Technology and Design Flow Automation for Photonic Integrated Circuits

D. Pustakhod, W. Yao, F. Lemaitre, V. Dolores-Calzadilla, M. Trajkovic, X. Leijtens, and K. Williams

Eindhoven University of Technology, Photonic Integration Technology Center, 5612 AZ, Eindhoven, the Netherlands

With the increased complexity of Photonic Integrated Circuits, the design tools and workflow play an important role in obtaining the correct chip designs. In this paper we present a design flow which includes technology, Process Design Kit definition, chip design, automated reticle generation, and wafer-level design assembly.

Introduction

Photonic Integrated Circuits (PICs) are used in an increasing number of applications [1, 2]. Multi-project wafer (MPW) runs have stimulated the development of new PIC designs with new application-driven functionalities. A broad PIC design ecosystem which includes numerous design and simulation tools has been developed to support this [3–5]. When designing for an MPW run, a user (circuit designer) commonly uses a Process Design Kit (PDK), which abstracts the complexity of the manufacturing process into a set of standardized building blocks to be used in the circuit. The PDK contains in addition information required for device/circuit simulations and design rules, eliminating the need for the user to deal with the fabrication process.

The concept of PDK is crucial as it enables the separation of the technology development, the building block development, and the circuit design. However, when developing application-specific circuits on a custom fabrication process, all three aforementioned design stages need to be effectively interconnected in order to lead to successful and error-free development. Therefore an automated design tool chain is called for to enable seamless propagation of changes done on any stage of the design process all the way to the final fabrication masks.

Overview

In this paper, we present an open-source tool chain and its corresponding work flow which cover all development stages. The workflow consists of three main stages: PDK development, circuit design, and mask assembly. The PDK development includes the definition of fabrication layers and their tolerances, cross-sections composition, basic and complex building blocks. The PDK is then used for creating custom application-specific chip designs on a higher abstraction level, which makes it accessible to users not involved in the development of the PDK. After finalization of the chip designs, they are verified and processed during the mask assembly process. The general concepts of such a comprehensive approach were reviewed in [4]. We present a concrete implementation of the design process using available open-source packages, which generates the PDK, the designs, and the mask set with a single script in less than 15 minutes on a standard laptop hardware.

In addition, we show examples of useful tooling such as device cross-section schematic generation and Design Rule Checking (DRC), which enable the detection of possible mistakes early in the design flow.

Process Design Kit development

Essentially a PDK provides designers with a set of building blocks (BB) and interconnects which are used for circuit design. However there are several hidden concepts which enable easy and error-proof conversion of the building blocks into the fabrication layers. At the bottom of the pyramid is the *technology process* defined as a sequence of fabrication steps. The *lithography masks* used in the fabrication are the result of the design process, i.e. each building block is in the end translated in one mask or the combination of several masks. These masks are generated from *layout mask* layers which contain the logical structure of the circuit and a lot of auxiliary information enabling the visual design checking, such as types of waveguides and metal tracks, positions of the BB pins, etc. The concept of *cross-sections* reduces the need to remember all the details about the layout masks by grouping them according to specific functionalities (waveguides, metal tracks, etc.) together with their dimensions. Finally, the *building blocks* and *interconnects* are the two types of components available to a designer using the PDK to implement a photonic circuit. In their implementation, both Interconnects and Building Blocks are drawn with the geometrical primitives in cross-sections.

