Optics EXPRESS

Auto-routing algorithm for field-programmable photonic gate arrays

AITOR LÓPEZ,[1,*] DANIEL PÉREZ,[1,2] PROMETHEUS DASMAHAPATRA,[1,2] AND JOSÉ CAPMANY[1,2]

[1] *ITEAM Research Institute, Universitat Politècnica de València, Camino de Vera s/n 46022 Valencia, Spain*
[2] *iPronics Programmable Photonics S.L., Camino de Vera s/n 46022 Valencia, Spain*
[*] *ailoher@iteam.upv.es*

Abstract: Programmable multipurpose photonic integrated circuits require software routines to make use of their flexible operation as desired. In this work, we propose and demonstrate the use of a modified tree-search algorithm to automatically determine the optimum optical path in a field-programmable photonic gate array (FPPGA), based on end-user specifications, circuit architecture and imperfections in the realized FPPGA arising, for example, from fabrication variations. In such a scenario, the proposed algorithm only requires the hardware topology and the location of the connections of the FPPGA defining the optical path to be programmed. The routine is able to optimize the path over multiple and competing objectives like the overall length, accumulated loss and power consumption. In addition, should any region of the circuit suffer from any potential damage that may affect the device performance, this algorithm is also able to provide basic self-healing and fault-tolerance capabilities by supplying alternative paths through the photonic arrangement.

© 2020 Optical Society of America under the terms of the OSA Open Access Publishing Agreement

1. Introduction

Photonic integration combines multiple optical components on a chip to enable optical signal processing while maintaining a low-form factor. It has mostly been utilized in the form of Application Specific Photonic Integrated Circuits (ASPICs), where each circuit is designed and optimized to perform a particular functionality. As with electronic integrated circuits, the non-recurring engineering costs including custom mask tooling, design hours, custom packaging and specific process developments reduce the cost effectiveness for moderate volume applications. Although Photonic Integrated Circuits (PICs) have been proven successful for a myriad of applications in the literature, only a few like transceivers and data centres have shown enough volume fabrication to compensate the cost overhead [1]. Similar to electronics, a solution leading to mass production and subsequent cost reduction for PIC manufacturing, multi-project wafers are fabrication runs where different designs from different users are combined on the same wafer providing cost sharing [2–4]. However, their time-to-market becomes limited by the design processes and by large development periods to minimum of 12-24 months per design-fab-packaging-test iteration, depending on the chip complexity [5].

In the quest to enable cost-effective and programmable photonic-driven solutions, field programmable photonic arrays (FPPGAs) have recently emerged as a new solution to achieve general-purpose functionality and flexible operation [6]. The design of such devices is based on a generic PIC hardware comprising a set of reconfigurable processing blocks and a core of photonic actuators and beamsplitters that are programmed by the user, enabling their use across a wide variety of optical signal processing functionalities [7,8]. The FPPGA core can be employed to program optical components such as optical splitters, combiners, couplers, routers, delay lines, optical filters, beamformer networks and multiport interferometers [7–9], as well as to route on-demand using High-performance Building Blocks (HPBBs). HPBBs are components that are outside of the FPPGA mesh core providing certain specific functionality to the circuit as a whole.

#382753 https://doi.org/10.1364/OE.382753
Journal © 2020 Received 8 Nov 2019; revised 14 Dec 2019; accepted 14 Dec 2019; published 3 Jan 2020

Such functionality can be exhibited in the form of high quality filters by using ring resonators or as EO converters using Mach-Zehnder modulators, to cite a few.

Very recently, novel waveguide mesh topologies have been reported to allow higher integration densities [10]. However, their scalability is currently constrained by several factors, some arising from physical hardware constraints like their accumulated insertion loss, power consumption, footprint, optical crosstalk, thermal-tuning crosstalk and electrical interfacing during the packaging stage and the others dealing with their precise control and configuration. This has led to current experimental demonstrations being limited to a moderate number (few tens) of unit cells owing greatly to the manual or semi-automated operation calibration employed, resulting in a time-consuming and very limited option when such arrangements expand to larger scales.

Recent demonstrations explore the use of advanced optimization methods for the self-configuration of the circuits, requiring several iterations until convergence into the desired functionality [10]. However, being a key building block for most of the optical signal processing functionalities, the implementation of a specific algorithm targeting the auto-routing between HPBBs and the synthesis of optical delay lines has not been addressed, to date. Importantly, its application could improve the efficiency of the optimization and programming process and expand the capabilities to a wide range of applications. The configuration of optical connections and optical delay lines in general-purpose waveguide meshes requires the selection and configuration of the programmable unit cells in the arrangement that will define the path followed by the optical signal. For example, consider the programmed FPPGA in Fig. 1. For a given specification, (a delay of 10 ns between two specific optical ports, or connection between optical sources and electro-optical modulators defined by nodes A and B.), there will be multiple solutions. In such a scenario, the implementation of a software algorithm capable of finding and automatically controlling the optimum path in term of loss, power consumption and other non-ideal effects is highly desired, [11].

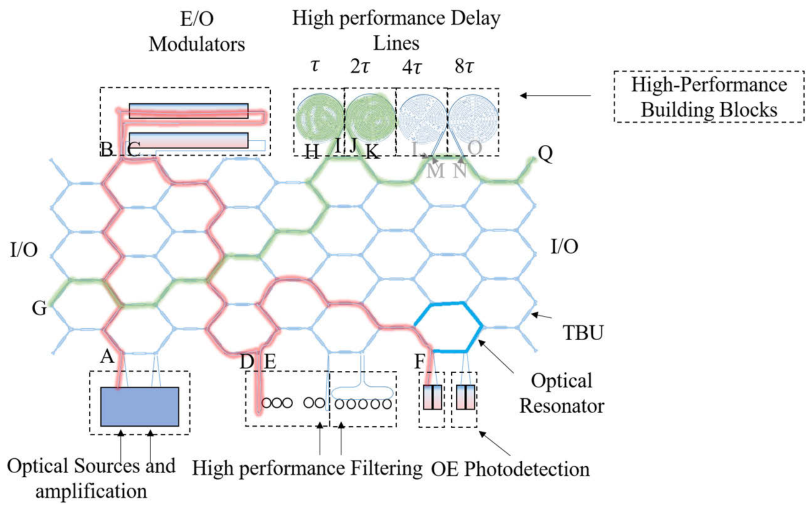


Fig. 1. The optical FPPGA [6] with two circuits configured simultaneously. Circuit 1 is defined by the interconnection of labelled nodes A-B, C-D, E-F. Circuit 2 is defined by the interconnection of labelled nodes G-H, I-J, K-Q.E/O: Electro-optical, I/O: optical inputs outputs, TBU: Tunable Basic Unit.

Following with the example, Fig. 1 illustrates the configuration of two circuits working in parallel. The first one (in red) requires the connection between points AB, CD and EF. The second one requires the connection between GH, IJ and KQ. In addition, the first circuit incorporates the formation of an optical resonator (in blue). This scenario requires an automated routine to build up the interconnection and the programmable waveguides for the structures programmed in the FPPGA core. To address this problem, we explore the use of auto-routing algorithms to find the optimum configuration of a waveguide mesh arrangement for different features. The reported solution, following our previous work done in [12], is inspired by the modification of Dijkstra's algorithm [13], widely used in many different research fields such as IP traffic engineering [14], artificial intelligence [15] and even for classical FPGA routing [16–18]. In addition, we demonstrate that the use of the algorithm leads to unprecedented self-healing and fault-tolerant capabilities of the PIC, where given a set of damaged areas of the circuit, the algorithm is able to find alternative sub-optimal paths through the photonic arrangement. Before concluding the paper, we discuss about the application scenarios and constrains of the proposed solution.

