Routing and placing filters for programmable photonics

Ferre Vanden Kerchove
Department of Information Technology
Ghent University - imec
Ghent, Belgium
Ferre.VandenKerchove@ugent.be

Didier Colle
Department of Information Technology
Ghent University - imec
Ghent, Belgium
Didier.Colle@ugent.be

Wouter Tavernier
Department of Information Technology
Ghent University - imec
Ghent, Belgium
Wouter.Tavernier@ugent.be

Wim Bogaerts
Department of Information Technology
Ghent University - imec
Ghent, Belgium
Wim.Bogaerts@ugent.be

Mario Pickavet
Department of Information Technology
Ghent University - imec
Ghent, Belgium
Mario.Pickavet@ugent.be

***Abstract*—Programmable photonic integrated circuits have the potential to increase the speed at which photonic applications are developed. However, in order to use these circuits effectively, there is a lack of efficient algorithms that compute an appropriate configuration for a given use case. We discuss a place-and-route algorithm that quickly calculates a solution to this problem. A specialized multi-stage simulated annealing process is introduced that gives a fast and effective way to configure the port-to-port response of programmable photonic integrated circuits.**

I. Introduction

Programmable photonic integrated circuits (PPICs) have the potential to substantially speed up and cheapen the development of photonic applications [1]. PPICs have the potential to serve a similar role as field-programmable gate arrays. These revolutionized the way electronic circuits were used. An essential part of this ecosystem consists of the algorithms that drive the various components that enable the programmable control of the chip's functionality. However, the algorithmic challenges that come with PPICs differ greatly from their electronic counterparts. This is due to the wavelength-dependent interference and resonance pattern that arises when light is folded back onto itself. This can be used to realize on-chip optical functionality, such as wavelength filtering and phase manipulation.

Current algorithms that study PPICs either only take routing into account as [2], [3], mainly focus on realizing a single filter with considerably large running times [4], [5], or manually place a predetermined filter in the mesh [6]. In order to fully utilize the potential of programmable photonics, multiple filters have to be realized on the same chip. The extension of the aforementioned single-filter algorithms to this requirement appears to be non-trivial. We argue that two barriers hamper the extension of these methods to realize multiple filters in the same mesh. Firstly, modeling the entire chip is computationally expensive, as evidenced by the large running times in [4], [5]. This raises potential scalability issues. Secondly, as the chip grows and the required functionality increases, manually placing filters is not scalable either [6].

In this paper, we tackle these issues through a multi-stage simulated annealing approach that splits the mesh into separate parts that each realize a single filter response. Before we detail this, we specify the problem in Section II. Then, filter synthesis

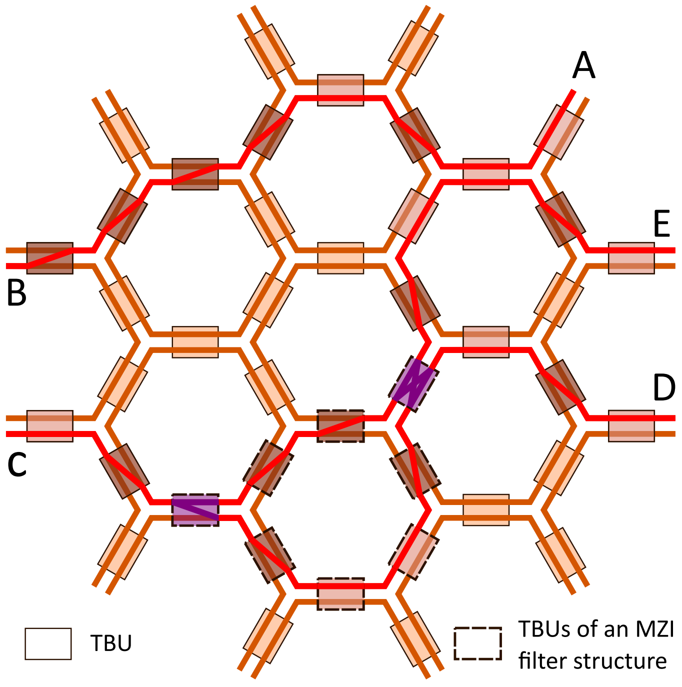

Fig. 1. A small mesh with a hexagonal architecture. A path is realized between A-B and an MZI between C and D-E.

