Automatic Waveguide Balancing Using Point Set Operations

Won Suk Lee*, Yusheng Bian*, Thomas Weeks III, Michal Rakowski, Francis O Afzal, Takako Hirokawa, Asif Chowdhury, Rod Augur, Jae Gon Lee, Alexander Martin and Ken Giewont

GlobalFounrdries, 400 Stone Break Rd Extension, Malta, NY 12020, USA

**These authors contributed equally to this work*

wonsuk.lee@globalfoundries.com; yushengbian@globalfoundries.com

Abstract: An algorithm based on point set operations is developed to solve the waveguide length-balancing problem in silicon photonics layout. The method is applicable to complex photonic circuits incorporating multiple waveguide levels (e.g. Si and SiN).

OCIS codes: (130.0250) Optoelectronics; (130.2790) Guided waves; (130.5990) Semiconductors; Guided waves; (130.1750) Components; (250.5300) Photonic integrated circuits; (250.3140) Integrated optoelectronic circuits

1. Introduction

As silicon photonics (SiPh) technology shows promise for high-speed integrated circuits (ICs), designers are in great need of automatic waveguide (WG) routing tools to facilitate their design cycle. Unlike automatic metal routing in electronic ICs, WG routing is more challenging due to additional restrictions to optimize the performance of photonic ICs (PICs), such as the length of the WG, the number of WG bends, and the guided modes and/or polarizations in the WG channel. Many commercial CAD vendors currently provide automatic WG routing tools focused on connecting one point to another using a given length and number of bends, while circumventing obstacles (e.g. other WGs or devices) [1-3]. Some additional WG routing methods are presented in [4] and [5] for specific applications. However, none of these methods demonstrate the ability to simultaneously balance the total length and number of bends in multiple WG paths.

In this paper, an automatic WG length-balancing algorithm is presented which is based upon simple operations on sets of Cartesian coordinates ("point sets"). The algorithm presented here is useful in cases where multiple paths need to be connected to multiple devices [6], and it can be adapted to complex PICs containing multiple WG levels such as Si and SiN [7-9]. For example, the algorithm can find useful applications in the design of ILOTs (In-Line Optical Testing system) layouts for wafer-level testing [7], as well as LiDAR (Light Detection and Ranging) systems, both of which require balanced WG paths between input/output optical I/O and functional photonic components.

2. Waveguide Balancing Algorithm

The problem of balancing multiple WG paths can be modeled, without loss of generality, as a rectangular box, with inputs and outputs on the left and top edges, respectively. All other cases can be viewed as rotations or cascaded composites of this base case. To further simplify the problem and to facilitate calculations on point sets, a discrete Cartesian grid is imposed on the box, such that the unit length is equal to two times the WG bend radius. All input/output points are then snapped (rounded) to this grid (Note that any input/output points which were initially off-grid will be restored to their original positions in the last stage of the algorithm). The final step of initialization is, for each input point, to create a path p_i connecting the input to its corresponding output using the minimum number of 90-degree WG bends. The following sets are then created as shown in Table 1.

Sets	
P	The set of all paths p_i, initially indexed by length. (Thus p_1 is the shortest, $p_{\|P\|}$ the longest.)
A_i	The set of all grid points on path p_i
B_i	The set of all bend (corner) points on path p_i
S_i	The set of all grid points on a unit square (thus $\|S_i\| = 4$) which extends from bend point $b \in B_i$
K	The set of differences k_i between the number of bends in the bendiest path, and the number of bends in path p_i
Integers	
$\|P\|$	The number of waveguide paths
l_i	The difference between the length of the longest path, and the length of path p_i
k_i	The difference between the number of bends in the bendiest path, and the number of bends in path p
Rules	
(Rule.1)	*Path Length-Difference Update:* XOR(B_i, S_i) updates l_i by $2\times(3 - \|A_i \cap S_i\|)$
(Rule.2)	$\|B_i \cap S_i\|$ *Lower/Upper Bounds:* $\|A_i \cap S_i\| - 2 \le \|B_i \cap S_i\| \le \|A_i \cap S_i\|$
(Rule.3)	*Number of Bends Update:* XOR(B_i, S_i) updates $\|B_i\|$ by $2\times(2 - \|B_i \cap S_i\|)$

