of the WRONoC.

In the optimization of WRONoCs, the number of drops depends on the signal path routing determined in the topology design and is irrelevant with the physical implementation. Besides, considering the small impact of bends on the insertion loss, the physical implementation usually ignores them but focuses on reducing waveguide lengths and crossings.

3 CRITICAL-PATH-AWARE PHYSICAL IMPLEMENTATION FLOW

This section presents the proposed automatic placement and routing tool CPONoC for WRONoCs that is applicable to various existing and emerging topologies. CPONoC aims to minimize maximum insertion loss by directly reducing the waveguide length and crossings of critical paths. The overall algorithm flow is described in Section 3.1, with the major steps detailed in the subsequent subsections.

3.1 Algorithm Flow

Our research methodology can be broadly divided into four steps:

Step 1: Topology Categorization:

As introduced before, current WRONoC routers handle the input and output ports in their topologies differently. For example, the λ-router divides the input and output ports into two sides in the topology, while the GWOR and Light routers place them adjacent to each other. Since the difference will require different treatments for insertion loss optimization, we classify these routers into physical-aware and non-physical-aware categories based on their input and output port characteristics.

Step 2: Topology Preprocessing

The input topology is first transformed into a graph representation. To minimize the insertion loss caused by waveguide crossings, we adopt the idea of planar embedding similar to that in PlanarONoC. For a physial-aware topology, the topology itself should be planar or can be regarded as planar by treating the crossings in the topology as graph vertices. For a non-physical-aware router, additional preprocessing is required. In addition, we perform node clustering based on their connectivity, which not only accelerates the optimization process but also improves the results.

Step 3: Force-Directed Placement:

We propose a force-directed algorithm to determine the placement of all vertices. In the iterative node movement process, we consider the impact of a move on the critical path to minimize the maximum insertion loss. Additionally, we address the congestion issues to facilitate the subsequent routing stage.

Step 4: Routing:

A simple L-shape routing is adopted to connect the edges among vertices, which contributes to simple waveguide routing results and thus better performances of optical signal transmission.

In the following subsections, we introduce each step with comprehensive details and examples.

3.2 Step 1: Topology Categorization

Currently, there are two distinct types of topologies for handling input and output interfaces of memory controllers and data transmission hubs. The first type separates the input and output ports to the two sides in a logic scheme, as the 4×4 λ-router shown in Fig. 4(a). Since each pair of an input and an output belongs to the same hub/controller, the input and output terminals are actually physically adjacent. Therefore, many crossings could easily be caused by directly merging the input and output ports, as illustrated in Fig. 4(b). The second type of topology assumes the input and output ports of the same hub/controller adjacent to each other, as the 4×4 GWOR shown in Fig. 4(c).

It is clear that the second topology type is more friendly to physical implementation since it considers the nature of physical implementation. It is also the reason why some existing work on physical implementation can easily tackle the topologies of the second type using straightforward heuristics. Since the two different types of topologies require different treatments in the following steps, we respectively regard the first and the second types as physical-aware and non-physical-aware topologies.

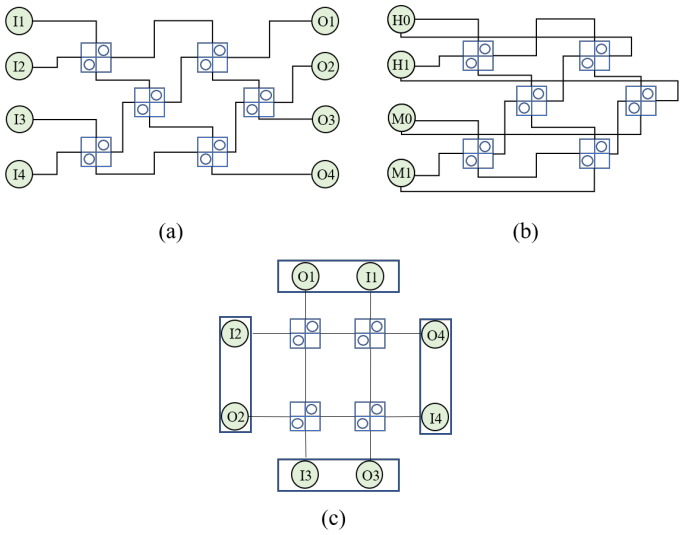


Figure 4: Two types of topologies. (a) A non-physical-aware topology separates the inputs and outputs into two sides. (b) Extra crossings can be easily generated when merging the input and output ports. (c) A physical-aware topology assumes the input and output of a hardware device are in adjacent locations.