2. Definitions of the photonic circuit and the method

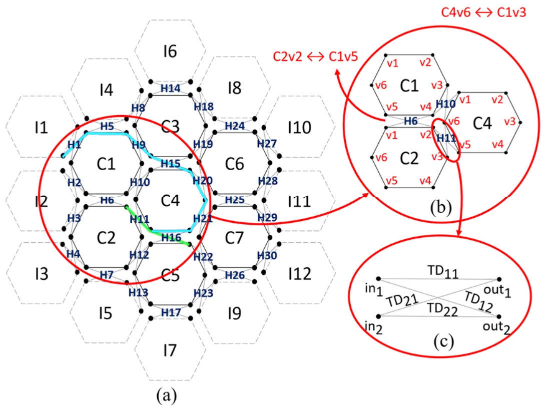
Figure 1 illustrates a hexagonal FPPGA core. This PIC performs two main tasks: First, it provides the dynamic connections between high-performance photonic building blocks connected to each optical port and the optical I/Os at the optical interface. Secondly, it enables the synthesis of optical delay lines, beamsplitters and combiners, and phase actuators by means of software programming each programmable unit cell or Tunable Basic Unit (TBU). When combined, these building blocks are employed to build-up more complex optical processing circuits. With the aim to enable the automatic reconfiguration and programming of each TBU to configure optical connections and delay lines, we apply a control routine inspired by *shortest-path* evaluation techniques. Before discussing the algorithm in more depth, we define first the key concepts in graph theory and their adaptation to waveguide mesh-based photonic integrated circuits.

- *Graphs*, the fundamental objects in graph theory, are systems of *nodes* connected in pairs by *edges*. In this work, the *nodes* are the physical optical ports of the TBUs and the *edges* represent the connections between the TBU ports.
- *Weights* are numerical values assigned to each graph *edge*. The overall weight of any *path* inside the graph (i.e., the route traversed with or without repeated nodes and consequently edges) will be given by the sum of the weights of the edges within such path. In this work, the weights are defined as the performance parameter to be optimized during the creation of the optical connection or delay line.

The fundaments of *shortest-path* evaluations consist of searching the shortest route between two nodes through a weighted graph with the purpose of finding the route that accumulates the least weight [19]. In addition, the proposed method should avoid a brute-force search of every possibility to ensure scalability. The method proposed in this work features a couple of differences with respect to the basic Dijkstra's algorithm, the most common example in this family of algorithms.

Figures 2(a)–2(c) illustrate the graphical representation of a 7-cell hexagonal mesh containing 30 TBUs. In this case, a balanced Mach-Zehnder interferometer (MZI) is used to form the TBU [7]. Each TBU has a phase actuator attached to each of its arms, allowing the independent tuning of both its power splitting ratio and phase, as shown in Fig. 2(d). Hence, by modifying each TBU configuration we can alter the full scattering matrix of the waveguide mesh arrangement, and therefore, allow the synthesis of a wide range of photonic structures. Each of its isolated optical nodes corresponds to an actual different input/output optical connection between TBUs.

In order to develop, apply and illustrate the performance of this auto-routing algorithm, we need to define and name our graph nodes (optical nodes), and edges (optical connections using TBUs). To do so, we name each cell and create an additional set of imaginary cells at the circuit perimeter, as illustrated in Fig. 2(a) by the dashed cells I_1-I_{12} (where 'I' stands for 'imaginary'). In addition, we name each inner vertex at each cell as v_{1-6}, and each of the 30 TBU as H_{XX}.



# TBU	State	in	out	IL (dB)	P_C (mA)	BUD (ps)	BUL (µm)
H11	Bar	C_2v_3	C_2v_2	0.53	185	13.5	975
		C_2v_2	C_2v_3	0.53			
		C_4v_5	C_4v_6	0.53			
		C_4v_6	C_4v_5	0.53			
	Cross	C_2v_3	C_4v_6	0.53	105		
		C_4v_6	C_2v_3	0.53			
		C_2v_2	C_4v_5	0.53			
		C_4v_5	C_2v_2	0.53			

(d)

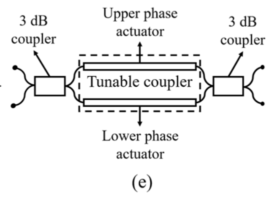


Fig. 2. (a) Graph representation of a 30 TBU waveguide mesh. TBUs, actual and imaginary cells are numbered from top to bottom and from left to right. Graph nodes within each hexagonal cell are numbered clockwise and hereinafter referred to as $\mathrm{C(/I)_x v_y}$, where $\mathrm{C_x}$ or $\mathrm{I_x}$ represents the actual ($x = \{1, 2, \ldots, 7\}$) or imaginary cell ($x = \{1, 2, \ldots, 12\}$) in which the node is located. Two light path examples are marked in blue and green respectively, (b) node representation where $\mathrm{v_y}$ denotes its position inside of it ($y = \{1, 2, \ldots, 6\}$), (c) internal connections of the TBU where TD: Transmission distance stands for the weight or cost to travel from one node to another. (d) Illustration of the eight possible TDs of TBU H11 along with several experimental figures of merit under use in this work as input for the algorithm, where IL: Insertion loss, $\mathrm{P_c}$: Power consumption, BUD: Basic Unit Delay, BUL: Basic Unit Length. (e) Schematic example of MZI implementing a TBU, whose phase imbalance is created by an independent, thermal tuning of each of its arms.

Finally, we name the *edges* as TD_{io}, where *i,o* represent the input and the output port number, respectively.

With these definitions, the representation of the graph of a single TBU can also be observed in Fig. 2(c). Next, in order to apply the algorithm, we can define a weight or transmission distance (TD) to each edge connecting two optical nodes in the arrangement. This TD is a weighted sum of the TBU's main figures of merit (FoM), which can be its *insertion loss* (IL), the *power consumption* (P_c) and its *basic unit length* (BUL) the sum of the tunable coupler length and the arc length of the access waveguides, and the *basic unit delay* (BUD) [7]. Figure 2(d) summarizes several of these FoM for TBU H11. For normalization purposes, the weights can simply add up to 1 and reflect the importance given to each parameter during the optimization process. Hence, the proposed algorithm is able to find by itself the optimum path with respect to any combination of these attributes by only changing this distribution of weights $\{c_i\}$, as desired:

$$TD_{xy} = c_1 \cdot \mathrm{IL} + c_2 \cdot \mathrm{BUL} + c_3 \cdot P_c + \ldots \tag{1}$$

After defining such distance for every TBU, a shortest path with the input node as the root propagates through the remaining ones by accumulating each TD prior to reaching destination port, as in classical tree-search algorithm implementations. However, it is subject to a couple of additional constraints: First, light cannot propagate through the same TBU twice consecutively, as this is not physically possible without it being recirculated by an external element. Secondly, the paths are only discarded if they go through the same node two times or if they force any of its constituting TBUs to be (simultaneously) in two different transmission states during the synthesis of any specific interferometric circuit. Instead, the algorithm keeps on running until a fixed number of TBU paths, defined as an input argument, reaches the destination port. Then, the routine selects the optimum path based on the accumulated TD values as given by Eq. (1).

