A Cross-Section Module To Bridge PIC Layout and Simulation Methods

L. Rossi,[1] M. Passoni,[1] and R. G. Broeke[1]

[1] Bright Photonic B.V., Horsten 1, 5612AX Eindhoven, the Netherlands

This paper presents a versatile cross-section module and control layer in Nazca Design to interface between mask design and simulation tools, e.g. to find effective indices, and create and compare compact models. The module provides a standardized way to define a 2D cross-section geometry and material information, and subsequently port it to one or more simulation tools. 3D structures are created by combining cross-sections with mask layout. After running a simulation the module extracts and converts the simulation result into a standard data format for building compact models and/or comparisons in the analysis tool of choice, e.g. Matplotlib, Pandas or other.

Introduction

Integrated photonics has matured quickly in the past few years, and many technology platforms are now available, for example via multi-project wafer runs. As the number of components in photonic integrated circuits increases, the design flow reaches deeper into the circuit level, which is based on compact models of components, such as S-matrices in case of linear models.

Since photonics is an analog discipline, a different wavelength, polarization, or slight variations in the device geometry quickly requires new simulations for the generation of reliable models. Fortunately, there are many simulation tools and methods available, both free and/or open source, such as Meep [3] and wgms3d [4], as well as commercial, such as Lumerical Solutions [5]. However, each tool comes with its own interface and syntax. Therefore, generating the same geometry across multiple solvers for simulation input can become cumbersome and error-prone. At the output side (commercial) tools typically come with some internal result analysis for items like indices and mode characteristics. However, analysis definitions and grid handling may differ per tool, and with data still inside the different tools direct comparisons between tools at data level are difficult. Thus, a designer is often faced with a plethora of waveguide structures and materials, as well as multiple tools having different input and output interfaces and data formats.

To address these bottlenecks, this paper presents a waveguide cross-section geometry module (XSG) embedded within the Nazca Design framework [6]. The module interfaces between mask design (waveguide and compact model design) and simulation tools. The module provides a versatile and easy way to define a generic cross-section in Nazca, and ports it to the desired tool with a standardized syntax. It can make use of Nazca's mask layout information to create 3D structures. Simulations can be launched from the Nazca API using standardized commands; results are collected from the simulation tool(s) and converted in a common data format for further analysis by the designer.

Methods

Most integrated optical components consist of a uniform cross-section across the wafer. Such devices can be decomposed into a two-dimensional cross-section and a one-dimensional polyline with locally varying width, as shown in Figure 1.

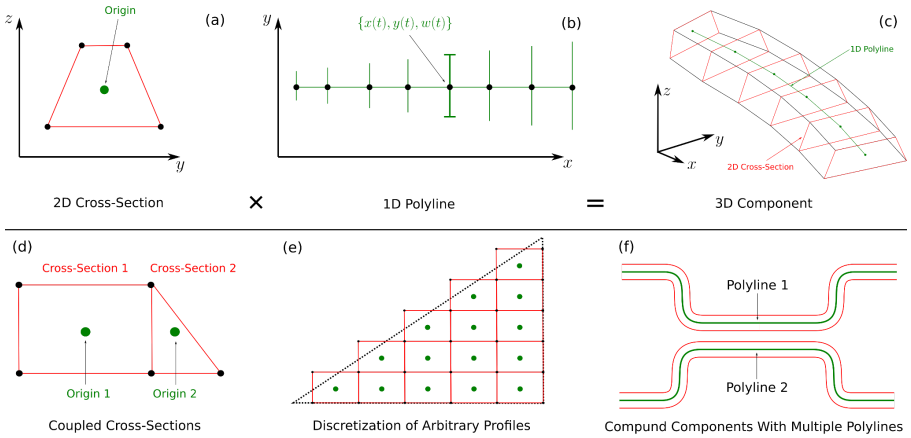

Figure 1: XSG module working principle. Decomposition of a 3D component (c) into a 2D cross-section (a) and a 1D polyline (b). The former (highlighted in red) describes the cross-sectional geometry and the materials therein employed (here left blank for simplicity); the polyline (highlighted in green) contains the path of the cross section and possible parameterized parameters of the cross section, such as a local width. Composite shapes can be used for more complex cross-sections and/or to simplify/enable interfacing with some tools. Examples are combining cross-sections having different polylines (d), discretization of arbitrary profiles with rectangular shapes (e), and a compound component (top-view/directional coupler) drawn with two separate polylines (f).