3.3 Step 2: Topology Preprocessing

Given an input WRONoC topology, We first model the topology as a *connection graph*, where N_{fixed} is a set of nodes representing the fixed components such as hubs and memory controllers, and $N_{moveable}$ is a set movable nodes such as PSEs or other switching elements. Collectively, $N = N_{fixed} \cup N_{moveable}$. An edge e belonging to the edge set E is constructed between two nodes if the two corresponding components are connected in the topology.

For a non-physical aware router, straight-line embedding [10] is first applied to derive a crossing-free initial solution. After that, we perform node clustering based on the embedding result. As shown in Fig. 5(a), three nodes forming a 3-clique in the straight-line embedding are clustered and contracted into one node. Note that the nodes of fixed components are excluded from the clustering process, and once nodes are clustered, they cannot be further merged with others. This process continues iteratively until no more nodes can be clustered, as shown in Fig. 5(b). The major reason for adopting node clustering is to reduce problem size and thus enhance the efficiency of the problem-solving process. In addition, clustering every three highly connected nodes can contribute to crossing minimization in the placement stage. For a physical-aware router, since the topology is planar or almost planar, we directly transform it into an initial planar embedding. After that, the same node clustering procedure is applied as that for a non-physical-aware router.

3.4 Step 3: Force-Directed Placement

We develop our placement method based on the Fruchterman-Reingold force-directed algorithm [11]. This algorithm models the edges of a graph as springs and the nodes as support points, forming a mechanical system and seeking force equilibrium. Using a spring system helps identify which nodes should be positioned closer together and farther apart. However, this model does not directly account for crossings. To make the existing approach suitable for our application, we adjust the attractive and repulsive forces among vertices to facilitate adequate moves of movable components. In addition, in each movement iteration, we calculate the insertion loss of each signal path, identifying the paths with the maximum insertion loss as the critical paths. Therefore, our

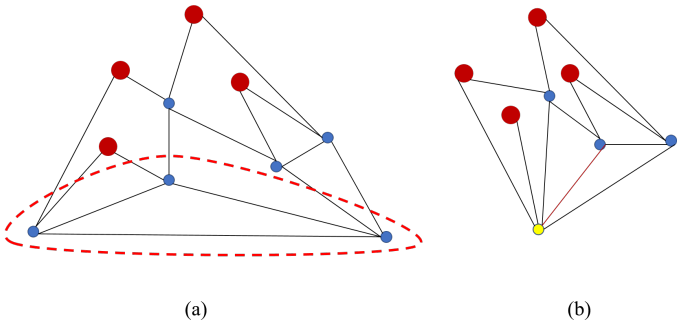


Figure 5: Clustering nodes based on their connectivity. (a) A straight line embedding of the 4x4 λ-router. (b) Clustering three nodes that form a 3-clique.

objective is different from that in [11], and the major task is to minimize the insertion loss along these critical paths.

In the following, we first explain some crucial elements of the proposed force-directed and critical path-aware placement approach. After that, the overall placement process will be summarized with a pseudo-code.

3.4.1 Force Computation. The force-directed algorithm balances attractive and repulsive forces between vertices to achieve force equilibrium. The magnitudes of the attractive and repulsive forces imposed on movable components are dependent on the area of the pre-specified movable region. For example, in a larger zone, with more available area, the repulsive force can be larger and the attractive force can be smaller. We define the movable region as the *zone of placement*, and it is determined by the bounding boxes of all vertices in N_{fixed}.