A proposed pseudocode to achieve this task can be found below. As a preliminary step, we index all the TDs from our proposed graphs. This aids in implementing the aforementioned constraint whereby a path cannot traverse through a TBU twice consecutively, thus leading to greatly expedited process. The algorithm starts by creating the graph framework from this ordered list of nodes and TDs and by setting the accumulated distance from initial node to itself as zero and to all the others as infinity. From then on, a shortest path tree with the input port as root propagates through the remaining ones in the graph by accumulating each TD prior to reaching the destination port. Similar to original Dijkstra's implementation, the paths that go through the same node more than one time during the process are discarded and the rest are stored inside the '*paths*' variable. Once the destination port has been reached, the resulting path is stored in 'pathsDest' variable and the process keeps running until a fixed number of paths defined in 'max_paths' variable also arrives to the destination. A potential issue arising from this implementation is that the number of paths stored in 'paths' variable would increase exponentially after each iteration, leading to equally large computational times while dealing with the synthesis of larger delay lines. A recommended alternative to overcome such a problem consists on implementing the same protocol for both source and destination ports of the path to be synthesized and store each path emerging from the intersection of both branches in 'pathsDest' (provided that the same set of aforementioned constraints are also met), hence preventing the path trees from both nodes to expand to greater extents.

To demonstrate the behaviour, we implemented the algorithm in Python and apply it to several application cases, as described in the next section. We run all the experiments using a desktop, 4-core, 3.60 GHz processor.

```
requires initialNode, destNode, listOfTDs
procedure findShortPath (initialNode, destNode, listOfTDs)
    % initialNode: Initial node of the synthesized path
    % destNode: Destination node of the synthesized path
    % listOfTDs: set of TDs of each TBU in the graph (load architecture)

    set currentNode as initialNode
    set paths, pathsDest equal to empty list
    add currentNode inside paths % We now have a first path containing 'currentNode'
    set max_paths

    setNodes() % Mark all graph nodes unvisited and store them
    setDistances() % Set the accumulated distance to zero for the initial node and to
            infinity for all the others
    while length(pathsDest) < max_paths do
      for path in paths
        currentNode = path[last] % We take the last element of 'path'
        TD(currentNode, prior TBU node), TD(currentNode, opposite TBU input node) = $\infty$
        % To  avoid backpropagation of light in next iteration
        for neighbour in neighbourList do % 'neighbourList' contains all nodes in
                                            the vicinity of 'currentNode'
          if TD(currentNode, neighbour) != $\infty$
            accumulate TDs from currentNode to neighbour,
            make a copy of path and add neighbour inside it
          if neighbour appears twice in path
            remove path from paths
          if neighbour == destNode
            add {path, accumulated ID} inside pathsDest
            break
```

3. Results

In this section, we will demonstrate the performance of the auto-routing algorithm using two hexagonal waveguide meshes of different sizes: a 7-cell and an 18-cell configuration, containing 30 and 81 TBUs, respectively. First, we will deal with the formation of optical delay lines and optical connections in such arrangements. Afterwards, we extend the application of our algorithm to the synthesis of optical interferometers.

3.1. Synthesis of optical delay lines

In our first experiment, we consider the optimum path with respect to the number of traversed TBUs; i.e., $c_1 = c_3 = 0$, $c_2 = 1$. In other words, we try to find the shortest path between two vertices in the graph (optical nodes). Under this assumption, we simulated a total of 5 different optimized optical paths (in both directions) in the waveguide mesh of 30 TBUs setting 'max_paths' to 1 to stop the process as soon as the first path is retrieved. Therefore, we are ruling out the possibility of finding other paths with the same length but optimized with respect to other FoM. The results can be found in Table 1, together with the average elapsed time after nine independent executions of the algorithm for each path synthesis, and in Fig. 3. In particular, note how Fig. 3(c) illustrates a clear example of a case where a shorter path between nodes C_1v_4 and C_2v_2 (or vice versa) could be traversed through C_2v_1, however, as discussed at the end of previous section, the forward propagation of light forces the synthesis of an optical loop consisting of 5 TBUs, as backwards propagation inside the TBU is not permitted both physically and by the algorithm.

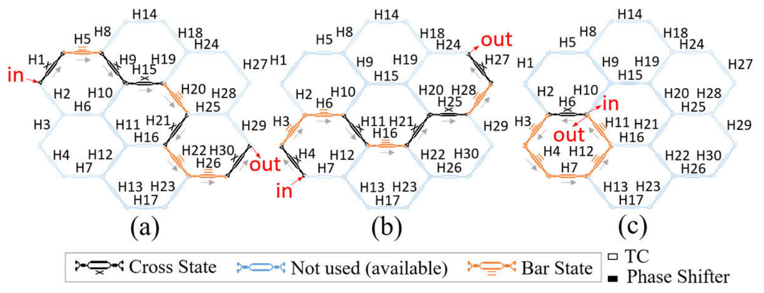


Fig. 3. Synthesis of optical delay lines #1 (a), #3 (b) and #5 (c) from Table 1 in a 30-TBU waveguide mesh.

Table 1. Results obtained from the synthesis of 5 different delay lines and interconnections optimized with respect to the number of crossed TBUs in a 30-TBU small waveguide mesh.

#	I/O	Obtained path	# TBUs	Average duration (s)
1	$I_{12}v_1 \rightarrow I_1v_4$	$[I_{12}v_1, C_7v_4, C_7v_5, C_7v_6, C_4v_3, C_4v_2, C_3v_5, C_1v_2, C_1v_1, I_1v_4]$	9	0.056
	$I_1v_4 \rightarrow I_{12}v_1$	$[I_1v_4, C_1v_1, C_1v_2, C_3v_5, C_4v_2, C_4v_3, C_7v_6, C_7v_5, C_7v_4, I_{12}v_1]$	9	0.054
2	$C_1v_1 \rightarrow C_4v_3$	$[C_1v_1, C_1v_2, C_3v_5, C_4v_2, C_4v_3]$	4	0.034
	$C_4v_3 \rightarrow C_1v_1$	$[C_4v_3, C_4v_2, C_3v_5, C_1v_2, C_1v_1]$	4	0.038
3	$I_{10}v_6 \rightarrow I_3v_3$	$[I_{10}v_6, C_6v_3, C_6v_4, C_7v_1, C_4v_4, C_4v_5, C_2v_2, C_2v_1, C_2v_6, I_3v_3]$	9	0.054
	$I_3v_3 \rightarrow I_{10}v_6$	$[I_3v_3, C_2v_6, C_2v_1, C_2v_2, C_4v_5, C_4v_4, C_7v_1, C_6v_4, C_6v_3, I_{10}v_6]$	9	0.051
4	$I_7v_2 \rightarrow I_8v_6$	$[I_7v_2, C_5v_5, C_5v_6, C_5v_1, C_4v_4, C_4v_3, C_6v_6, C_3v_3, I_8v_6]$	8	0.053
	$I_8v_6 \rightarrow I_7v_2$	$[I_8v_6, C_3v_3, C_6v_6, C_4v_3, C_4v_4, C_5v_1, C_5v_6, C_5v_5, I_7v_2]$	8	0.050
5	$C_1v_4 \rightarrow C_2v_2$	$[C_1v_4, C_2v_1, C_2v_6, C_2v_5, C_2v_4, C_2v_3, C_2v_2]$	6	0.049
	$C_2v_2 \rightarrow C_1v_4$	$[C_2v_2, C_2v_3, C_2v_4, C_2v_5, C_2v_6, C_2v_1, C_1v_4]$	6	0.048

Secondly, we changed the values of the coefficients to $c_1 = c_2 = 0$, $c_3 = 1$ to search for those synthesized paths that minimize power consumption. Before starting, we would need to know in advance the required driving power to set each individual TBU to cross/bar states. Due to fabrication induced phase errors, each TBU is in a random state under passive conditions implying that the electrical power required to set them to cross or bar is equally random. Such values have been obtained by means of a basic routine to characterize the response of every phase actuator and TBU [9]. If we run the algorithm, under this new criterion to the same optical ports as employed in Fig. 3 we obtain different paths, as illustrated in Fig. 4. Comparing the power required by the path in Fig. 3 and the path in Fig. 4, we can obtain power consumption improvements –assuming that all heaters have the same resistance– of 45.08%, 30.12% and 20.35%, respectively. This capability can be essential in large-scale waveguide mesh arrangements.