This work was supported in part by the Flemish Research Foundation (FWO-Vlaanderen) under Grants G020421 (GRAPHSPAY) and 11O0923N, and in part by the European Research Council (ERC) under Grant 725555 (PhotonicSWARM).

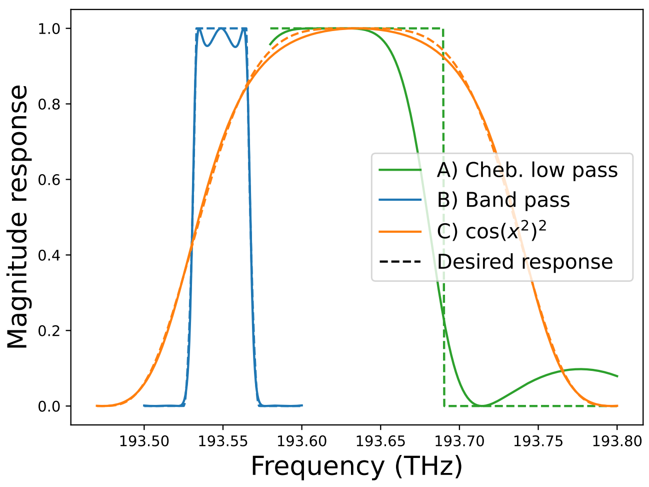

Fig. 2. Three desired filter responses and their synthesized filter response. A specific filter structure was chosen for each of these, which now needs to be placed in the mesh. For the low pass, a Chebyshev type II filter was chosen to approximate the desired response. This could have easily been a Chebyshev type I filter, an Elliptical filter, or a Butterworth filter.

is briefly addressed in Section III. The separate parts are placed through multiple stages of simulated annealing, each focusing on different objectives. This is discussed in detail in Section IV. Afterwards, the routing algorithm of [2] is used to calculate the necessary paths. To the best of our knowledge, this paper is the first that realizes more than 20 light processing functions in a single mesh, which we showcase in Section V.

II. Problem Statement

A PPIC consists of a mesh of waveguides connected through Tunable Basic Units (TBUs) arranged in a regular pattern. Each TBU can be configured to be in either bar mode, cross mode, or have a specific coupling factor. *Ports* are the locations where light can be coupled into and out of the mesh. In Fig. 1, the mesh is configured such that light is routed from port A to port B, and a Mach-Zehnder interferometer (MZI) is implemented from port C to ports D and E. This MZI is one example of a light processing function that can be realized on the chip. The optical functionality considered in this paper encompasses routing, either single port to single port, or single port to multiple, and wavelength filtering, the latter is also just *filtering*. The techniques in this paper can be easily extended to phase manipulation as well.

The problem studied in this paper is the following: Given a set of routing demands and light processing functions with corresponding input and output ports, find an appropriate configuration of the chip that realizes the required functionality. In the next section, we briefly discuss how the coupling coefficients are calculated, i.e., a process known as filter synthesis.

III. Filter Synthesis and Filter Structures

This section is meant as a small outline of how filter synthesis is handled by our algorithm. In (Vanden Kerchove, in preparation), we detail this in a much greater fashion.