The placement procedure begins with placing the vertices in N_{fixed} in their fixed positions, and the remaining vertices in $N_{moveable}$ are randomly placed within the zone. After that, we define a parameter k as follows:

$$k = \sqrt{\frac{\text{zone area}}{|N|}}, \tag{3}$$

which can be interpreted as the congestion level within the space. Therefore, k should be inversely proportional to the attractive force and proportional to the repulsive force.

In addition, attractive forces act only between two vertices connected by an edge to minimize waveguide lengths. In contrast, repulsive forces are exerted between any pair of vertices to evenly distributed movable components, which facilitate the following waveguide routing. We define d as the Manhattan distance between two vertices in N. When defining the attractive force, we aim for a pair of nodes connected by an edge to be closer, especially if the edge belongs to some critical path. To enhance critical path optimization, the attractive force is multiplied by a weight that increases with the number of iterations. Therefore, the calculation formula for the attractive force, F_a, magnitude is given by:

$$F_a = w_i \times \left(\frac{d^2}{k}\right), \tag{4}$$

where w_i is the weight of the i-th iteration. On the other hand, repulsive forces primarily prevent vertices of fixed and non-fixed components from overlapping with each other and ease waveguide routing. Thus, we design the formula of repulsive forces, F_r, as follows:

$$F_r = \frac{k^2}{d}. \tag{5}$$

3.4.2 Critical Path-aware Switching Component Movement. Using attractive and repulsive forces, we determine the directional motion vectors for all nodes. Because all forces are pairwise in nature, for two nodes in $N_{movable}$, their attractive forces are added up. On the other hand, when one of the two nodes of an edge is in N_{fixed}, we do not move its position but add its attractive force to the other movable and connected node.

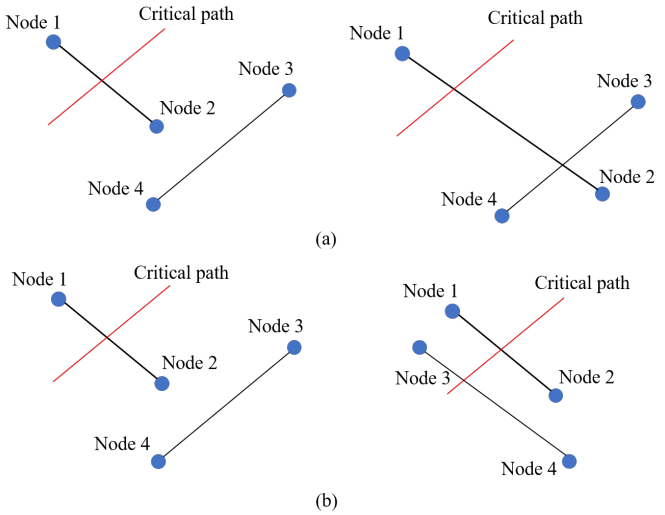


Figure 6: Critical path-aware node movement. (a) Moving Node 2 does not affect the critical path. (b) Moving Node 3 impacts the critical path.

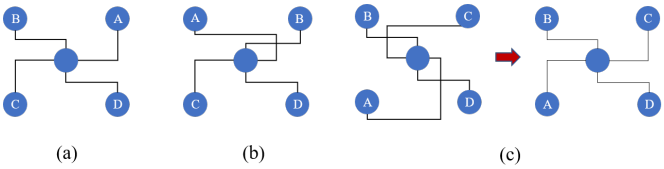


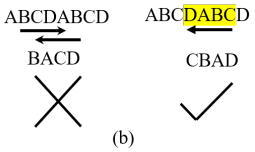
Figure 7: Additional crossing generation and resolving. (a) The connection order of the center PSE in the counter-clockwise direction is $< A, B, C, D >$, determined based on the relative component locations (top-right, top-left, bottom-left, and bottom-right). (b) The connection order is $< B, A, C, D >$, causing an extra crossing. (c) The connection order is $< C, B, A, D >$, the two extra crossings can be resolved by flipping the PSE.