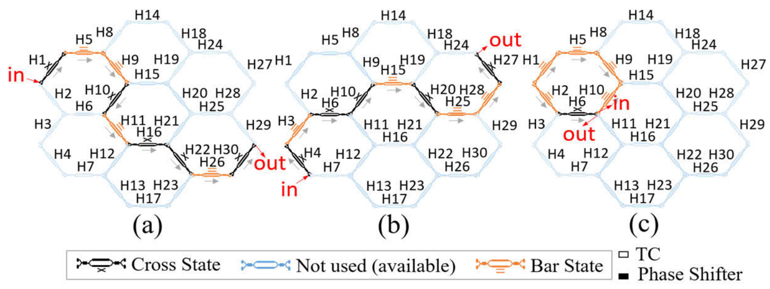


Fig. 4. Re-synthesis of optical circuits from Fig. 2 using our auto-routing algorithm optimizing with respect to power consumption.

Next, we study the optimization of the 30-TBU waveguide mesh with respect to the overall IL of the optical route, thus having $c_2 = c_3 = 0$, $c_1 = 1$. We define the IL of each individual TBU from a truncated Gaussian random variable distribution (to satisfy that its sign will be positive) with mean 0.59 and a standard deviation of 0.05 to account for any potential fabrication error as reported in [9]. We then repeated our simulation considering an average IL of 0.15 with the same standard deviation for each TBU as reported in [20].

Here we can illustrate one of the main virtues of this algorithm: its potential use to provide *self-healing* or *fault-tolerant* capabilities to any multipurpose waveguide mesh arrangement. To demonstrate this behavior, Fig. 5 shows a scenario where our algorithm is able to find an alternative path between nodes C_2v_1 and C_4v_2. Here, TBU H_{10} is simulated to represent a malfunction (which in reality can, for example, arise from an error during fabrication) which results in its losses to be -20 dB. After applying the algorithm, we obtain solutions featuring an overall IL of 2.82 dB and 1.01 dB for both IL distributions, respectively. Such values are indeed not so far away from the ones before applying extra H_{10} malfunction, which were 1.84 and 0.47 dB. The average elapsed time taken by the routine was in both cases in the order of 0.16 s. Owing to this feature, the functionality of programmable PIC based on waveguide meshes can be sustained owing to its tolerance for fabrication errors and regardless of any local impairments, very much unlike the case of an ASPIC.

This auto-routing algorithm also supports the synthesis of more than one optical path at the same time, leading to the creation of multi-in, multi-out systems such as the one shown in Fig. 1. To do so, we first focus on the synthesis of the first structure and, once finished, we set all the TDs corresponding to the opposite transmission states and directions for each of its constituting TBUs to infinity. Once finished, we can run the algorithm sequentially to create new structures while maintaining the previous ones, as illustrated in the example from Fig. 6(a) and Fig. 6(b). Looking at both results, we observe how these will depend on the order in which we synthesize each individual element. As a rule of thumb, it will be preferable to start from those optical paths

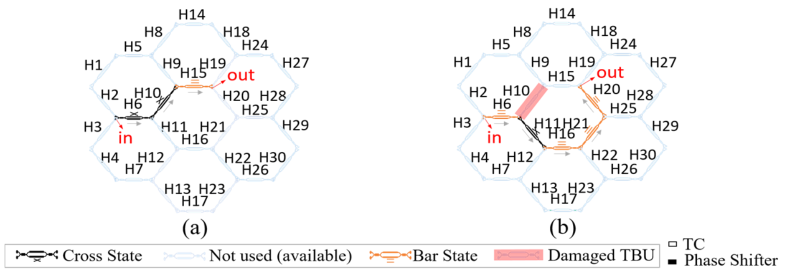


Fig. 5. Demonstration of self-healing capability in a 30-TBU waveguide mesh. After a malfunction in TBU H_{10} (a), a sub-optimum path can be reconfigured through H_{11}, H_{16} and H_{21} (b).

whose inputs and outputs lay closer, so that a smaller number of edges would become unusable for the rest of the paths waiting to be synthesized using the algorithm.

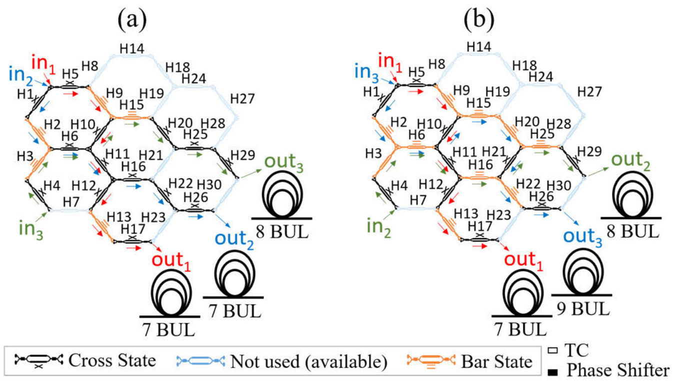


Fig. 6. Consecutive synthesis of multiple delay lines at a time in a 30-TBU waveguide mesh in two different orders. Inputs and outputs are numbered following the order in which their corresponding optical paths were synthesized.

In order to explore the behavior of the algorithm in a large-scale waveguide mesh arrangement we applied it to an alternative circuit. The graph representation of the new 18-cell hexagonal waveguide mesh under study can be found in Fig. 7. This time, apart from the actual cells, it also includes 20 imaginary cells that surround them as in the previous graph design.

Following again the procedure described in Section 2, we start by searching the optimum path with respect to the number of traversed cells to synthesize several optical connections, whose results can be observed in Table 2 and Fig. 8. This time, the average duration of ten independent experiments, scales up to a few seconds for larger delay lines. Again, we forced the process to stop when the first result was retrieved.

Moreover, we also demonstrate the self-healing and fault-tolerant capabilities of the algorithm using this larger mesh. This time we considered them to feature an average IL of 0.59 dB with a standard deviation of 0.05 and a larger number of damaged TBUs, seven, as can be observed in Fig. 9. On the extreme right and left hand corner of the image, we can observe the optimum path between nodes I_7v_6 and $I_{16}v_4$, featuring an overall IL of 8.36 dB. If we repeat such experiment considering a malfunction in TBUs H_{14}, H_{15}, H_{32}, H_{41}, H_{51}, H_{58} and H_{71} –with said malfunction increasing each of their IL to 20 dB– and perform the optimization routine with respect to the *overall loss*, the algorithm returns an alternative path, which can be observed on the right hand side of the same figure, with an accumulated IL of 13.34 dB. However, the elapsed time (around

Table 2. Results obtained from the synthesis of 5 different delay lines optimized with respect to the number of crossed TBUs in our large waveguide mesh.

