Journal of Computational Science xxx (2013) xxx–xxx

Contents lists available at SciVerse ScienceDirect

Journal of Computational Science

journal homepage: www.elsevier.com/locate/jocs

Improving the design cycle for nanophotonic components

Martin Fiers*, Emmanuel Lambert, Shibnath Pathak, Bjorn Maes, Peter Bienstman, Wim Bogaerts, Pieter Dumon

Photonics Research Group (INTEC), Ghent University – IMEC, Sint-Pietersnieuwstraat 41, B-9000 Ghent, Belgium

ARTICLE INFO

Article history:
Received 24 February 2011
Received in revised form 17 February 2013
Accepted 17 May 2013
Available online xxx

Keywords:
Nanophotonics
Designing and modeling optical components
Optical circuit design
Parametrized cell
Python

ABSTRACT

We present IPKISS, a software framework that greatly simplifies the design of nanophotonic components. In this approach, all steps in the workflow are based on a single high-level definition of the component, in a Python script. Because there is only one description, the design flow becomes less error prone due to incorrect definitions, and the overall reproducibility is greatly improved.

Furthermore it enables easy closed-loop modeling of components and circuits. Also, previous work can easily be built upon because lower level blocks can seamlessly be replaced by new blocks. While we illustrate the application in photonics, this software and the used design patterns can be extended to other domains such as RF design and to multidomain physics such as opto-electronics.

1. Introduction

In a typical research or design environment, fabrication of micro-and nanoscale devices is an expensive process with long turnaround times. Prior to submitting a design for fabrication, these devices are typically modeled and simulated in software. For example, in the field of nanophotonics, electromagnetic simulations are used to calculate how light propagates through such a device. Often it is also required to perform tolerance analysis on the design parameters as well as on effects of the fabrication process. One major difficulty that arises when designing these devices is that the different simulation tools have their own user interface and moreover have their own representation to define components. Defining these devices in different tools is a laborious job, and there is a considerable risk of introducing errors in the specification of the device in each tool.

The main characteristic of our approach is that a component is defined only once on a high level [1]. This component is available as a parametrized cell (PCell), a concept originating from the design of electronic circuits. Then, the necessary representations (e.g. a discretized matrix representing the component, a cross-section, a list of polygons, port positions) to drive the different tools (simulation, visualization, routing) are extracted from this definition. The transition to different simulation tools should only be written once in a generic way, which makes simulations much less error prone. It is also much easier to reproduce earlier results and to change sub-parts of the design. In this way, many variations can easily be compared to one another (e.g. a different simulation method, an improved component, or a modified design).

Python is our programming language of choice. The main reason for using this programming language is the flexibility which it offers: it can be used to make very complex software designs, yet it has a relatively low threshold for researchers without extensive programming skills. Our software toolset revolves around a central design framework called IPKISS [1], which can interface with different in-house and third-party simulation tools.

The paper is structured as follows: as the reader might not be familiar with photonics, we very briefly describe this specific research field in Section 2. In Section 3, we illustrate a typical workflow, i.e. the steps needed to design a nanophotonic component. We show which design problems typically arise and demonstrate how the software framework improves this flow. In Section 4, the technical design and implementation of the framework is described, and in the fifth section we illustrate how we use the software to efficiently design a complex optical component: An Arrayed Waveguide Grating. We conclude by providing license information. As previously noted, it is easy to extend this architecture beyond the horizon of photonics: electronic design, multidomain physics and so on. Throughout the paper, we use Python code to explain several core concepts. The code aims to be descriptive rather than to explain all details.

* Corresponding author. Tel.: +32 92643272.
E-mail addresses: martin.fiers@intec.ugent.be, mfiers@gmail.com (M. Fiers), emmanuel.lambert@intec.ugent.be (E. Lambert), shibnath.pathak@intec.ugent.be (S. Pathak), bjorn.maes@umons.ac.be (B. Maes), peter.bienstman@intec.ugent.be (P. Bienstman), wim.bogaerts@intec.ugent.be (W. Bogaerts), pieter.dumon@intec.ugent.be (P. Dumon).