The cross-section contains information on the materials and the cross-sectional geometry and is defined with the XSG module; the one-dimensional polyline describes the (x, y) coordinates of the central points of the polygons on the mask and the local width w associated with them as $\{x(t), y(t), w(t)\}$, that is, a function of a normalized parameter $t \in [0, 1]$. This reuses the same concepts and methods as other mask elements, such as interconnects in Nazca. The XSG module contains two-dimensional shapes characterized by the vertices and the refractive index, which are defined by the user with three geometrical objects without an intrinsic orientation and a first-in-last-out masking property: trapezoids with variable sidewall angle, circles, and polygons with custom vertices.

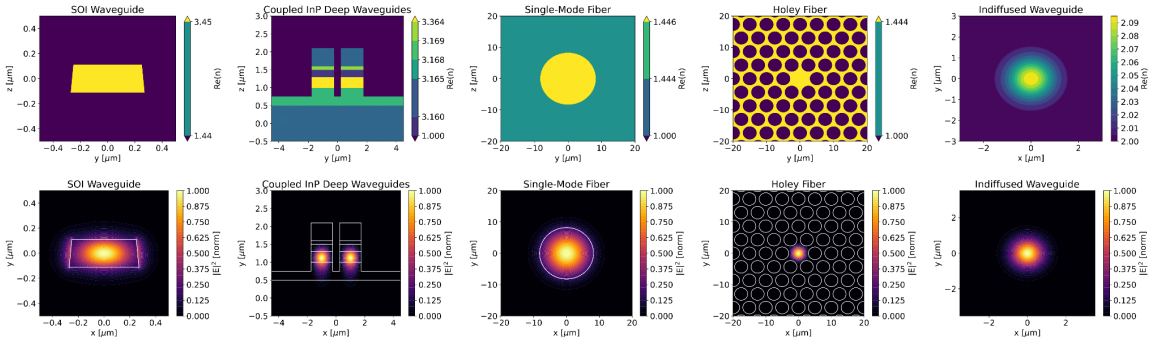

Figure 2: Example geometries that can be handled with the cross-section module and the associated fundamental mode electric field intensity at 1550 nm calculated with Lumerical MODE in these examples. Left to right: box waveguide, coupled waveguides, single-mode fiber, holey fiber, and in-diffused waveguide with Gaussian profile.

Figure 2 shows non-exhaustive examples that can be generated with the cross-section module. Rectangular geometries and circular geometries can be handled with trapezoids and circles, respectively; any other profiles can be directly defined from the vertices or can be imported from a function of position (e.g. diffusion profiles) by sampling its contour lines. Figure 1 shows how 3D components can be generated unifying the top- and side-view by extruding two-dimensional cross-sections into a set of three-dimensional vertices that describe polyhedra with a position-dependent width defined by the polyline.

The simulation flow with the XSG module is shown schematically in Figure 3. Overall, one design associated with one cross-section can be simulated with one standard command across multiple solvers to acquire a set of results represented in a common data format.
Firstly, the layout, the cross-section, and the index models of the materials are instantiated in Nazca. Then, the XSG module generates the vertices of the cross-sectional shapes and associates to them the material indices, which are extracted by evaluating the index models at the desired simulation settings. Combining the cross-sectional vertices, the refractive indices and the top-view polylines generates a unified, tool-independent geometry description. Furthermore, cross-sections can be parametrized with functions to easily perform sweeps by feeding different input settings into the module. Generating 3D simulations from the framework without exporting the layout to file and subsequently loading it in a tool allows one to retain valuable information such as the coordinates of Nazca pins, useful to place sources and monitors.
After the unified geometry with vertices and indices is defined in Nazca, it is translated into the desired tool description with and the simulation is then run by translating the desired settings, such as wavelength and polarization, into a set of standardized commands. Each tool requires a dedicated interface for the translation of the geometry and the commands: if a Python API is available, it is used directly; if a program is executed as a binary from the command line, the geometry is saved into a file and then run from Python as a shell command. Tools that do not offer an accessible interface cannot be used. When the interface communicates to the tool interface that the simulation is finished, the module extracts the results from the tool by querying the API or by reading the output files generated by the solvers. Eventually, Nazca converts the data into a common format including metadata for further analysis.