#	I/O	Best Path	# TBUs	Average duration (s)
1	$I_1v_2 \rightarrow I_{20}v_2$	$[I_1v_2,I_6v_5,C_1v_2,I_8v_5,C_5v_2,C_5v_3,C_9v_6,C_9v_5,C_{10}v_2, C_{13}v_5,C_{14}v_2,C_{17}v_5,C_{18}v_2,I_{19}v_5,I_{20}v_2]$	14	0.703
	$I_{20}v_2 \rightarrow I_1v_2$	$[I_{20}v_2,I_{19}v_5,C_{18}v_2,C_{18}v_1,C_{14}v_4,C_{13}v_5,C_{10}v_2,C_9v_5, C_9v_6,C_5v_3,C_5v_2,I_8v_5,C_1v_2,I_6v_5,I_1v_2]$	14	0.690
2	$I_8v_6 \rightarrow I_{13}v_4$	$[I_8v_6,I_8v_5,C_5v_2,C_5v_3,C_9v_6,C_9v_5,C_{10}v_2,C_{10}v_3,C_{14}v_6,C_{14}v_5,I_{13}v_2,I_{13}v_3,I_{13}v_4]$	12	0.503
	$I_{13}v_4 \rightarrow I_8v_6$	$[I_{13}v_4,I_{13}v_3,I_{13}v_2,C_{14}v_5,C_{14}v_6,C_{10}v_3,C_{10}V_2,C_9v_5, C_9v_6,C_5v_3,C_5v_2,I_8v_5,I_8v_6]$	12	0.508
3	$C_3v_6 \rightarrow C_{16}v_2$	$[C_3v_6,C_3v_5,C_3v_4,C_7v_1,C_6v_4,C_{10}v_1,C_9v_4,C_{13}v_1, C_{12}v_4,C_{16}v_1,C_{16}v_2]$	10	0.507
	$C_{16}v_2 \rightarrow C_3v_6$	$[C_{16}v_2,C_{16}v_1,C_{12}v_4,C_{12}v_5,C_9v_2,C_9v_1,C_5v_4,C_6v_1, C_2v_4,C_3v_1,C_3v_6]$	10	0.503
4	$C_9v_4 \rightarrow C_{10}v_2$	$[C_9v_4,C_{13}v_1,C_{13}v_2,C_{13}v_3,C_{13}v_4,C_{13}v_5,C_{10}v_2]$	6	0.182
	$C_{10}v_2 \rightarrow C_9v_4$	$[C_{10}v_2,C_9v_5,C_9v_6,C_9v_1,C_9v_2,C_9v_3,C_9v_4]$	6	0.203
5	$C_2v_5 \rightarrow C_8v_6$	$[C_2v_5,C_2v_4,C_6v_1,C_5v_4,C_5v_3,C_8v_6]$	5	0.176
	$C_8v_6 \rightarrow C_2v_5$	$[C_8v_6,C_5v_3,C_5v_4,C_6v_1,C_2v_4,C_2v_5]$	5	0.177

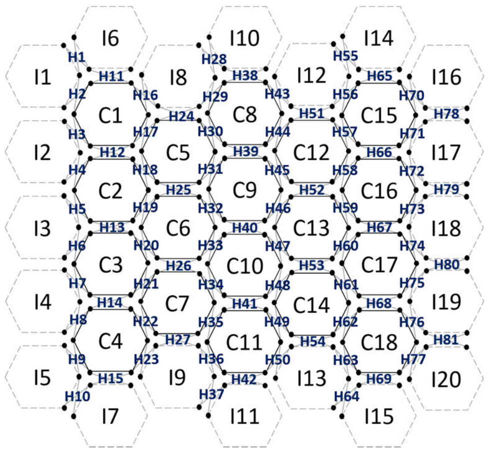


Fig. 7. Graph representation of a larger waveguide mesh. TBUs, actual and imaginary cells are numbered from top to bottom and from left to right. Graph nodes within each hexagonal cell are numbered clockwise. Nodes numeration follows the same criteria than in Fig. 2.

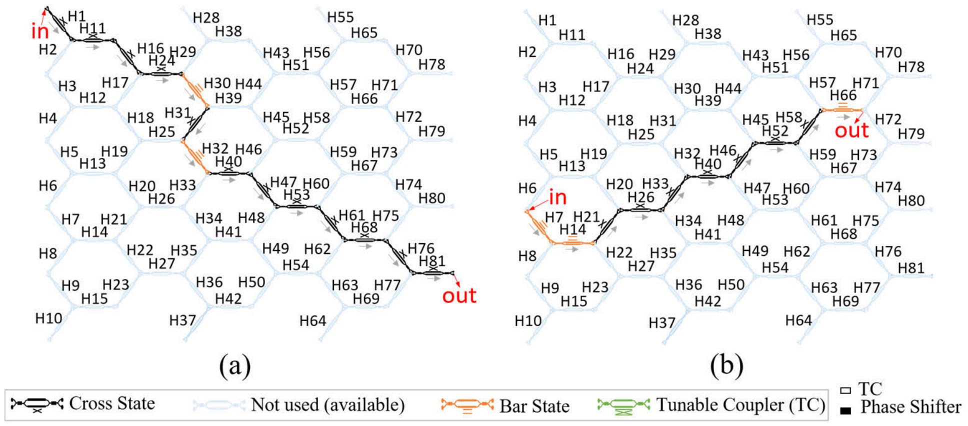


Fig. 8. Synthesis of optical delay lines #1 (a) and #3 (b) from Table 2 in the proposed 81-TBU waveguide mesh.

35 s) to achieve this task is longer than for the small mesh. This is because we were forced to run the algorithm up to the search of 22-TBU-length paths to find a solution, which did not cross any of the damaged TBUs.

We finish this subsection by demonstrating the same multi-in, multi-out capability of the algorithm for the 81-TBU waveguide mesh such as we did for the smaller one. The results can be observed in Fig. 10, where we show the combined synthesis of four different optical paths of 15 TBUs each. We expect that disposing of broader waveguide meshes with a larger number of TBUs facilitates the synthesis of a larger number of optical paths minimizing any potential hampering between them. The elapsed time to finish the process is approximately the sum of the time spent in each individual path synthesis, in the order of hundreds of milliseconds each.

Finally, one could find the optimum value of a delay line with a specific number of TBU or delay modifying the TD expression. This capability is essential for the programming of interferometric structures, as described in the following section.

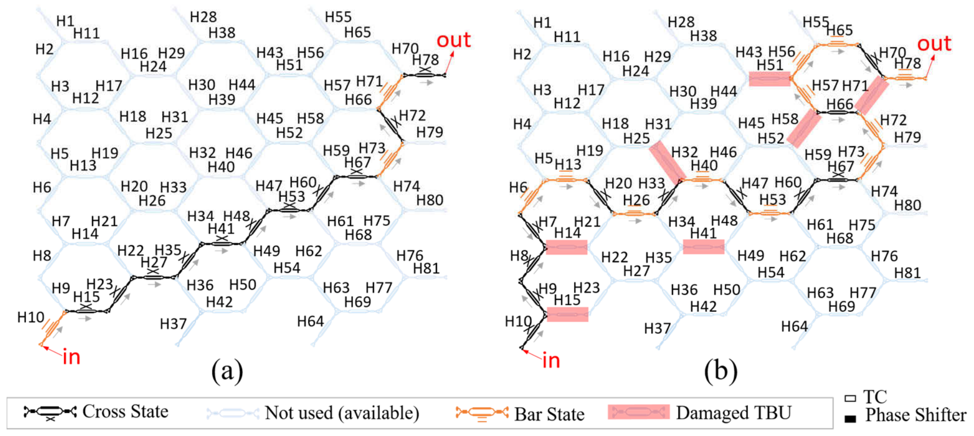


Fig. 9. Demonstration of self-healing and fault-tolerant capability in an 81-TBU waveguide mesh.