After calculating the motion vectors for all nodes according to the forces, the new position of each node can be determined. However, moving nodes by merely considering the forces may worsen the maximum insertion loss caused by critical paths. For example, moving Node 2 in Fig. 6(a) does not affect the critical path, and thus it is usually feasible to be conducted. However, as shown in Fig. 6(b), moving Node 3 will cause an additional crossing on the critical path, and thus, it may greatly increase the maximum insertion loss. Therefore, we check if the increase in the maximum insertion loss Δil_{max} exceeds a threshold value using Equation (2). If it does, the node move is forbidden. In addition, it is also undesirable to cause too many additional crossings due to a node move, even if they are not on critical paths. This is because every path may become a critical path during the optimization process, and additional crossings are difficult to remove once created.

3.4.3 Connection Order Consideration. It can be observed that after modeling the topology of an optical router as a graph, the graph only keeps the connection relationship among vertices while losing the connection order information of switching components, which may lead to unpredictable waveguide crossings in the subsequent steps of moving switching components. Fig. 7 illustrates an example, where Fig. 7(a) shows a PSE in the center and connects counter-clockwise to four neighboring components with the connection order $< A, B, C, D >$. With forced-directed moves, an extra crossing may occur if the positions of A and B are swapped, as illustrated in Fig. 7(b). This crossing cannot be detected in the connection graph, because each node in a general graph does not record/limit the connection order of its incident edges. In contrast, Fig. 7(c) illustrates another situation after PSE

ABCD	ABDC	ACBD	ACDB	ADBC	ADCB
BACD	BADC	BCAD	BCDA	BDAC	BDCA
CABD	CADB	CBAD	CBDA	CDAB	CDBA
DABC	DACB	DBAC	DBCA	DCAB	DCBA

(a)



(b)

Figure 8: The feasibility of connection order. (a) Enumerated connection orders. (b) Whether a connection order causes additional crossing can be checked with substring comparison.

movement, where two crossings are caused, while they can be resolved by flipping the center PSE.

To determine whether the move of a switching component could cause unresolvable crossings, we check the *connection order* of the component before and after the move. When the connection order is changed after a component move, it can be guaranteed that no crossing occurs if the connection order can be obtained by rotating or/and flipping the component. For example, the connection in Fig. 7(b) is $< B, A, C, D >$, which cannot be derived by rotating and flipping the center PSE. In contrast, the connection order $< C, B, A, D >$ is actually a reversed and rotated order of $< A, B, C, D >$, and thus the two crossings in Fig. 7(c) can be resolved by flipping the center PSE.

Fig. 8(a) lists all the permutations of the connection order for the PSE that we consider, and the feasible connection orders are highlighted. To check whether a connection order is feasible or not for any switching elements with varied numbers of ports, a systematic method is proposed. We first regard the connection order before a component move as a string and replicate it to become twice as long, which is denoted as S_1. The connection order after the component move is also regarded as a string and denoted as S_2. After that, we check whether S_2 is a substring of S_1 or a substring of the reverse of S_1. If it is, the change in the connection order will not cause any additional crossing; otherwise, crossing-free movement is not guaranteed. For example, S_1 is $< A, B, C, D, A, B, C, D >$ for the instance in Fig. 7(a), and S_2 is respectively $< B, A, C, D >$ and $< C, B, A, D >$ in Figs. 7(b) and (c). As illustrated in Fig. 8(b), since $< B, A, C, D >$ is not a substring of both S_1 and the reverse of S_1, it causes additional crossings. In contrast, since $< C, B, A, D >$ is a substring of the reverse of S_1, it is determined as a feasible connection order. The checking process can be done by linearly scanning S_1 and thus can be regarded as a constant-time operation.

3.4.4 Congestion Avoidance. To prevent the subsequent routing step from causing additional crossings due to routing congestion, we divide the zone of placement into a grid and ensure that the number of nodes contained in each grid cell after node movement does not exceed a user-defined value γ. Otherwise, the node will be moved to a neighboring grid cell that minimally impacts il_{max}, as shown in Fig. 9.

