TUM SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY

TECHNISCHE UNIVERSITÄT MÜNCHEN

Research Internship

Research Internship Report

Zhao Wei

TUM SCHOOL OF COMPUTATION,
INFORMATION AND TECHNOLOGY

TECHNISCHE UNIVERSITÄT MÜNCHEN

Research Internship

Research Internship Report

Exploration of high-throughput based on ring-structure for wavelength-routed optical networks-on-chips

Author:	Zhao Wei
Supervisor:	Prof. Dr.-Ing. Ulf Schlichtmann
Advisor:	Dr.-Ing. Tsun-Ming Tseng
Submission Date:	31.08.2022

TUM School of Computation,
Information and Technology

I confirm that this research internship is my own work and I have documented all sources and material used.

Munich, 31.08.2022 Zhao Wei

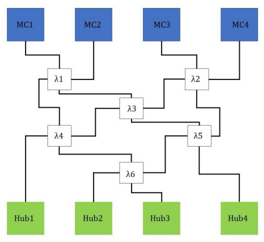
Abstract

Wavelength-routed optical network-on-chips (WRONoCs) is considered to be one of the most appealing multicore fields with increasing connectivity and scalability. The design of WRONoCs consists of many aspects. For example, physical layout, message routing, and waveguide routing. It offers high bandwidth and low latency by stacking an additional optical layer which avoids traffic contention as well as arbitration. However, due to the high connectivity of the WRONoCs, the problem of the losses such as crossing loss and propagation loss needs to be carefully addressed. This report proposes an algorithm named Hierarchical ring. It enables multiple sub-rings interconnected by rings on the upper hierarchy based on the conventional ring structure. Hierarchical ring considers the communication requirements among the nodes and it aims to minimize the propagation loss compared to the conventional method while maintaining the same bandwidth. Compared to conventional ring routing, Hierarchical ring has great potentials to adapt for future demands of application-specific implementations.

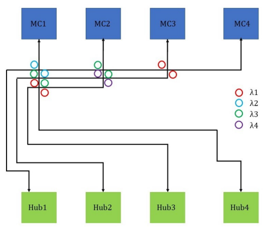
Contents

1 Introduction

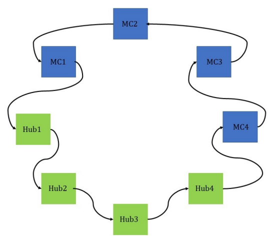
The arising interest in multicore starts with Moore's Law, named after Gordon E. Moore. In the early sixties, he observed that the number of transistors in an integrated circuit doubled every 24 months. However, power density becomes increasingly problematic with the increased frequency and microarchitecture improvements. The focus thus shifts towards the multicore systems. The design of different interconnect topologies determines the system performance as well as the latency and losses. WRONoCs with an additional optical layer are becoming more and more attractive as further bandwidth improvement on the electrical layer is hardly achieved while maintaining the same feasibility and scalability. The state-of-the-art WRONoCs topology approaches include λ-router [1], snake [2] and ring [3] (see Figure 1.1).



(a) λ-router



(b) snake



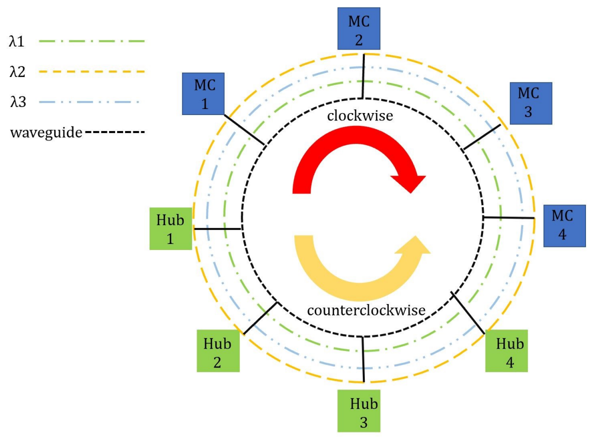
(c) ring

Figure 1.1: Different state-of-the-art topologies.