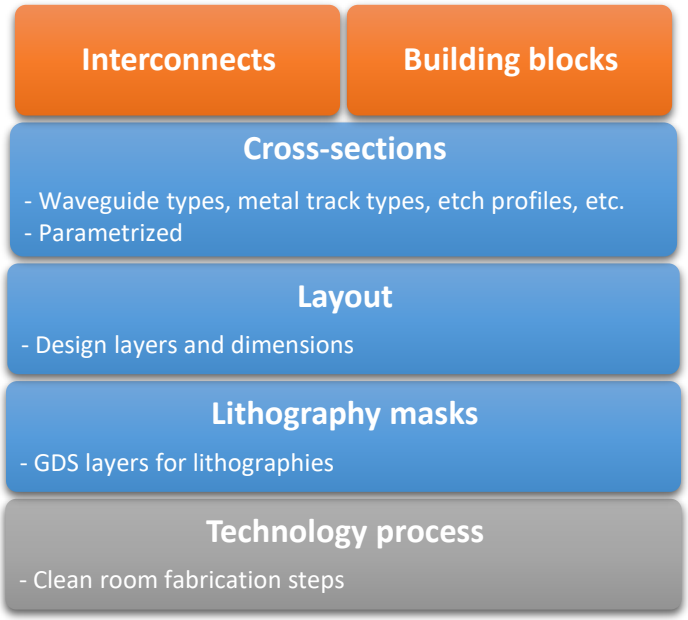

Figure 1. Levels of PDK. The bottom level, Technology, defines the steps to be followed during the device fabrication in the clean room. The top levels, Interconnects and Building Blocks, are available to the circuit designer via the PDK. The intermediate levels are hidden inside the PDK.

The Nazca Design python package [6] used for PDK implementation supports all the abstraction layers described above, and therefore implementing such a structure can be done with a single software engine.

For visualizing the building blocks and the circuits we are using the KLayout mask viewing software [7]. It has a broad set of useful functions and additional packages, which enable automated verification of the designs. The Design Rule Checking and the cross-section engines are two noteworthy tools. The first one allows the creation of scripts which automatically check for any violation of fabrication rules, and thus give quick feedback on the compliance of the circuit design to the PDK and design rules. The second tool (klayout_pyxs [8]) provides the possibility to visualize the result of the fabrication process for a given set of masks (see example in Fig 2.). This functionality is essential when developing the technology process and transferring it into the fabrication masks and the cross-sections of the PDK.

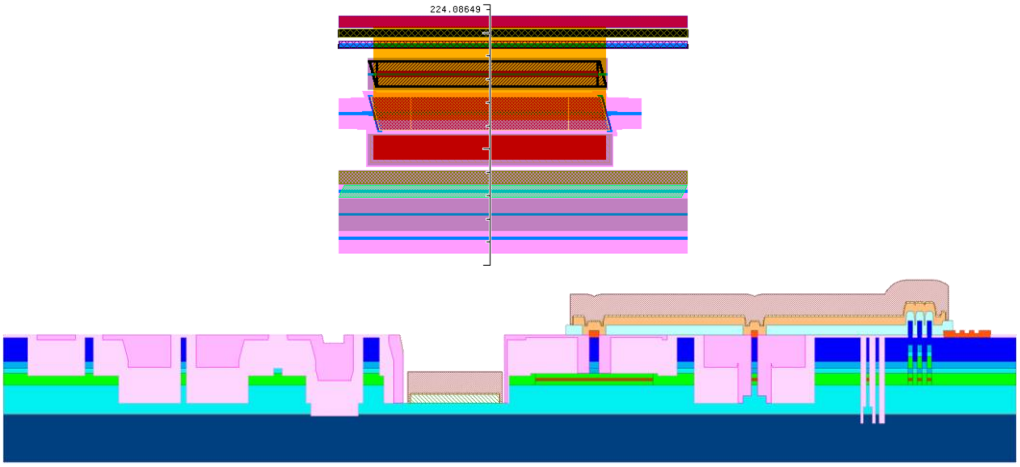

Figure 2. Top-view of the mask design (top) and the automatically generated cross-section view (bottom). Top: colors represent different layers, bottom: colors represent different materials.

Cell design

Once the PDK is developed, it is distributed in the form of a python package to the designers (users) of the cells for the fabrication run. The designer has access to high-level components such as BBs and interconnects defined in the PDK. These components are used to construct the circuit design following a standardized cell template, also made available with the PDK. In an example implementation of the workflow, we utilized nine cells of 4.6×4 mm in size. After the cell designs are finalized (tape-out), they are the input of the mask assembly step, which is the next stage of the workflow.

Mask assembly

The mask assembly process is organized in four principal steps shown in Fig. 3. After receiving the user cell designs, they are assembled into a Supercell onto a 3×3 grid (Fig. 2a). When making a supercell, we use only the fabrication layers (18 in total) of the GDS files provided by the users. In addition to the user designs, the supercell also contains process control modules (PCMs) that characterize various fabrication steps.