3.4.5 Overall Placement Flow. Algorithm 1 outlines the overall placement procedure. Lines 1-2 perform initialization. In each optimization iteration, Line 4 calculates the attraction and repulsion forces for each node, and Line 5 determines the new position of each node according to its motion vector. Lines 6-20 try to move a node at a time. Whether a node is moved to its new position is determined by its impact on the maximum insertion loss (il_{max}), the number of crossings (ΔC), and whether the target grid is congested or not. If the move of a node benefits or does not greatly deteriorate the overall physical implementation result, it will be accepted.

3.5 Step 4: Routing

Having an optimized placement of movable components, waveguide routing is conducted by applying simple L-shaped routing. We first separate each clustered node generated in the pre-processing stage into

2	3	4
2	5	1
1	2	3

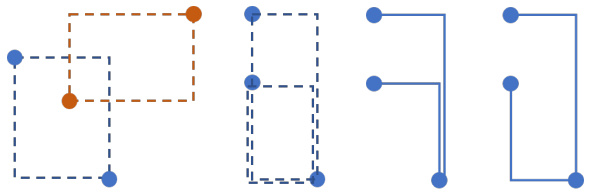
+0.1	+0.2	+0.5
+0.1	+0	+0.15
+0.3	+0.05	+0.2

Figure 9: When congestion occurs, the concerning node moves to a nearby grid cell considering the impact on insertion loss. (a) Numbers of nodes contained in the grid cells. (b) Quantified impact on insertion loss when moving a node to a nearby grid cell.

Algorithm 1 Critical Path-aware Movable Component Placement

Input: *A connection graph derived from Step 2*
Output: *Optimized placement locations of the movable components*

1: Construct the placement zone according to fixed nodes
2: Randomly place movable nodes and record their positions in a vector V_{pos}
3: **for** a fixed number of iterations **do**
4: Calculate the attraction and repulsion forces for all nodes according to V_{pos}
5: Calculate the new positions for all nodes based on the forces and record them in another vector V'_{pos}
6: **for** each node **do**
7: Determine the connection orders before and after the move
8: Calculate Δil_{max} based on its positions in V_{pos} and V'_{pos}
9: Calculate ΔC with straight lines and connection orders
10: **if** $\Delta il_{max} < \alpha$ && $\Delta C < \beta$ **then**
11: Calculate grid congestion
12: **if** the congestion of the target grid cell $< \gamma$ **then**
13: Move the node to the new position
14: Update V'_{pos}
15: **else**
16: Move the node to a non-overflow neighboring grid cell with the least Δil_{max}
17: Update V'_{pos}
18: **end if**
19: **end if**
20: **end for**
21: $V_{pos} \leftarrow V'_{pos}$
22: **end for**



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

Figure 10: L-shape routing. (a) The boundary is not completely enclosed. (b) The boundary is enclosed. (c) Shift overlapping line segments to eliminate the overlap. (d) Choose different directions to avoid overlap.

three nodes and connect them without crossing. After that, we route the waveguides on critical paths with L shapes. Finally, we route the remaining waveguides by generating as few crossings as possible and avoiding generating crossings with the waveguides on critical paths. We use some heuristics that have been widely adopted in many routers. For example, for the two nets shown in Fig. 10(a), the blue net needs

Table 1: Experimental results for 8×8 routers with different hardware configurations. "il_{max}", "C", "L", and "t" separately denote the maximum insertion loss, the number of crossings, the total waveguide length, and the runtime in seconds.