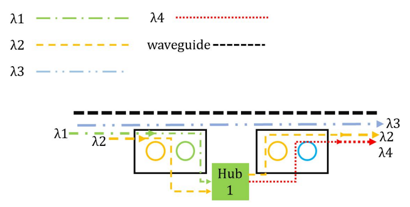
[4] illustrates the worst-case losses comparison between different topologies. The result shows that λ-router and snake mainly suffer from crossing loss (due to waveguide crossing) while the major loss of ring structure is the propagation loss which depends on the waveguide distance between the source and destination. Compared to other topology implementations, ring structure has the lowest loss down to 46% to 56% reduction by avoiding waveguide crosses and reducing the signal propagation distances with the use of clockwise and counter-clockwise directions. Moreover, this result could be further improved with better parameter value of propagation loss, thus making ring structure to become a promising topology for future WRONoCs. Another advantage of ring structure is that regardless of the distance between source and destination, its worst case loss is the lowest. This enables a larger number of cores implementations with high robustness.

2 Preliminary and Background

Ring is a contention-free network with high throughput. In this architecture, a wavelength is reusable for multiple communications on a single waveguide that greatly relaxes the constraints of the number of wavelengths. Figure 2.1 shows the physical structure as well as the conceptual view of the ring. One physical ring waveguide is shared among multiple virtual cycles (or rings) each assigning a wavelength for different sources and targets. Figure 2.1b shows the physical view of the optical network interface (ONI) of the ring. A microring resonator (MRR) is a ring-formed waveguide. The signal will resonate with MRRs and couple into MRR if the wavelength of the signal is an integer multiple of MRR's wavelength (see Figure 2.1b orange path), otherwise it will continue in its original direction without being impacted (see Figure 2.1b blue path).



(a) conceptual view of different wavelengths passing through the same waveguide



(b) physical view of ONI for reception and transmission

Figure 2.1: ring structure.

However, propagation loss is dominant in the ring structure. To date, there is no optimization approach regarding signal path length minimization as well as considering communication requirements in ring. In this report, I propose a nouveau topology: Hierarchical ring. Hierarchical ring aims to minimize the propagation loss while fulfilling communication requirements to the maximum extent by forming density-based clusters. Each cluster (see the light green rectangle in Figure 2.2) in my topology is designed to group the nodes whose communication requirements with each other are high. I then connect my clusters to form a large cycle (see the dark green dashed rectangle in Figure 2.2) that connects all the clusters to meet the demand of inter-cluster communication. My design flow consists of three stages:

(1) graph partition based on the communication requirements and the Euclidean distance between nodes using the DBSCAN (Density-based spatial clustering of applications) method. (2) clockwise and counter-clockwise weighted Hamiltonian path routing of each cluster based on the same features as (1). (3) global Hamiltonian path routing for connecting all the clusters.

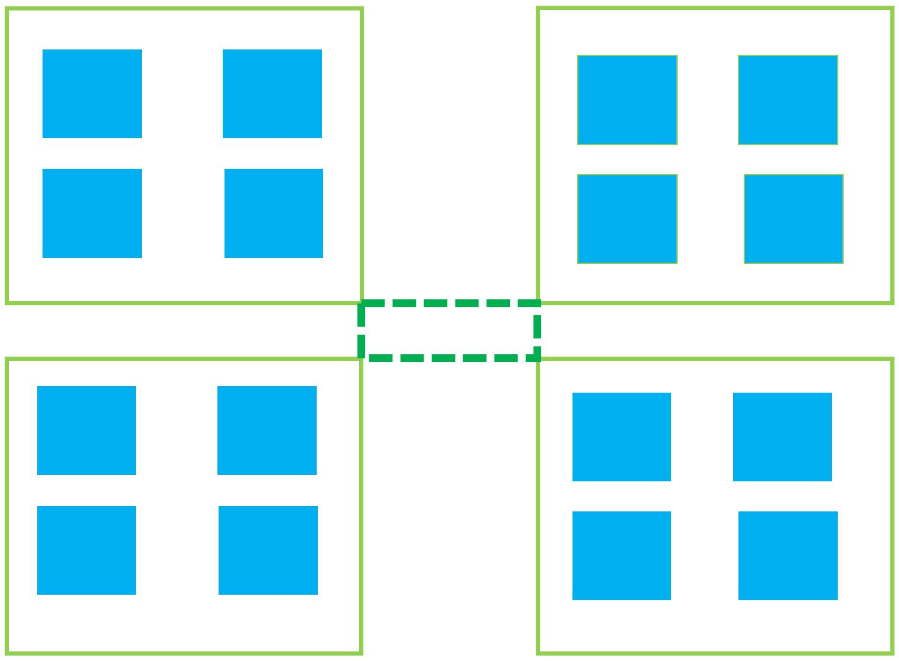


Figure 2.2: light green rectangles represents an example of partitioned result; dark green rectangle is a example of global routing.