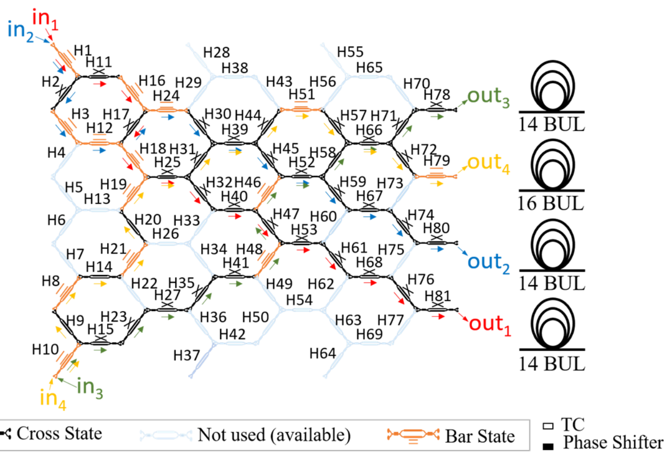


Fig. 10. Synthesis of multiple delay lines at a time in a 81-TBU waveguide mesh. Inputs and outputs are numbered following the order in which their corresponding optical paths were synthesized.

3.2. *Synthesis of interferometric structures*

Most interferometric structures are combinations of optical beamsplitters, combiners and mirrors with delay lines of specific length. This length or length difference impacts the periodic response frequency or free spectral range (FSR). For that purpose, this algorithm can also be applied as a sub-routine for the automated configuration of optical filters employing the simplified *place_and_route* routines reported in [21]. Precisely, a routine performs the placement of the tunable couplers and then the *auto-routing* algorithm presented in this paper works sequentially between the ports of the tunable couplers to define the delay lines. However, to date, there is no fully automated demonstration of the placement of the tunable couplers and combiners. The current approach requires them to be allocated by the user, hence corresponding to an intermediate software complexity level with both, user and processor, sharing responsibilities in the synthesis of such configurations as proposed in [7,8]. In addition, the optimization of the splitting and combination ratios to achieve arbitrary extinction ratios (ER) is beyond the scope of this paper, which only provides and demonstrates automatic mechanisms for establishing optical paths (i.e., to configure each TBU either to cross or bar state).

From the initial parameter requirements, the process changes depending on the filtering architecture to be synthesized. For instance, for the case of an unbalanced MZI, the architecture is defined by two tunable couplers or combiners (TC) joined by two optical paths of different length. In this case, we can first run the algorithm to set the optimum path across one of the MZI's arms using the TBUs, which function as TCs, in order to minimize the overall IL of the device. Afterwards, we perform a second run of the algorithm between the remaining ports of the TCs to find a path length matching the targeted FSR. We can either halt the process as soon as a first candidate comes up or wait longer for more suitable options to appear (featuring lower IL or power consumption, for instance, as shown in Section 2). If we aim to characterize other structures, such as Sagnac filters or optical ring resonators (ORRs) we would only need to run the process once between the available ports of the TBU acting as TCs, along with the same termination protocol, as originally proposed in [21]. These elements are the base for more complex photonic circuits [22].

Figure 11 illustrates the use of our algorithm to synthesize several different well-known interferometric structures in a waveguide mesh. The elapsed time (which as always depends on the cavity length to be obtained by the algorithm) to achieve each task was in the order of ms as can be seen in Table 1. Also note that H15 in Fig. 11(b) and H16 in Fig. 11(f) are employed using all their ports. This TBU configuration property, known as TBU re-usability [7,9,21], can be employed both for independent circuits working in parallel and to accommodate crossings of a delay line or interconnects with itself.

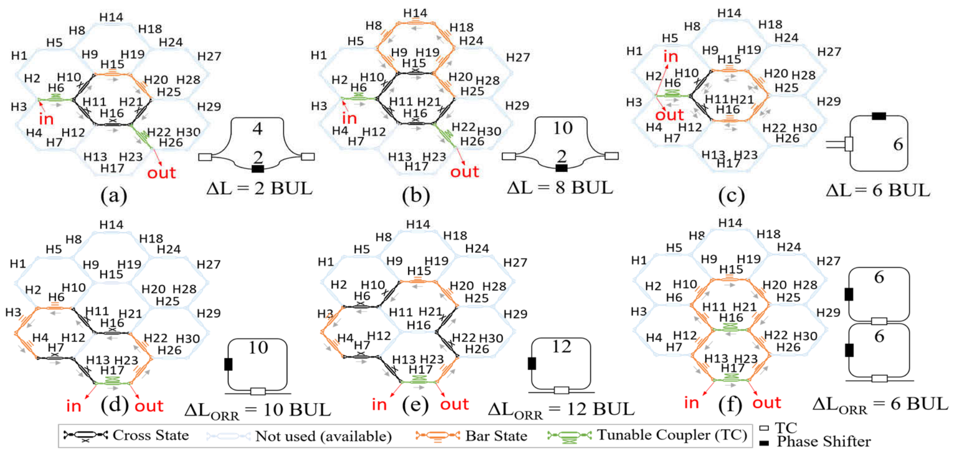


Fig. 11. Synthesis of several interferometric circuits in our small waveguide mesh using our auto-routing algorithm. Clockwise from top left: a 2-TBU MZI (a), an 8-TBU MZI (b), a 6-TBU Sagnac filter (c), a 10-TBU ORR (d), a 12-TBU ORR (e) and a 6-TBU, second order coupled resonator optical waveguide (CROW) (f).

Finally, Fig. 12 illustrates the synthesis of three interferometric circuits using the 81-TBU mesh. Compared to those obtained in Fig. 11, the average elapsed time after ten independent trials to synthesize each is significantly larger: 0.797 s for the 10-TBU MZI of Fig. 11(a), 18.546 s for the 14-TBU ORR of Fig. 11(b) and 0.598 s for the 14-TBU Sagnac filter of Fig. 11(c). This is not only because the number of components of this new mesh is more than the double of the previous one, but also because the synthesized path differences are also considerably larger. As in previous scenarios, we stopped the process as soon as the first result was retrieved. Comparing the elapsed time to the one it takes to perform the process from source node only (around 10-15 min), the improvement is evident.