		Pos(a)				Pos(b)				Pos(c)				Pos(d)			
Topology	Work	il_{max}	C	L	t	il_{max}	C	L	t	il_{max}	C	L	t	il_{max}	C	L	t
λ-router	Proton+	7.9	27.0	20817.0	146.9	7.8	29.0	18513.0	136.5	7.5	31.0	17748.0	135.4	6.6	27.0	12726.0	134.0
	PlanarONoC	5.23	7.0	24140.0	0.3	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
	ToPro	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
	Our	**4.8**	17.0	11755.0	7.2	**5.2**	15.0	16600.0	7.8	**5.3**	18.0	13801.0	8.2	**4.7**	16.0	11958.0	6.2
GWOR	Proton+	8.4	38.0	13000.0	77.1	8.5	37.0	14661.0	7.7	8.0	36.0	12528.0	90.7	8.1	35.0	13806.0	79.0
	PlanarONoC	6.38	10.0	28620.0	0.1	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
	ToPro	3.8	8.0	14200.0	0.19	5.0	8.0	22200.0	0.15	4.5	8.0	18400.0	0.14	4.0	10.0	13500.0	0.17
	Our	**3.6**	11.0	9851.0	10.2	**4.2**	10.0	14528.0	11.9	**4.0**	13.0	10323.0	11.4	**4.0**	12.0	11247.0	9.7
Light	Proton+	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
	PlanarONoC	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
	ToPro	5.5	12.0	21000.0	0.19	6.4	6.0	33300.0	0.2	5.2	12.0	19000.0	0.15	4.3	12.0	13500.0	0.07
	Our	**4.4**	15.0	11131.0	8.9	**4.7**	11.0	17249.0	8.5	**4.2**	14.0	10639.0	8.3	**3.8**	12.0	10142.0	8.2

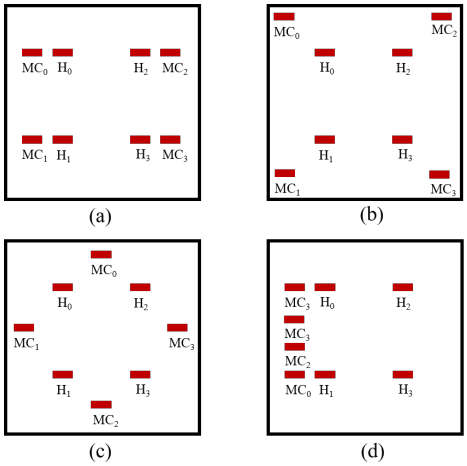


Figure 11: Four different positions of the memory controller.

to route the lower L-shaped path to avoid crossing with the red net. In Fig. 10(b), for the two nets whose bounding boxes are overlapped at their boundaries, the segments need to be slightly shifted to eliminate waveguide overlap, as shown in Fig. 10(c), or route them with adequate directions, as shown in Fig. 10(d). Some additional crossings could be generated during L-shaped routing for some nets that are originally non-crossing when they are represented by straight lines. In such cases, we apply monotonic routing for these nets to find their non-crossing routing paths.

4 EXPERIMENTAL RESULTS

We implemented CPONoC in the C++ programming language on a 2.00 GHz Linux workstation with 56 GB memory. Three state-of-the-art works reporting the best physical implementation results are compared by conducting their binaries, namely Proton+ [5], PlanarONoC [7], and ToPro [8]. Three ONoC topologies including λ-router [1], GWOR [2], and Light [3] are adopted as the benchmarks, and four hardware configurations with different positions of fixed ports are considered, which are separately denoted as Pos(a), Pos(b), Pos(c), and Pos(d) and shown in Fig. 11. The number of optimization iterations in the force-directed placement method is set to 100. α and β representing the thresholds for Δil_{max} and ΔC (see Algorithm 1) are respectively set to 0.5 and 2. We impose stricter requirements for crossings because additional waveguide crossings will become a bottleneck for maximum insertion loss minimization in later optimization iterations. The grid cell congestion threshold γ introduced in Section 3.4.4 is set to 5 in our experiments.