2.1 DBSCAN method

DBSCAN is a graph partition method without defining a prior number of clusters. I define the value of parameters radius ε and minimum points *minPts* at the beginning. Three types of points are elaborated in DBSCAN: 1) core point: the number of reachable points is greater or at least equal to *minPts* given , 2) border point: reachable by at least one core point given , and 3) noise point: not reachable by any core point. To perform the algorithm, I initially pick an unprocessed point in the graph. If it is the core point then I mark all the neighboring points as border points of this point which will form a cluster, otherwise, I mark it temporarily as a noise point to see if it could be further updated to a border point of another cluster. Then I continue my process of repeating the first step until all the points have been visited once. It is noteworthy that DBSCAN is normally sensitive to the values of ε and *minPts*, and subtle differences may entail completely different partition sizes and partitioned points.

2.2 Hamiltonian path

A Hamiltonian path is a graph path that visits each vertex exactly once. If a Hamiltonian path exists and the starting vertex is the same as the ending vertex of the path, the resulting graph cycle is called a Hamiltonian cycle. So far, the weight of the edge in the Hamiltonian path is default and is assumed to be identical on every edge. Therefore, I take communication requirements and Euclidean distance between two nodes into the weight consideration, resulting in a weighted Hamiltonian cycle problem. The objective is to explore a weighted Hamiltonian cycle that fulfills all my requirements.

2.3 Integer linear programming

An integer linear programming (ILP) problem is a mathematical optimization or feasibility program in which the objective function and the constraints are both linear. The goal of using ILP in my model is to meet communication requirements and minimize the total propagation loss of the network. The variables in my model represent decisions. For example, '1' is when ILP decides to include an edge of the graph in my model. A standard form of ILP is 1) set the target functions, 2) subject the problems to a set of linear equations and constraints, and 3) obtain an optimal solution or non-feasible result. The advantage of the ILP problem is not only providing an optimal solution for my problem but also being more dynamic and flexible since I could add more features to the model without violating the previous constraints.

3 Proposed Algorithms and Models

In this section, I first provide an overview of my routing flow and then illustrate each part in detail.

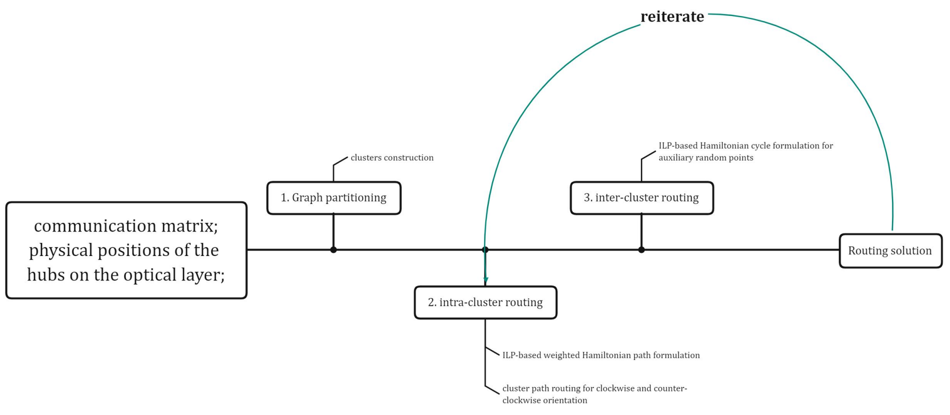

reiterate
clusters construction
1. Graph partitioning
ILP-based Hamiltonian cycle formulation for auxiliary random points
3. inter-cluster routing
communication matrix; physical positions of the hubs on the optical layer;
Routing solution
2. intra-cluster routing
ILP-based weighted Hamiltonian path formulation
cluster path routing for clockwise and counter-clockwise orientation

Figure 3.1: For pictures with the same name, the direct folder needs to be chosen.