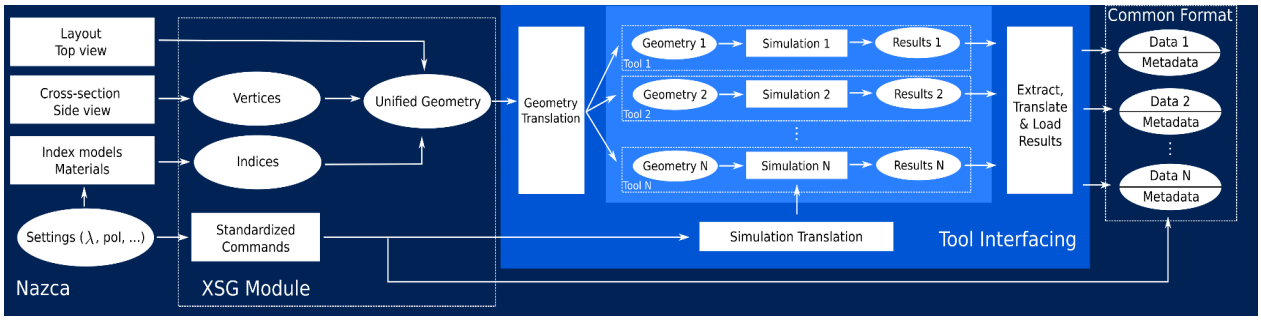

Figure 3: Simulation flow with the cross-section module. Vertices and indices are generated from Nazca from the side-view and material models and are joined to the layout into a unified, tool-independent geometry. Such geometry is exported to tools through a translation, and simulations are run through translated standardized commands. Results are extracted, translated and loaded into Nazca in a common format for further comparison with simulation metadata generated by the standardized command within the XSG module.

Examples

This section presents two application examples to demonstrate a 2D and 3D problem handled by the XSG module. They show how from a single geometry and layout definition a comparison between results across tools and simulation methods is made. As a first example we report on the calculation of the modality of a silicon nitride asymmetric double stripe.

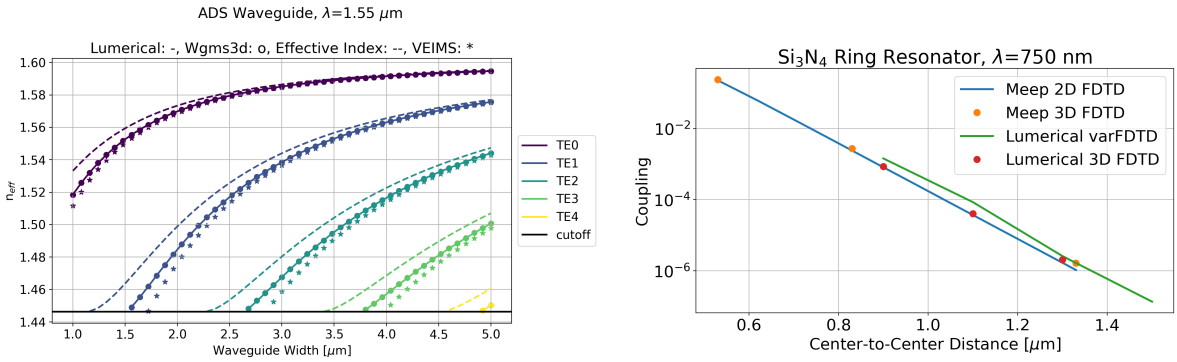

Figure 5: Modality calculation of a silicon nitride asymmetric double stripe waveguide performed with Lumerical (solid), wgms3d (circles), a variational mode solver (stars), and the effective index method (dashed) (a) and TE0 to TE0 transmission in a ring resonator as a function of ring-to-bus distance with Meep 2D (solid blue) and 3D (orange circles) FDTD, as well as Lumerical varFDTD (solid green) and 3D FDTD (red circles) (b).

waveguide, as shown in Figure 5a. This only requires a 2D geometry provided as a parametrized cross-section. The geometry from the cross-section module was fed into Lumerical, wgms3d, a variational eigenmode solver [7], and a multilayer solver performing the effective index method. As a second example we simulate a 3D silicon nitride ring resonator coupled to a straight bus waveguide with slanted sidewalls, where the power coupling from bus to ring as a function of the center-to-center distance is depicted in Figure 5b. The same geometry was sent to Meep 2D- and 3D-FDTD through “prisms”, as well as Lumerical varFDTD and 3D FDTD.

Conclusion

We presented a cross-section geometry (XSG) module within Nazca-Design to interface between mask design and 2D and 3D simulations tools. A single cross-section geometry is exported to multiple tools. Simulations are run from Nazca and results can be retrieved in a standardized data format for further comparison and analysis by the designer.

Acknowledgments

This work was supported by the European Commission within the project InPulse, grant number 824980. We would like to thank Thjis van den Brink for implementing a Python version of the variational eigenmode solver.

References

[1] Ardavan F Oskooi, *et al.* Meep: A flexible free-software package for electromagnetic simulations by the fdtd method. Computer Physics Communications , 181(3):687702, 2010.

[2] wgms3d - Full-vectorial waveguide mode solver. www.soundtracker.org/raw/wgms3d/

[3] Lumerical: High-Performance Photonic Simulation Software. www.lumerical.com

[4] Nazca-Design™ photonic IC design framework https://nazca-design.org

[5] Alyona Ivanova, *et al.* A variational mode solver for optical waveguides based on quasi-analytical vectorial slab mode expansion. arXiv preprint arXiv:1307.1315 , 2013.