1877-7503/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.jocs.2013.05.008

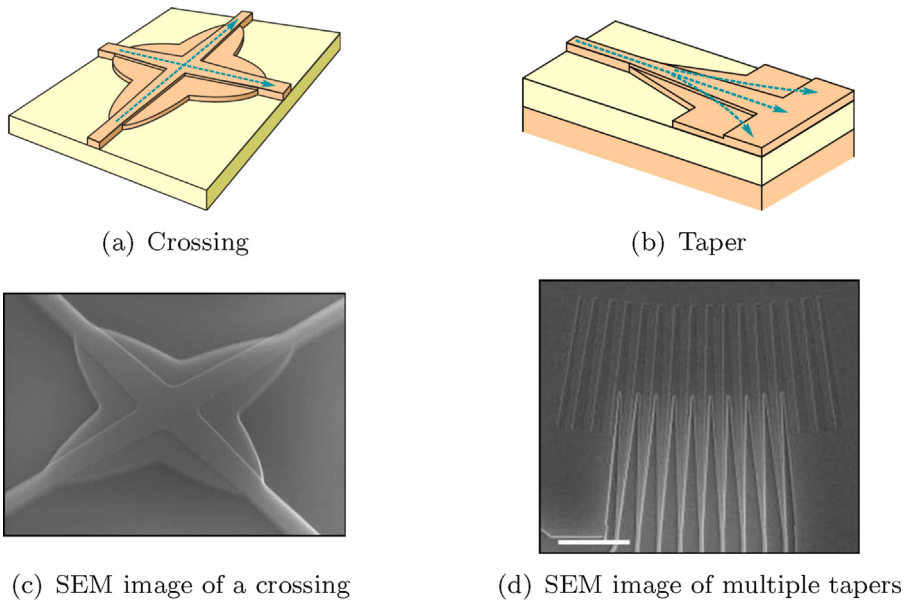

Fig. 1. Some examples of nanophotonic subcomponents, used for designing small integrated optical circuits. Because a nanophotonic circuit is planar, crossings (left) are sometimes needed. Tapers (right) are used to spread light from a narrow waveguide to a broad one. On the bottom, Scanning Electron Microscope (SEM) pictures of the fabricated devices are shown.

2. Photonics

Photonics is the field of manipulating, generating and detecting light (photons) by means of optical components. This is in contrast to electronics, in which electrons are the information carriers. Some examples of photonic devices are: lasers, optical receivers and transmitters, CD/DVD drives and LED lighting. A recent trend in photonics is the drive towards miniaturization of components, and integrating many of them on a single chip. These so-called (nano)photonic integrated circuits have a better performance, are more robust, and consume less power than bulk photonics, low-contrast integrated photonics and electronics. One excellent material for making such optical chips is silicon. Silicon has very low absorption losses in the wavelength range that is used for fibre-optic communications (1300 nm and 1550 nm). Fortunately, silicon is already widely used in electronic chip fabrication, so we can reuse standard Complementary Metal Oxide Semiconductor (CMOS) technology to manufacture photonic chips. In this technology, the silicon on insulator (SOI) wafer is patterned using deep UV lithography [2]. This opens the door to wafer-scale fabrication of nanophotonic chips, leading to devices that can be manufactured in large volumes at low cost.

A few subcomponents of a nanophotonic circuit are displayed in Fig. 1. The resulting device consists of submicrometer wide silicon lines on top of a thick glass layer. Because silicon has a very high index of refraction, the submicron line acts as a waveguide for light: electromagnetic waves with wavelengths between 1.3 μm and 1.55 μm can travel along the line (a so-called "photonic wire") without much loss.

By optimizing the geometry of the silicon, the light can be manipulated. Fig. 1(a, c) shows a crossing of two waveguides, where the geometry is engineered such that there is no *crosstalk* between the waveguides. In Fig. 1(b, d) we change the width of the silicon around the *core* of the waveguide and then stop the waveguide, so that light can diffract in the thin layer of silicon on the chip.

3. Workflow for designing a component

To illustrate the problems associated with a manual workflow (that is, before adopting the framework), as well as the innovation brought by our framework, we will discuss the workflow for designing a typical photonic integrated component: a multimode interferometer (MMI). Although we use an optical component to illustrate the workflow, readers from other research domains might identify the same or similar problems based on their own experience.

In Section 3.1 we briefly introduce this device and its typical design steps. We show that in the classical workflow (3.2) there are a lot of manual interactions, leading to a slow, and more importantly an error-prone workflow. A workflow based on our software (3.3) shows how one can circumvent these problems.

3.1. Example device: MMI

We will illustrate our workflow using a device that splits the light in a waveguide into two equal parts, an important building block in photonic IC design. It is called a multimode interferometer (MMI), and is depicted in Fig. 2. This example is representative to many photonic design problems and is practiced by most photonic designers today, irrespective of the specific tools they use in each step of the problem. The MMI consists of a sequence of waveguide elements of different widths and shapes. Each waveguide supports a number of electromagnetic waveguide modes, i.e. eigensolutions of the light distribution in the dielectric medium.

There are several aspects to modeling this device, which are illustrated in Fig. 2. When exciting the MMI with a mode in an input waveguide, one needs to know the shape (spatial distribution) of this mode, called the mode profile. We calculate this waveguide mode profile using an eigenmode solver (Fig. 2, top left). The mode has a gaussian-like profile, as shown in the figure. The mode profile is then entered as input for a full-wave time-domain simulation to calculate the light propagation in the device (Fig. 2, top right). As three-dimensional (3D) full-wave simulations are computationally very intensive, one will first run an approximate simulation in 2D using well-known approximation methods and only then run full 3D simulations.

In order to get a highly accurate representation of the device characteristics, a 3D simulation is then performed. From this simulation, the scatter matrix is extracted, leading to a high-level description of the building block. In a circuit simulation tool (Fig. 2, bottom right), several of these building blocks are combined, in

Please cite this article in press as: M. Fiers, et al., Improving the design cycle for nanophotonic components, J. Comput. Sci. (2013), http://dx.doi.org/10.1016/j.jocs.2013.05.008