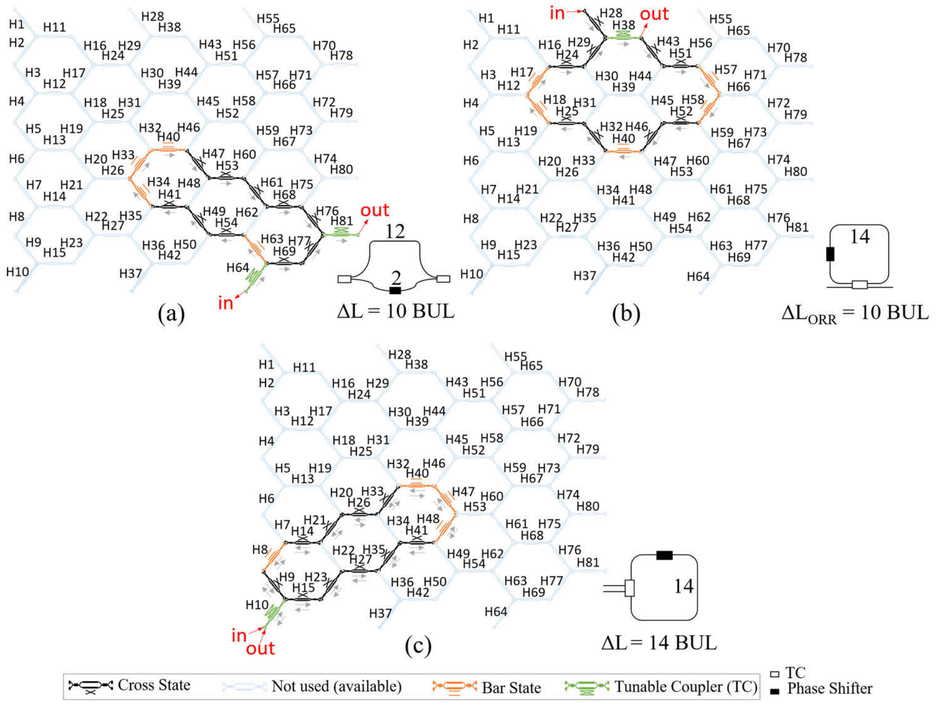


Fig. 12. Synthesis of several interferometric circuits in our large waveguide mesh using the proposed auto-routing algorithm. Clockwise from top left: a 10-TBU MZI (a), a 14-TBU ORR (b) and a 14-TBU Sagnac filter (c).

Note how several of the structures from Fig. 10 occupy separate regions of the waveguide arrangement. A recent experimental demonstration confirms that waveguide mesh arrangements can support multiple operations at the same time through the simultaneous synthesis of several circuits working in parallel with crosstalk values better than 24 dB [23]. Although, the proposed algorithm can only work on one single interferometric structure at a time, one can fix the TBUs employed by one circuit and launch a second algorithm to program a second circuit to coexist.

4. Discussion

The algorithm presented in this work supports optimum self-configuration of optical interconnections and delay lines in a FPPGA core with respect to the main figures of merit of its tunable basic units –such as the IL, the power consumption and its basic unit length– in an elapsed time of a few seconds. During the preparation of this manuscript, another graph-based optimization topology has also been published [24]. In that contribution, a graph is associated to a hexagonal waveguide mesh arrangement topology and demonstrates its application to the synthesis of optimized optical interconnects between the mesh ports with respect to power loss. The same idea proposed in this present contribution of multiple target optimization could be employed to optimize multiple key performance parameters simultaneously. Moreover, our contribution propose the application of auto-routing techniques for the synthesis of delay lines in interferometric circuits and multiple input-multiple output circuit topologies. As an illustrative example, Fig. 1 shows a FPPGA architecture where the algorithm is applied to self-configure a full electro-optical circuit and a pure optical processing engine, simultaneously. Considering the required future work missing in current demonstrations, including this one, there are some features and routines that need to be developed to exploit the flexibility inherent in the FPPGA architecture and general-purpose waveguide mesh arrangements in general. In particular, the programming of a large number of simultaneous paths might be addressed with solutions beyond the sequential application of the algorithm. Future work should address these considerations and provide benchmarking efficiency metrics.

As mentioned in Section 2 and 3, the presented routine requires certain information prior to its execution. First, we need to load the full circuit architecture in the shape of a *graph.* Then, we specify the features to be optimized and the location of the input and output ports that delimit the optical interconnection. As covered in the text for the case of optical filter synthesis, one can feed the periodic frequency of the filter as well, and the algorithm will make the translation. For the case where the IL is being optimized, only optical power monitors placed at the external perimeter are required. In other words, there is no need for characterizing or monitoring the individual IL of each TBU to run the algorithm as long as the delay line to be synthesized connects two input/output ports of the waveguide mesh, as it only considers the accumulated IL of the overall path. For the case where the delay lines or internal interconnections are not connected to the perimeter, for instance to program an interferometric circuit, the algorithm requires the knowledge of each individual TBU IL values. This issue can be addressed either with internal TBU monitoring or by means of periodic pre-characterization routines. To ensure the scalability of the circuit and avoid monitoring system overhead the latter solution is preferred. Pre-characterization routines can be implemented by building up a linear system of equations describing a number of optical interconnections greater than the number of existing TBUs. It should also be noted that a pre-characterization of the power splitting ratio as a function of the applied electrical power is also required as this is the enabler which helps in setting the paths as per desire and hence, achieving the (re-)configuration setting.

Regarding the scalability, we have compared two circuit sizes and different optical interconnection lengths. Clearly, longer interconnections have required longer execution times arising from the nature of the algorithm itself. If finding an optimum path with respect to other feature is not necessary, early stopping conditions could be implemented to speed up the progress to some extent. When it comes to implementing simple optical delay lines (optical interconnections constrained by the number of TBUs), scaling up the size of the mesh does not provide significant delays to the process –which, depending on their length, usually lasts in the order of a few

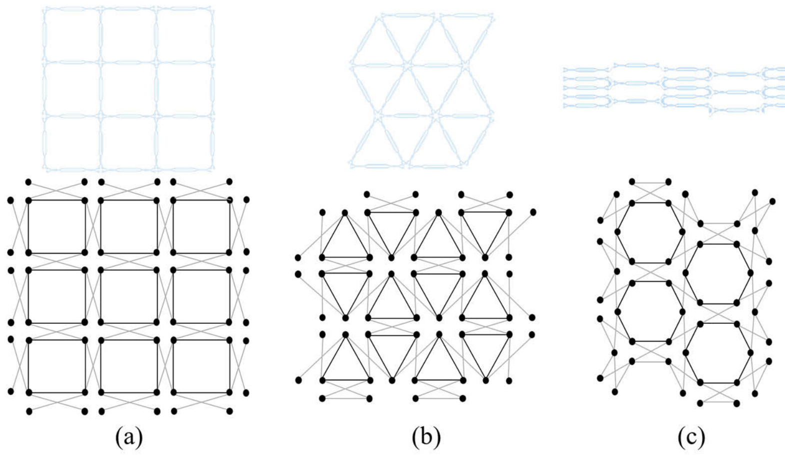


Fig. 13. Several reported mesh topologies along with their corresponding graph representations: (a) rectangular mesh, (b) triangular mesh, (c) planar hexagonal mesh.

seconds. The proposed algorithm can be applied for self-configuration and circuit programming in any arbitrary waveguide mesh arrangement architecture, either geometrical or flattened lattices that allows a higher integration density [10]. Several of these mesh topologies along with their corresponding graph representations appear in Fig. 13. An exciting area of research will deal with the application of the proposed algorithm and its combination with advanced optimization methods, to reduce the number of variables to be optimized and achieve better convergence rates [10].