Table 1. Definitions of Sets, Integers, and Rules

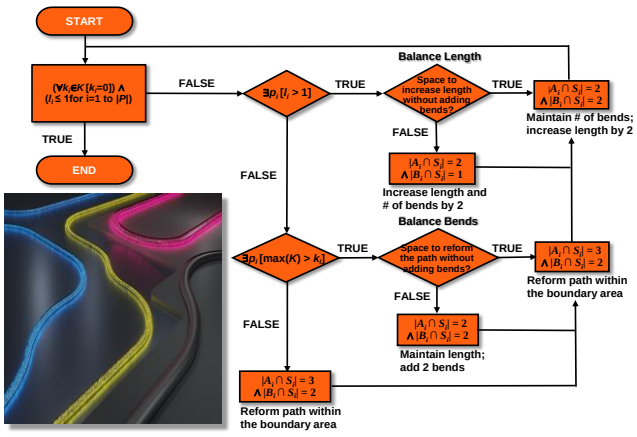

Fig 1. Flow diagram of the WG balancing algorithm

The proposed algorithm performs the following point set operations iteratively on each p_i for $i = 1$ to N, until $k_i = 0$ for all $k_i \in K$, and $l_i \leq 1$; i.e. until all WG paths have the same number of bends and a length difference of no more than one unit length:

(1) When $l_i > 1$, apply (Rule.1) and (Rule.3) to find a unit square S_i which increases the total length without increasing the number of bends; i.e. $|A_i \cap S_i| = 2 \wedge |B_i \cap S_i| = 2$. In the case where the initial placement of WGs is too close, find S_i in a direction which increases both the total length and the number of bends, i.e. $|A_i \cap S_i| = 2 \wedge |B_i \cap S_i| = 1$.

(2) When $\max(K) > k_i$ at p_i, find S_i such that $|A_i \cap S_i| = 3 \wedge |B_i \cap S_i| = 1$ in order to increase the number of bends while maintaining the path length. When there is not enough room to re-form the path, an S_i will not be found. Instead, find an S_i satisfying $|A_i \cap S_i| = 3 \wedge |B_i \cap S_i| = 2$. This condition guarantees the current path is re-formed without any changes in either the total length or the number of bends. In the next iteration, the number of bends is examined again.

(3) If neither of conditions (1) or (2) are met, search for S_i satisfying the condition $|A_i \cap S_i| = 3 \wedge |B_i \cap S_i| = 2$. The purpose of this step is to give sufficient room for extending other paths.

Iterating the above point set operations results in all paths having the same number of bends, but a difference in total WG length may still exist due to the initial snapping of the input/output points to the grid. By recalling that all of the final WG paths are spaced at least one unit length apart, it is easily proved that restoring snapped input/output points to their original positions will not overlap other WG paths. Therefore, merely shifting the balanced paths and extending straight WGs at the input and output points, will resolve the quantization error from grid approximation.

An additional benefit of this proposed method is that it can be directly applied to the case of "heterogeneous WG" paths or WGs formed at multiple levels with different materials; e.g., a WG path consisting of alternating segments of different WG materials such as Si and SiN. Paths are already quantized by unit length segments of grid after the iterations have finished but before executing the last step of correcting approximation errors. Therefore, using the set A_i of grid points of path p_i, the location and number of segments of path p_i may be determined and even replaced with other types of WGs. WG path balance is maintained by replacing the same number of segments on all paths $p_i \in P$. The complete flow of operations are described in Fig. 1.

3. Example Cases

Fig. 2 illustrates the flow of the WG balancing algorithm, starting from an initial state with four inputs and outputs as shown in Fig. 3(a). The WGs are shown in green. The green-and-red triangular shapes on the left edge (inputs) are the grating couplers. Unit squares are shown in orange. Each WG bend is one unit square in length.