Figure 3.1 summarizes my algorithm flow, consisting of three stages: 1) graph partition, 2) intra-cluster routing, and 3) inter-cluster routing. During the first stage, clusters containing different nodes are constructed. I further do the weighted Hamiltonian cycles in each cluster with aid of auxiliary random points. In the following, I route the random points to form the global Hamiltonian cycle. At last, I reiterate the model to get different optimization results based on different random points' locations, I then choose the locations with the minimum cost.

3.1 Graph partitioning

As I mentioned before, DBSCAN will be applied in Hierarchy ring model. The algorithm of DBSCSN performs clusters allocation given parameters: radius ε, minimum nodes *minPts*, communication requirements between different nodes, and their physical locations.
DBSCAN starts by picking up an arbitrary nodes P that has not been visited. One function deduces the number of neighboring nodes within the radius ε of the node P, and if it contains at least *minPts* nodes, a cluster C is forming. Otherwise, the point is labeled as noise. For each node Q of the neighboring nodes, I update the label of it as C indicating that it belongs to the same cluster as P. I then seek the neighboring nodes of Q. If the neighboring nodes of Q are larger or equal to *minPts*, I expand the neighboring node of Q to the neighboring

node of P and continue my iteration. A cluster is completely found as I go through all the neighboring nodes of P. Then, a new unvisited node is retrieved and will be processed. DBSCAN can be used with any distance function. In my model, I consider communication requirements between two points as an additional feature besides Euclidean distance.

3.2 Intra-cluster routing

As described in Section Preliminary and Background, few attempt to consider the aspect of nodes' traffic. Therefore, I intend to improve the aforementioned work by starting intra-cluster routing. However, if some of the nodes in the cluster are simultaneously working for intra-cluster communication and inter-cluster communication, a highly complex problem will be aroused against the designer's choice accordingly. I propose a heuristic approach to this problem which is to use auxiliary random points. I define four random points as the I/O port of each cluster to do the inter-cluster communication with other clusters. All the Hamiltonian paths must use these four points as the starting point and the ending point of each cluster. To further minimize the propagation loss on the optical layer, I establish two routing directions in the model: the clockwise and the counter-clockwise Hamiltonian path routing. I ensure the distinction between these two directions by using residual networks [5]. I construct a residual network, denoted G (V, E), where V denotes the nodes and E edges representing whether the communication exists between the nodes. In the beginning, the number of available capacities on the direct edge (u,v) and (v,u) are the communication requirements between u and v. If one direction is determined to pass edge (u,v), the amount of available capacity of this edge will become zero, leading to prohibiting the same routing path with the other direction and thus two distinct paths are guaranteed.

I formally define the optimization problem for intra-cluster routing as follows.

3.2.1 Input data

- CM: A square communication matrix $C_{i,j} \in R^{N\times N}$ with N being the sum of nodes and where each $C_{i,j}$ represents the communication requirement between node i and node j. In my test cases, we apply WRONoC applications with hubs and memory controllers. Each hub represents a node and each memory controller represents a node as well. $C_{i,j} = 0$ indicates there is no communication between node i and node j since I omit the communication in one hub (e.g. $C_{i,i} = 0$) as well as the communication between their memory controllers.

$$\begin{pmatrix} 0 & C_{H1,H2} & C_{H1,MC1} & C_{H1,MC2} \\ c_{H1,H2} & 0 & C_{H2,MC1} & C_{H2,MC2} \\ C_{H1,MC1} & C_{H2,MC1} & 0 & 0 \\ C_{H1,MC2} & C_{H2,MC2} & 0 & 0 \end{pmatrix}$$

- Physical positions of the hubs and memory controllers on the optical layer.

- Partitioned results of DBSCAN in the form of arrays. Each array represents a cluster containing different hubs and memory controllers.