5. Conclusion

This work proposes an auto-routing algorithm to automatically route light across a FPPGA core through an optimized path with respect to different and combined criteria, including the overall accumulated optical loss of the resulting optical path, the number of traversed elements or/and the required power consumption. In the last feature, an example in a fabricated waveguide mesh arrangement results in power savings between 20.35 and 45.08% for different circuit configurations. The algorithm is expected to help in self-configuration of general-purpose programmable photonic integrated circuits inside the FPPGA core. We have demonstrated its application in a large-scale waveguide mesh arrangement of up to 81 programmable unit cells. Finally, we demonstrated the fault-tolerant and self-healing capabilities that the algorithm provides and its use in multipurpose waveguide mesh arrangements to compensate for fabrication errors across the circuit. This feature becomes a fundamental difference maker between programmable photonics and application specific circuits.

Funding

European Research Council; Generalitat Valenciana; European Cooperation in Science and Technology; Ministerio de Ciencia, Innovación y Universidades; Advanced Instrumentation for World Class Microwave Photonics Research (IDIFEDER/2018/031); Infraestructura para Caracterización de Chips Fotónicos (EQC2018-004683-P).

Acknowledgments

The authors acknowledge financial support by the ERC ADG-2016 UMWP-Chip ERC-2016-ADG-741415, the ERC PoC-2019 FPPAs ERC-2019-POC-859927, the Generalitat Valenciana Future MWP technologies and applications PROMETEO 2017/103 research excellency award, and the COST Action CA16220 EUIMWP, the Advanced Instrumentation for World Class Microwave Photonics Research IDIFEDER/2018/031 and the Infraestructura para caracterizacion de Chips Fotonicos EQC2018-004683-P.

Disclosures

Authors declare no conflict of interest.

References

1. R. Soref, "The Past, Present and Future of Silicon Photonics," IEEE J. Sel. Top. Quantum Electron. **12**(6), 1678–1687 (2006).
2. M. Streshinsky, R. Ding, Y. Liu, A. Novack, C. Galland, A. E.-J. Lim, P. G.-Q. Lo, T. Baehr-Jones, and M. Hochberg, "The Road to affordable Large Scale silicon photonics," Opt. Photonics News **24**(9), 32–39 (2013).
3. D. Inniss and R. Rubenstein, Silicon Photonics: Fueling the Next Information Revolution (Elsevier Science, 2016).
4. M. K. Smit, X. Leijtens, H. Ambrosius, E. Bente, J. van der Tol, B. Smalbrugge, T. de Vries, E.-J. Geluk, J. Bolk, R. van Veldhoven, L. Augustin, P. Thijs, D. D'Agostino, H. Rabbani, K. Lawniczuk, S. Stopinski, S. Tahvili, A. Gilardi, W. Yao, K. Williams, P. Stabile, P. Kuindersma, J. Pello, S. Bhat, Y. Jiao, D. Heiss, G. Roelkens, M. Wale, P. Firth, F. Soares, N. Grote, M. Schell, H. Debregeas, M. Achouche, J.-L. Gentner, A. Bakker, T. Korthorst, D. Gallagher, A. Dabbs, A. Melloni, F. Morichetti, D. Melati, A. Wonfor, R. Penty, R. Broeke, B. Musk, and D. Robbins, "An introduction to InP-based generic integration technology," Semicond. Sci. Technol. **29**(8), 083001 (2014).

5. L. Carroll, J.-S. Lee, C. Scarcella, K. Gradkowski, M. Duperron, H. Lu, Y. Zhao, C. Eason, P. Morrissey, M. Rensing, S. Collins, H.-Y. Hwang, and P. O'Brien, "Photonic Packaging: Transforming Silicon Photonic Integrated Circuits into Photonic Devices," Appl. Sci. **6**(12), 426 (2016).
6. D. Pérez, I. Gasulla, and J. Capmany, "Field-programmable photonic arrays," Opt. Express **26**(21), 27265–27278 (2018).
7. D. Pérez, I. Gasulla, J. Capmany, and R. A. Soref, "Reconfigurable lattice mesh designs for programmable photonic processors," Opt. Express **24**(11), 12093–12106 (2016).
8. L. Zhuang, C. G. H. Roeloffzen, M. Hoekman, K.-J. Boller, and A. Lowery, "Programmable photonic signal processor chip for radiofrequency applications," Optica **2**(10), 854–859 (2015).
9. D. Pérez, I. Gasulla, L. Crudgington, D. J. Thomson, A. Z. Khohkar, K. Li, W. Cao, G. Z. Mashanovich, and J. Capmany, "Multipurpose silicon photonics signal processor core," Nat. Commun. **8**(1), 636 (2017).
10. D. Pérez, "Programmable Integrated Silicon Photonics Waveguide Meshes: Optimized designs and control algorithms," *IEEE J. Sel. Top. Quantum Electron.*, DOI: 10.1109/JSTQE.2019.2948048 (to be published).
11. D. Pérez and J. Capmany, "Scalable analysis for arbitrary photonic integrated waveguide meshes," Optica **6**(1), 19–27 (2019).
12. A. López, Implementation of Self-reconfigurable Integrated Optical Filters based on Mixture Density Networks (Universidad Politécnica de Madrid, 2019).
13. E. W. Dijkstra, "A note on two problems in connection with graphs," Numer. Math. **1**(1), 269–271 (1959).
14. J. McQillan, I. Richer, and E. Rosen, "The New Routing Algorithm for the ARPANET," IRE Trans. Commun. Syst. **28**(5), 711–719 (1980).
15. H. Wang, Y. Yu, and Q. Yuan, "Application of Dijkstra algorithm in robot path-planning," in *Proc. 2nd Int. Conf. Mechanic Automation and Control Engineering*, (IEEE, 2011), pp. 1067–1069.
16. L. McMurchie and C. Ebeling, "PathFinder: A Negociation-Based Performance-Driven Router for FPGAs," in *Proc. 3rd Int. ACM Symp. Field-Programmable Gate Arrays*, (IEEE, 1995), pp. 111–117.
17. M. Tommiska and J. Skyttä, "Dijkstra's Shortest Path Routing Algorithm in Reconfigurable Hardware," in *Proc. Int. Conf. Field Programmable Logic and Applications*, (Springer, 2001), pp. 653–657.
18. A. Sharma and S. Hauck, "Accelerating FPGA routing using architecture-adaptive A* techniques," in *Proc. IEEE Int. Conf. Field-Programmable Technology*, (IEEE, 2005), pp. 225–232.
19. R. J. Trudeau, *Introduction to Graph Theory* (Courier Corporation, 1993).
20. P. Dumais, Y. Wei, M. Li, F. Zhao, X. Tu, J. Jiang, D. Celo, D. J. Goodwill, H. Fu, D. Geng, and E. Bernier, "2×2 Multimode Interference Coupler with Low Loss Using 248 nm Photolithography," in *Optical Fiber Communication Conference, OSA Technical Digest (online)* (Optical Society of America, 2016), paper W2A.19.
21. D. Pérez, Integrated Microwave Photonic Processors using Waveguide Mesh Cores (Universitat Politècnica de València, 2017).
22. C. K. Madsen and J. H. Zhao, *Optical filter design and analysis: a signal processing approach* (Wiley, 1999).
23. A. López, D. Pérez, P. DasMahapatra, and J. Capmany, "Dynamic Reconfiguration in Field-Programmable Photonic Arrays," presented at *the 45th European Conference on Optical Communications*, Dublin, Ireland, 23-26 Sept. 2019.
24. X. Chen and W. Bogaerts, "A Graph-based Design and Programming Strategy for Reconfigurable Photonic Circuits," in *Proc. of Photonics Society Summer Topical Meeting Series* (IEEE, 2019), paper ME2.2.

59.64.129.234 {ts '2021-01-14 10:02:25'}