Filter synthesis is the computation of the *filter structure* and the coupling coefficients of the TBUs in coupling mode. We go more in-depth about the filter structure at the end of this section. To synthesize a filter, we need to be able to calculate a filter response in the first place, given a configuration of TBUs. We simplify and extend the method discussed in [7], [8] in order to obtain an analytical, fully differentiable filter response $O(\omega, \mathbf{p})$ where $\mathbf{p}$ represents the coupling coefficients of the relevant TBUs. A desired filter response is defined by the complex transfer function $U(\omega)$ on the range $[\omega_{\text{min}}, \omega_{\text{max}}]$. We take a grid of N_{grid} equidistant frequencies $\omega_1, \ldots, \omega_{N_{\text{grid}}}$ at which the filter is evaluated. We define the error for a single desired filter response:

$$\text{Error}(\mathbf{p}) := \sum_{k=1}^{N_{\text{grid}}} |O(\omega_k, \mathbf{p}) - U(\omega_k)|^2. \tag{1}$$

Suppose that we have a fixed configuration of TBUs. We then treat this problem as a multivariate optimization problem in $\mathbf{p}$. The access to an analytical representation of the derivative enables us to employ derivative optimization techniques that converge faster than derivative-free methods in general. We use L-BFGS-B [9] built into Scipy, a gradient descent-based optimization technique. This analytical expression, combined with the derivatives, allows us to calculate good coupling coefficients that approximate the desired filter response in a matter of seconds.

However, how well we can approximate the desired filter response is inherently limited by the *filter structure*. This encompasses how the different light paths are laid out, split, and looped back, e.g., a ring resonator, two cascaded MZIs, a double ring loaded MZI [10], etc. For example, in Fig. 1, the filter structure of an MZI is indicated.

In order to find a good filter structure, our algorithm tries a variety of structures and cascades them as well, until an appropriate one is found. In Fig. 2, three different desired filter responses are shown, and the response that can be realized by relatively simple filters placed in the mesh.

IV. Place and Route Through Simulated Annealing

Once a filter structure is computed for every desired filter response, we need to find a good location for it in the mesh. Finding this, on its own, is already a difficult problem. Besides that, the following paths are necessary for every filter: paths from the input ports to the start of the filter structure and paths from the end of the filter structure to their appropriate output ports. We call these the wiring paths. Just calculating these paths is difficult as well, which is why we split up this place-and-route problem into three separate stages with separate goals, becoming gradually more computationally expensive. If at any stage, no solution is deemed possible, then the entire problem is deemed infeasible.

Firstly, the filters need to be placed such that no two filters have a conflict. Two filters have a conflict when both assign the same TBU a different mode or coupling coefficient. A simulated annealing algorithm is employed in order to solve this. Interestingly, placement and wiring (what we call routing) in electronics were one of the original usages that were given as example problems in one of the introductory papers of simulated annealing [11].

Algorithm 1 Simulated Annealing

```
1: **Input:** Initial starting point $s$, initial temperature $T$, cooling schedule $\alpha$, score function $f$
2: **Output:** Approximate global optimum solution $s^*$
3: $s^* \leftarrow s$
4: **while** $s$ has conflicts **do**
5:     find neighbor $s'$ of $s$
6:     $\Delta \leftarrow f(s') - f(s)$
7:     **if** $\Delta < 0$ **then**
8:         $s \leftarrow s'$
9:     **else**
10:        Accept $s'$ with probability $e^{-\Delta/T}$
11:    **end if**
12:    **if** $f(s) < f(s^*)$ **then**
13:        $s^* \leftarrow s$
14:    **end if**
15:    Update temperature: $T \leftarrow \alpha \cdot T$
16: **end while**
17: **return** $s^*$
```

Simulated annealing is mainly characterized by its starting point s, the neighborhood function, here called on Line 5, and the score function f. For a starting state, we calculate for every filter structure a place that minimizes the total distance needed for the wiring paths. The neighborhood function chooses a filter that has a conflict and moves that filter in one of the following ways.

1) Shift the filter in a random direction over a small distance.
2) Mirror the filter over its starting TBU.
3) Rotate the filter around the center point of the mesh.