3.2.2 Sets, indices and variables

- Sets and Indices:

  i, j∈cluster C_n: indices and set of each cluster (n∈ [1, the number of partitioned clusters])

  n ∈Random points R_n: indices and set of random points (n∈ [1, 4])

  Pairings = (i, j) ∈ **C** × **C**: set of pairings within one cluster where **C** defines as the union of C_n and R_n ($C_n \cup R_n$)

  G= (**C**, **Pairings**): A graph where the set **C** defines the set of the vertices and the set **Pairings** defines the set of edges

  $d_{i,j} \in R^+$: Distance between i and j, $\forall(i, j) \in$ **Pairings**

  $C_{i,j} \in R^+$: Communication flow from Random point i to j

- Variables:

  $k \in [1, 2]$: the indices of two Hamiltonian paths in cluster C_n

  $b_s^{k,v}$: the starting point in the k Hamiltonian path

  $b_e^{k,v}$: the ending point in the k Hamiltonian path

  $b_{edgemodel}^{m,i,j}$: the edge in model of Hamiltonian path k. Note that $b_{edgemodel}^{m,i,j} = 1$ implies that edge(i,j) will be adopted in the routing path k

3.2.3 Output data

Two routing paths of each cluster C_n

3.2.4 Constraints

- Constraints for random points in C_n:

  for each individual path k, I choose one random point out of four as the starting point:

$$\textstyle\sum_{v \in R_n} b_s^{m,v} = 1, \forall m \in k \tag{3.1}$$

  for each individual path k, I choose one random point as the ending point:

$$\textstyle\sum_{v \in R_n} b_e^{m,v} = 1, \forall m \in k \tag{3.2}$$

  To avoid signal conflicts, for each random point, I only map its function one time in C_n:

$$b_s^{1,v} + b_s^{2,v} + b_e^{1,v} + b_e^{2,v} = 1, \forall v \in R_n \tag{3.3}$$

- Mutual exclusion of two pairs of edges in each path in C_n

  if one random point Rn is selected for path k, then the edges could be connected to this random point. In contrast, if Rn is not selected, I remove all the edges connected to this random point:

$$b_s^{m,v} = \sum_{x \in C_n} b_{edgemodel}^{m,v,x}, \forall m \in k, v \in R_n \tag{3.4a}$$

$$b_e^{m,v} = \sum_{x \in C_n} b_{edgemodel}^{m,x,v}, \forall m \in k, v \in R_n \tag{3.4b}$$

- Constraints for weighted Hamiltonian paths

  if the random point is the starting point, there is no incoming edge:

$$b_s^{m,v} \leq 1 - \sum_{x \in C_n} b_{edgemodel}^{m,x,v}, \forall m \in k, v \in R_n \tag{3.5}$$

  if the random point is the ending point, there is no outgoing edge:

$$b_e^{m,v} \leq 1 - \sum_{x \in C_n} b_{edgemodel}^{m,v,x}, \forall m \in k, v \in R_n \tag{3.6}$$

  there is no edge connected between 2 random points:

$$b_{edgemodel}^{m,R_1,R_2} = 0, \forall m \in k, \forall (R_1, R_2) \in \textbf{Pairings} \tag{3.7}$$

  for each cluster point C_n in path k, the number of incoming edges is equal to the number of outgoing edges:

$$\sum_{x1 \in C_n} b_{edgemodel}^{m,x1,v} = \sum_{x2 \in C_n} b_{edgemodel}^{m,v,x2}, \forall m \in k, \forall v \in C_n \tag{3.8}$$

  for each cluster point C_n in path k, the number of incoming edges is equal to 1 and the same as the number of outgoing edges:

$$\sum_{x \in C_n} b_{edgemodel}^{m,x,v} = 1, \forall m \in k, v \in C_n \tag{3.9a}$$

$$\sum_{x \in C_n} b_{edgemodel}^{m,v,x} = 1, \forall m \in k, v \in C_n \tag{3.9b}$$

  an extra constraint is set up to prevent the sub-cycle, particularly for the routing path from (random points, cluster points) to (cluster points, random points):

$$\sum_{x \in C_n} b_{edgemodel}^{m,x,v} + \sum_{x \in C_n} b_{edgemodel}^{m,v,x} \leq 1, \forall m \in k, v \in R_n \tag{3.10}$$

  Residual network constraint: if one directed edge is chosen by one path in **Pairings**, I prohibit the selection of the same directed edge in another path in **Pairings**.