At initialization, there is no unit square S_i which can be added to p_1, p_2, and p_3 to increase length without also adding bends. Therefore, each path with $l_i > 1$ (viz. p_1, p_2, p_3) has to increase both its length and number of bends. So the maximum number of bends is incremented by two, and two bends are added to the longest path (p_4) without increasing its length, using unit squares S_i, shown in orange in Fig. 2(a). Two bends are also added to p_1, p_2, and p_3, and their lengths are increased. At this stage, Fig. 2(a) shows p_3 now matching p_4 in length and number of bends, but p_1 and p_2 do not have sufficient routing area to match the length of p_4. To further increase the lengths of the shorter paths p_1 and p_2, additional bends are required, while the lengths of paths p_3 and p_4 already match, requiring only additional bends. Thus, paths p_3 and p_4 are re-formed as shown in Fig. 2(b). Now the lengths of paths p_2, p_3, and p_4 are balanced, and only the length of path p_1 needs to be increased in order to complete the balancing. Since p_1 cannot be further stretched to the left, additional bends need to be added to all paths. However, p_3 and p_4 must be re-formed to accommodate additional bends, such that the bends do not increase length. This re-formation is shown in Fig. 2(c), and the additional bends are subsequently added to p_3 and p_4 in Fig. 2(d). At last, p_1 can grow vertically until it achieves the required length, and all paths are balanced.

Fig. 3 shows a summary of this work, a comparison between paths that were manually balanced, and paths balanced automatically using the proposed method, evaluated in the Cadence® Virtuoso® Design Environment's SKILL programming language. An example case study of an ILOTs layout is shown in Fig. 3(b) [7]. Fig. 3(a) shows the initialized state of the algorithm: All four given paths are drawn on the unit grid with the minimum number of bends. Note that the first number of bends can differ among each path depending on the positions of input/outputs (not pictured). Algorithmically-balanced paths are shown in Fig. 3(c). Each orange square in the figure is equivalent to a point set S_i and indicates how point set operations were performed during iteration. The resulting balanced paths in Fig. 3(c), compared to the manually-balanced paths in Fig. 3(b), are ~50 μm shorter in length and have one less bend. It can be claimed as another advantage of this method that resultant paths have shorter lengths and fewer bends unless

a certain length and a certain number of bends are required for the designer to introduce additional loss as desired. It is possible to create paths with given lengths by giving offset values to l_i.

Finally, Fig. 3(d) shows the algorithm applied to the heterogeneous case of a path containing Si and SiN WGs. Green lines represent crystallized silicon WG, and pink lines represent SiN (with implied taper connection).

One challenge of balancing out WG path lengths is that there may exist many different or no solutions, depending on the area through which paths are allowed to route. The proposed method produces a solution only if there is enough room to route, given the number of paths $|P|$ and the placement of the inputs and outputs. Hence there are some limitations to this method, including optimizing area efficiency, finding the absolute minimum number of bends, the absolute shortest total length, and determining whether any solution exists at all. We are currently developing improved algorithms to overcome these limitations.

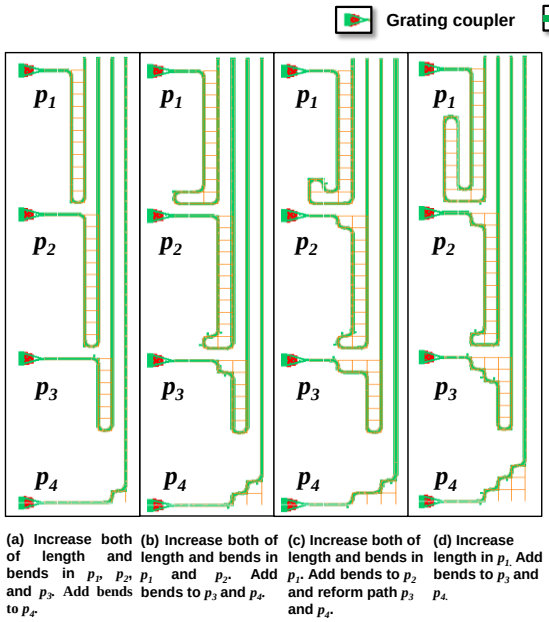

Fig. 2. WG balancing flow.