The first two are local moves that are chosen with high probability, especially the first one with 95% probability. The last one is meant to escape local minima and is only chosen as 2% of the moves. This move set is chosen such that every filter could end up everywhere and in a limited number of moves. The score function is the sum of the number of TBUs in conflict of every filter.

The second stage is a check for the feasibility of routing the wiring paths, that is, the paths that are needed to guide the signals from the input ports to the filter start, and from the filter end to the output ports. As discussed before, routing these is hard. Thus, we first check if the wiring paths of every filter can be realized when the other filter structures are in place, but before their wiring paths are routed. This is in contrast to stage three, where all wiring paths are routed simultaneously. We rerun the simulated annealing algorithm with the previous ending point as the starting point and the same neighborhood function. A filter has conflicts either when it has a conflicting usage of a TBU with another filter, or its wiring paths cannot be realized without conflicts. Similarly, the scoring function contains an additional penalty for every filter that cannot have its wiring paths routed. It is computationally expensive to conduct this wiring path check, which is the reason for the split of this stage and the first.

In the third stage, another round of simulated annealing is conducted. If multiple wiring paths have conflicting uses for TBUs, then this is penalized in the scoring function. The starting state is the ending state of the previous stage, and the neighborhood function stays the same. This is also the stage at which normal routing and splitting are handled. We use the algorithm of [2] and take only routing into account. Splitting can be handled by the algorithm in [12] instead.

This multi-stage simulated annealing algorithm is able to find a working configuration for the entire chip in many cases, as we showcase in the next section.

V. Results

The desired filter responses are largely high and low pass filters, but they also include some arbitrary functions such as $\cos(x^2)^2$ and modified bandpass filters that have a specified roll-off.

The filters are always 1x2, meaning they have one input port and two output ports, but our method can be generalized to mxn filters. The desired filter response is routed to the first output port, and the complement is routed to the second output port.

There is currently a lack of test sets that represent realistic usage of PPIC. The lack of similar algorithms means that there is no baseline to compare against. This is why we have two types of test sets.

In the first, we construct (semi-)manually a problem instance. For this, we place and route filter structures sequentially until we can no longer add more. This is then added as a problem to the test set. For every problem in this test set, we know for sure that it has a solution. The second test set is randomly generated. Input and output ports are chosen uniformly from the set of all ports.

The algorithm performs well on the sequentially generated test set. The test set contains 100 problem instances with varying mesh sizes, going from radius $r = 5$ to $r = 15$ and containing up to 14 different filters. The algorithm finds a solution to 92% of the sequentially generated problems. These numbers are not too surprising, as the sequentially generated test sets are not able to pack a lot of filters.

Next, we look at the performance of randomly constructed problem instances. As we do not know which ones have a solution, we show the results for the problems where we find one. The total time is recorded for simulated annealing, excluding the time to synthesize, in order to avoid too much clutter. The average time to synthesize a single filter structure is about 1 second.

In Fig. 3, the results are shown for varying mesh sizes and number of problems. For smaller meshes, the average time needed for simulated annealing is in the order of seconds. If the meshes become large and the number of problems stays the

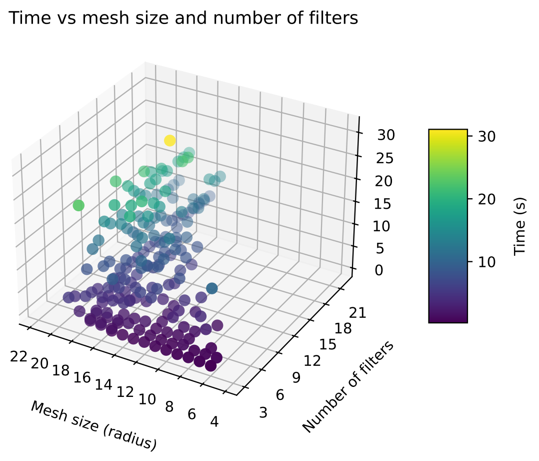

Fig. 3. Time needed to solve problems as a function of the radius of the mesh and the number of problems.

same, then the time decreases again, as the problem becomes easier to solve. More space for the same number of filters gives more flexibility and more potential solutions. However, as the number of filters increases, the runtime steadily rises.