$$b_{edgemodel}^{1,v_1,v_2} + b_{edgemodel}^{2,v_1,v_2} \leq 1, \forall (v_1, v_2) \in \textbf{Pairings} \tag{3.11}$$

- Objective function

  I obtain the optimal result of two weighted Hamiltonian path by setting the following target function:

$$\min L = \sum_{m \in k} \sum_{(i,j) \in \textbf{Pairings}} d_{i,j} \cdot b^{m,i,j}_{edgemodel} - c_{i,j} \cdot b^{m,i,j}_{edgemodel} \tag{3.12}$$

3.3 Inter-cluster routing

To satisfy the inter-cluster communication concurrently, an ILP-based inter-cluster routing is adopted. Note there are several differences between intra-cluster routing and inter-cluster routing:

- The cluster hubs are not involved
- The weight of the edge relies exclusively on the physical locations of random points
- There will be no starting point nor ending point since a cycle will be formed instead of a path in the previous work (see 3.2 intra-cluster routing)

A similar formal model based on [6] will be described in the following:

3.3.1 Input data

Physical positions of the random points on the optical layer

3.3.2 Set,indices and variables

- Sets and Indices:

  $i, j \in R_n$

  Pairs $= (i, j) \in R_n \times R_n$

  $S \subset R_n$: A subset of the set of R_n

  $G = (R_n, \textbf{Pairs})$: A graph where the set R_n defines the set of the vertices and the set **Pairs** defines the set of edges

- Variables

  $d_{i,j} \in R^+$: Distance between i and j, $\forall (i, j) \in$ **Pairs**

  $x_{i,j} : x_{i,j} = 1$ if I decide to connect Random point i with Random point j. Otherwise, the decision variable is equal to zero.

3.3.3 Output data

A Hamiltonian cycle with all the random points

3.3.4 Constraints

- Constraints for the Hamiltonian cycle
  Symmetry Constraints: for each edge (i, j) is selected, then edge (j, i) is likewise selected:

$$x_{i,j} = x_{j,i}, \forall (i,j) \in \textbf{Pairs} \tag{3.13}$$

  Each vertex has one incoming edge and one outgoing edge:

$$\textstyle\sum_{(i,j)\in\textbf{Pairs}} x_{i,j} = 2, \forall i \in R_n \tag{3.14}$$

  Subtour elimination: These constraints ensure that for any subset of nodes of the set of Random points, there is no cycle:

$$\textstyle\sum_{(i\neq j)\in S} x_{i,j} \leq |S| - 1, \forall S \in R_n \tag{3.15}$$

- Objective function
  I obtain the optimal result of two weighted Hamiltonian paths by setting the following target function:

$$\min L = \textstyle\sum_{(i,j)\in\textbf{Pairs}} d_{i,j} \cdot x_{i,j} \tag{3.16}$$

It is worth mentioning that preventing multiple sub cycles in a Hamiltonian cycle needs an exponential number of these constraints. Consequently, I use a callback function to find violated subtours and add these constraints to the model as lazy constraints. The function *subtourelim* starts by retrieving the solution and then comparing the number of solution edges with the appropriate one. Since I have n random points, n edges should be counted in the solution if the optimization model correctly forms a Hamiltonian cycle. Therefore, if the number of edges in the solution is smaller than I expected, I invalidate this solution from the solution space and reiteratemymodel to search for new solution.

```
Input: solution model
Output: new lazy constraints
if valid solution then
    retrive solution sol = model.cbGetSolution()
    select the edges in sol if sol [i,j] > 0.5
end
tour=subtour(edges selected)
if len(tour) <len(Card(random Points)) then
    new lazy constraints are added
end
```

Algorithm 1: fonction subtourelim.

Eventually, I reiterate the intra-cluster routing and inter-cluster routing by applying different locations of random points each time. I terminate the process until the final result converges.

4 Results and Discussion

Hierarchy ring is implemented in Python and all the simulations discussed in this report were conducted on a 2.4GHz CPU. Firstly, I evaluate the result from DBSCAN. Secondly, I discuss the optimization results based on my model.

4.1 DBSCAN results

I initially implement the case with 4 hubs and 4 memory controllers then with 8 hubs and 8 memory controllers. I separately increase the value of radius ε and minimum points *minPts* to examine their impact on the graph partitioning.