The experimental results are shown in Table 1, where "il_{max}" represents the maximum insertion loss, "C" represents the number of crossings, "L" represents the total waveguide length, and "t" represents runtime in seconds. It can be observed in Table 1 that for the λ-router, ToPro has no data because it cannot tackle non-physical-aware topologies. PlanarONoC achieves better results compared to Proton+, while it is hardly applied to various hardware configurations without additional manual effort. Compared to PlanarONoC for Pos(a), CPONoC achieves a 8% reduction in maximum insertion loss by focusing on critical paths during the force-directed optimization process. Compared to the layouts synthesized by Proton+, the layouts produced by CPONoC consistently reduce the number of crossings and waveguide lengths, resulting in an average reduction of 32.25% in maximum insertion loss. For the two physical-aware topologies (GWOR and Light), ToPro has the best physical implementation results among the three state-of-the-art works. Compared to ToPro, CPONoC achieves an average reduction of 10% in maximum insertion loss across the four positions for GWOR, and achieves 18.6% reduction for Light.

5 CONCLUSION

In this paper, we propose CPONoC, a physical implementation tool that categorizes optical routers and applies customized treatments to accommodate different topologies. Instead of aiming to eliminate all waveguide crossings, CPONoC minimizes the insertion loss of critical paths by seeking a good trade-off between the number of crossings and the waveguide length. The experimental results show that CPONoC outperforms three state-of-the-art design automation approaches given different input topologies and positions of ports.

REFERENCES

[1] M. Briere, B. Girodias, Y. Bouchebaba, G. Nicolescu, F. Mieyeville, F. Gaffiot, and I. O'Connor, "System Level Assessment of an Optical NoC in an MPSoC Platform," Proc. Design, Automation & Test in Europe Conference & Exhibition, 2007.

[2] X. Tan, M. Yang, L. Zhang, Y. Jiang, and J. Yang, "On a Scalable, Non-Blocking Optical Router for Photonic Networks-on-Chip Designs," Proc. Symposium on Photonics and Optoelectronics, 2011.

[3] Z. Zheng, M. Li, T.-M. Tseng, and U. Schlichtmann, "Light: A Scalable and Efficient Wavelength-Routed Optical Networks-On-Chip Topology," Proc. Asia and South Pacific Design Automation Conference, 2021.

[4] A. Boos, L. Ramini, U. Schlichtmann, and D. Bertozzi, "PROTON: An automatic place-and-route tool for optical Networks-on-Chip," Proc. IEEE/ACM International Conference on Computer-Aided Design, 2013.

[5] A. v. Beuningen, L. Ramini, D. Bertozzi, and U. Schlichtmann, "PROTON+: a placement and routing tool for 3d optical networks-on-chip with a single optical layer," ACM Journal on Emerging Technologies in Computing Systems, vol. 12, no. 4, article 44, 2015.

[6] A. v. Beuningen and U. Schlichtmann, "PLATON: A Force-Directed Placement Algorithm for 3D Optical Networks-on-Chip," Proc. ACM International Symposium on Physical Design, 2016.

[7] Y.-K. Chuang, K.-J. Chen, K.-L. Lin, S.-Y. Fang, B. Li, and U. Schlichtmann, "PlanarONoC: Concurrent Placement and Routing Considering Crossing Minimization for Optical Networks-on-Chip," Proc. ACM/ESDA/IEEE Design Automation Conference, 2018.

[8] Z. Zheng, M. Li, T.-M. Tseng, and U. Schlichtmann, "ToPro: A Topology Projector and Waveguide Router for Wavelength-Routed Optical Networks-on-Chip," Proc. IEEE/ACM International Conference On Computer Aided Design, 2021.

[9] J. Chan, G. Hendry, A. Biberman, and K. Bergman, "Architectural design exploration of chip-scale photonic interconnection networks using physical-layer analysis," Proc. Conference on Optical Fiber Communication, collocated National Fiber Optic Engineers Conference, 2010.

[10] M. Chrobak, and T. Payne, "A Linear-time Algorithm for Drawing a Planar Graph on the Grid," Information Processing Letters, vol. 54, no. 4, pp. 241–246, 1995.

[11] T. M. J. FRUCHTERMAN, and E. M. REINGOLD, "Graph Drawing by Force-directed Placement," Software, Practice & Experience, vol. 21, no. 11, pp 1129—1164, 1991.