VI. Conclusion and future work

In this paper, we propose an effective approach that provides a working configuration for a programmable photonic integrated circuit, even when multiple filters are required. The simulated annealing algorithm is split up into three stages in order to minimize the computational cost and detect infeasibility early. The efficacy of the algorithm is showcased through two test sets. One is constructed to contain problems for which we have a working configuration. The fact that the algorithm is often able to find a solution again shows the potential of our approach. The second test set contains problems that are randomly constructed, giving more insight into the timing performance.

One of the key challenges of our simulated annealing approach is the dependency on the regularity of the mesh. Otherwise, we cannot easily move the filter structure around with our current neighborhood function. This is a noticeable drawback. Irregular architectures can provide greater flexibility in terms of free spectral range and realize certain structures more compactly. Extending this algorithm to these irregular architectures does not seem trivial. Future work could consider how to solve this filtering problem on irregular architectures, and then use this as a means of evaluating different architectures based on various factors such as versatility, the number of filters it can handle, routing capacity, etc.

References

[1] W. Bogaerts and L. Chrostowski, “Silicon Photonics Circuit Design: Methods, Tools and Challenges,” *Laser & Photonics Reviews*, vol. 12, no. 4, p. 1700237, Apr. 2018.

[2] F. Vanden Kerchove, X. Chen, D. Colle, W. Tavernier, W. Bogaerts, and M. Pickavet, “An Automated Router With Optical Resource Adaptation,” *Journal of Lightwave Technology*, vol. 41, no. 18, pp. 5807–5819, Sep. 2023.

[3] A. López, D. Pérez, P. DasMahapatra, and J. Capmany, “Auto-routing algorithm for field-programmable photonic gate arrays,” *Optics Express*, vol. 28, no. 1, p. 737, Jan. 2020.

[4] Z. Gao, X. Chen, Z. Zhang, U. Chakraborty, W. Bogaerts, and D. S. Boning, “Automatic synthesis of light-processing functions for programmable photonics: Theory and realization,” *Photonics Research*, vol. 11, no. 4, p. 643, Apr. 2023.

[5] ——, “Gradient-Based Power Efficient Functional Synthesis for Programmable Photonic Circuits,” *Journal of Lightwave Technology*, pp. 1–10, 2024.

[6] D. Pérez, A. López, P. DasMahapatra, and J. Capmany, “Multipurpose self-configuration of programmable photonic circuits,” *Nature Communications*, vol. 11, no. 1, p. 6359, Dec. 2020.

[7] D. Pérez and J. Capmany, “Scalable analysis for arbitrary photonic integrated waveguide meshes,” *Optica*, vol. 6, no. 1, p. 19, Jan. 2019.

[8] E. Sanchez, A. Lopez, and D. Perez-Lopez, “Simulation of Highly Coupled Programmable Photonic Circuits,” *Journal of Lightwave Technology*, pp. 1–12, 2022.

[9] R. Byrd, L. Peihuang, and J. Nocedal, “A limited-memory algorithm for bound-constrained optimization,” *SIAM Journal on Scientific Computing*, 1996.

[10] M. Wang, C. Xiangfeng, U. Khan, and W. Bogaerts, “Programmable wavelength filter with double ring loaded MZI,” *Scientific Reports*, 2022.

[11] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, “Optimization by simulated annealing,” *Science*, vol. 220, no. 4598, p. 671–680, May 1983.

[12] X. Wang, F. Vanden Kerchove, R. Raikar, M. Pickavet, W. Bogaerts, and D. Stroobandt, “A Novel Connection-based Multicasting Router for Programmable Photonic Circuits,” *Journal of Lightwave Technology*, pp. 1–11, 2024.