As shown in Table 4.1 and Table 4.2, the first column represents the value of radius ε, the first row represents the value of defined *minPts* and each entry represents the number of partitions at given value ε and *minPts*. For each column of the tables, the number of partitions decreases with the increasing radius value. The value of ε is in terms of hundreds since I adjust the weight coefficient of communication requirements and Euclidean distance to average their impact. However, the number of partitions is not sensitive to the change of *minPts* (see each row in Table 4.1 and Table 4.2).The results suggests that the value of *minPts* is not critical to the graph partition in this case. Furthermore, comparison of the mean and variance of the two cases are shown in Table 4.3. The results show that the contribution of the number of *minPts* is insignificant in both cases. Since my model is currently applied within the aforementioned scope, the values of radius ε and minimum hubs *minPts* are set as 700 and 3 for 4-cores, which are the mediums of the available partition numbers.

Table 4.1: The impact of *minPts* on the number of partitions in different values of ε in architecture 4Hub×4MC.

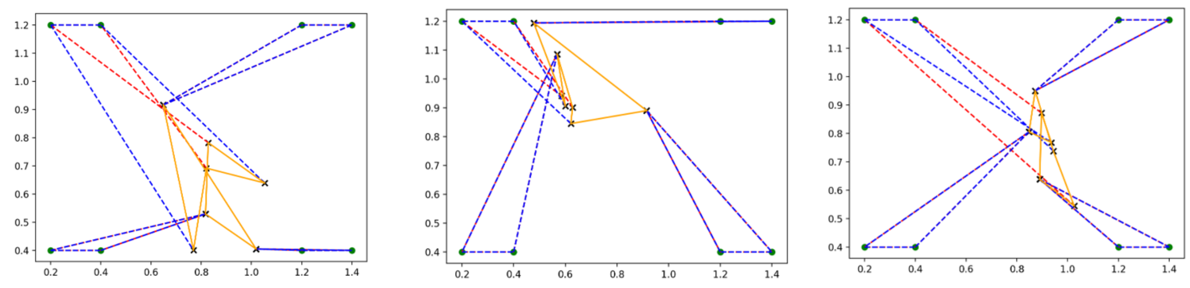
number of *minPts*	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
ε = 100	6	6	6	6	6	6	6	6
ε = 200	5	5	5	5	5	5	5	5
ε = 300	5	5	5	5	5	5	5	5
ε = 400	4	4	4	4	4	4	4	4
ε = 800	3	3	3	4	4	4	4	4

Table 4.2: The impact of *minPts* on the number of partitions in different values of ε in architecture 8Hub×8MC.

number of *minPts*	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8-16**
ε = 100	10	10	10	10	12	12	12	12
ε = 200	6	6	6	6	8	8	8	8
ε = 300	6	6	6	6	7	7	7	7
ε = 400	4	4	4	6	6	6	6	6
ε = 800	2	2	2	2	2	3	3	3

Table 4.3: Mean and variance of the number of clusters on varying ε in 2 architectures.

	4Hub×4MC		**8Hub×8MC**	
	number of partitioned clusters		**number of partitioned clusters**	
	Mean	Variance	Mean	Variance
ε=100	6	0	11.5	0.75
ε=200	5	0	7.5	0.75
ε=300	5	0	6.75	0.1875
ε=400	4	0	5.625	0.609375
ε=800	3.625	0.234	2.6875	0.2148



(a) 20 iterations (b) 200 iterations (c) 2000 iterations

Figure 4.1: optimization results.

4.2 Intra-cluster routing and Inter-cluster routing result

I run my optimization model on 4Hub×4MC architecture. I set the value of radius ε and minimum hubs *minPts* as 700 and 3 respectively. Since the random points are not included in the model, I need to rerun the model each time on applying different random points locations. Figure 4.1 shows the routing result when I run 20,200,2000 times respectively. However, the result does not converge until 2000 times. Besides, there are some evident issues that need to be addressed:

- Some random points are overlapping in my routing result. Additional constraints need to be added to prevent it.
- I do not have noise points in this optical layer layout. The existence of noise points for other layouts awaits further exploration.
- Random points are not included in my optimization model which may lead to extra redundancy of the model that needs to run a large amount of time to converge. Future optimization may be achieved by the integration of random points into the model. However, including random points induces non-linearity of the constraints. For example, the distance function calculation follows the non-linear square calculation which is prohibited by the ILP. Due to the time limitation, I could not carry out this improvement.
- Instead of one cycle, inappropriate constraints may lead to two independent cycles, which refers to two mutually exclusive cycles. [7] proposes a method by flipping the boundary edges on the grid graph to merge two cycles. However, this approach is only valid in grid graph.
- My optimization model on 4Hub×4MC architecture is rudimentary for this Hierarchical ring which reveals the major problems in my design. The extension to 8Hub×8MC architecture will be conducted once the aforementioned problems being addressed.

5 Conclusions

In this report, I propose a Hierarchical ring that improves the conventional ring by considering communication requirements and physical locations between nodes. I establish my topology in three steps: graph partitioning, cluster routing and global routing. Hierarchical ring has great potential. Further study of this topology could be conducted in 3 aspects. First, set constraints to prevent overlapping of the random points and rebuild the model considering random points to remove the extra redundancy to the model. Second, perform wavelength assignment and communication parallelism based on [8] and [9]. Last, compare optimization results in three scenarios: 1) Hierarchical ring structure, 2) Hierarchical ring structure with communication-parallelism, and 3) the state of the art with Hierarchical ring structure. Comparison criteria includes the performance and the network complexity, which could be accounted by the waveguide crossings, the MRR usage and optimization time.

Bibliography

[1] S. Beux, I. O'Connor, G. Nicolescu, G. Bois, and P. Paulin. "Reduction methods for adapting optical network on chip topologies to 3D architectures". In: *Microprocessors and Microsystems* 37 (Feb. 2013), pp. 87–98. DOI: `10.1016/j.micpro.2012.11.001`.

[2] L. Ramini, P. Grani, S. Bartolini, and D. Bertozzi. "Contrasting wavelength-routed optical NoC topologies for power-efficient 3d-stacked multicore processors using physical-layer analysis". In: *2013 Design, Automation Test in Europe Conference Exhibition (DATE)*. 2013, pp. 1589–1594. DOI: `10.7873/DATE.2013.323`.

[3] S. Le Beux, J. Trajkovic, I. O'Connor, and G. Nicolescu. "Layout guidelines for 3D architectures including Optical Ring Network-on-Chip (ORNoC)". In: *2011 IEEE/IFIP 19th International Conference on VLSI and System-on-Chip*. 2011, pp. 242–247. DOI: `10.1109/VLSISoC.2011.6081645`.

[4] S. Le Beux, H. Li, G. Nicolescu, J. Trajkovic, and I. O'Connor. "Optical crossbars on chip, a comparative study based on worst-case losses". In: *Concurrency and Computation: Practice and Experience* (Oct. 2014), pp. 2492–2503. DOI: `10.1002/cpe.3336`. URL: `https://hal.inria.fr/hal-01117004`.

[5] L. R. Ford and D. R. Fulkerson. "Maximal Flow Through a Network". In: *Canadian Journal of Mathematics* 8 (1956), pp. 399–404. DOI: `10.4153/CJM-1956-045-5`.

[6] B. Mike. *Traveling Salesman Problem*. 2011. URL: `https://gurobi.github.io/modelingexamples/traveling_salesman/tsp.html`.

[7] C. Umans and W. Lenhart. "Hamiltonian cycles in solid grid graphs". In: *Proceedings 38th Annual Symposium on Foundations of Computer Science*. 1997, pp. 496–505. DOI: `10.1109/SFCS.1997.646138`.

[8] M. Li, T.-M. Tseng, D. Bertozzi, M. Tala, and U. Schlichtmann. "CustomTopo: A Topology Generation Method for Application-Specific Wavelength-Routed Optical NoCs". In: *2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. 2018, pp. 1–8. DOI: `10.1145/3240765.3240789`.

[9] M. Li, T.-M. Tseng, M. Tala, and U. Schlichtmann. "Maximizing the Communication Parallelism for Wavelength-Routed Optical Networks-On-Chips". In: *2020 25th Asia and South Pacific Design Automation Conference (ASP-DAC)*. 2020, pp. 109–114. DOI: `10.1109/ASP-DAC47756.2020.9045163`.