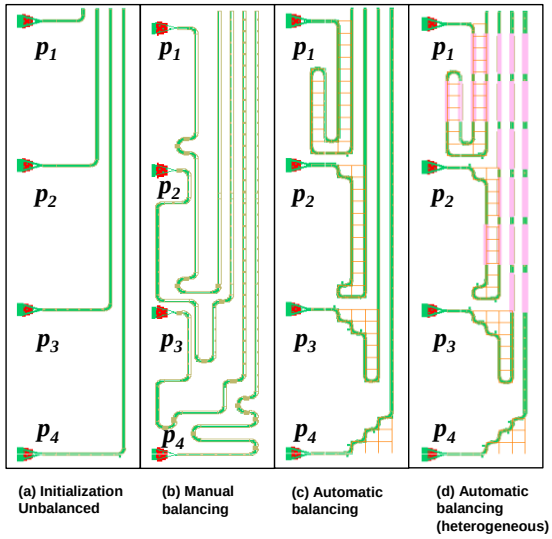

Fig. 3. Examples of balanced WG paths for different cases.

4. Summary

We present a WG-balancing algorithm for point-to-point routing scenarios involving multiple WG paths (e.g., from input or output grating couplers to the input or output of photonic devices), aiming to generate balanced routes without manual intervention. In the ILOTs layout test case, the algorithm was even able to find a solution with a shorter total length compared to manual balancing. Currently, phase-aware routing is mostly done manually or using a dedicated script [10]. The method presented here improves upon the current practice in that it provides a general solution to the phase-aware routing problem which can be applied to different types of circuits with varying topologies.

5. References

[1] A. Boos, L. Ramini, U. Schlichtmann and D. Bertozzi, "PROTON: An automatic place-and-route tool for optical Networks-on-Chip," *2013 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, San Jose, CA, pp. 138-145 (2013).

[2] J. S. Orcutt and R. J. Ram, "Photonic Device Layout Within the Foundry CMOS Design Environment," in *IEEE Photonics Technology Letters*, vol. 22, no. 8, pp. 544-546 (2010).

[3] C. Y. Lee, "An Algorithm for Path Connections and Its Applications," in *IRE Transactions on Electronic Computers*, vol. EC-10, no. 3, pp. 346-365 (1961).

[4] A. Annoni *et al.*, "Automated Routing and Control of Silicon Photonic Switch Fabrics," in *IEEE Journal of Selected Topics in Quantum Electronics*, vol. 22, no. 6, pp. 169-176 (2016).

[5] T. Krishna *et al.*, "Automatic place-and-route of emerging LED-driven wires within a monolithically-integrated CMOS−III-V process," *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, Lausanne, pp. 344-349 (2017).

[6] N. Bai *et al.*, "Mode-division multiplexed transmission with inline few-mode fiber amplifier," Opt. Express 20, 2668-2680 (2012).

[7] K. Giewont *et al.*, "300-mm Monolithic Silicon Photonics Foundry Technology," in *IEEE Journal of Selected Topics in Quantum Electronics*, vol. 25, no. 5, pp. 1-11 (2019).

[8] Y. Bian *et al.*, "Monolithically integrated silicon nitride platform," in *Opt. Fiber Commun. Conf.*, Th1A.46 (2021).

[9] Y. Bian *et al.*, "Towards low-loss monolithic silicon and nitride photonic building blocks in state-of-the-art 300mm CMOS foundry", in *OSA Frontiers in optics/Laser science*, FTu6E.3 (2020).

[10] W. Bogaerts and L. Chrostowski, "Silicon Photonics Circuit Design: Methods, Tools and Challenges," *Laser Photonics Rev.* 12(4), 1700237 (2018).