In the second step (Fig. 3b), we generate reticles for each lithography by taking the corresponding fabrication layer of the supercell and placing it on a separate mask. The reticle includes the reticle border, which can be seen in purple in Fig. 3b. The size of the reticle inner field and the border are determined by the machine types used in fabrication, thus limiting the size of the cells and the supercell grid size. The total number of reticles in our process is 15.

When the reticle designs are ready, the jobs for the machines are programmed. Inside the job, the location of the image on the reticle is specified together with the locations where this image should be projected on the wafer. We use the third and the forth steps (Fig. 3c and 3d) to check the correctness of the jobs. First we take the corresponding images from the reticles and reconstruct the macro-cells from these images. These macro-cells represent the set of lithographies applied to a specific region of the wafer. After the macro-cells are assembled, they are put onto the different positions on the future wafer, and in this way we see the full layout of the images on the wafer. This layout can be used for fabrication of the wafer-size masks for the contact lithography if needed.

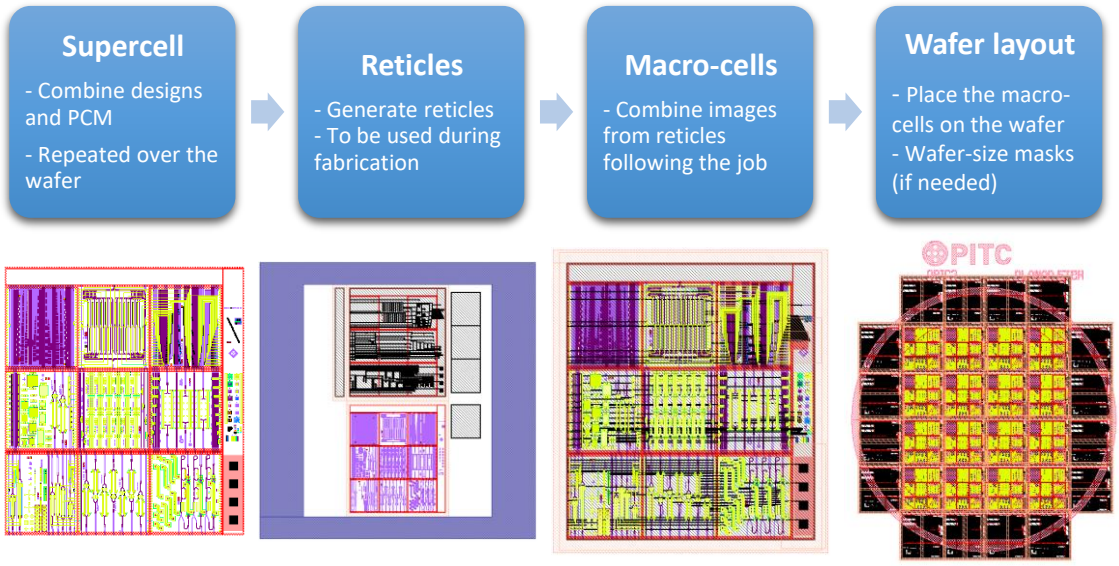

Figure 3. Steps of mask assembly process. Four steps of the process are shown with the examples of the masks generated on each step. The masks are shown not to scale.

The mask assembly uses another python package, klayout [9]. It provides a comprehensive and reliable API for accessing GDS layers and cells including full cell hierarchies.

Conclusions

We have presented an automated PIC design and mask assembly workflow based on tools centered on the open-source Nazca Design package and the KLayout software. The former is used for PDK creation (layers, building blocks, circuit design), and the latter is used for mask assembly and DRC. The complete generation of the PDK, the designs, and the mask set uses a single script and takes less than 15 minutes on a standard laptop hardware. An additional python package klayout_pyxs is used for cross-section generation. Due to its open-source nature the tool chain can be royalty-free which makes it an attractive photonic design automation (PDA) solution for universities and small companies.

Acknowledgements

The authors would like to acknowledge the Stimulus OPZuid program through the Open Innovation Photonic ICs project (PROJ-00315) for supporting this work.

References

[1] S. Arafin and L. Coldren, “Advanced InP Photonic Integrated Circuits for Communication and Sensing,” IEEE Journal of Selected Topics in Quantum Electronics, vol. 24, no. 1, pp. 1–12, 2018.
[2] S. Latkowski and D. Lenstra, “Lasers in InP generic photonic integration technology platforms,” Advanced Optical Technologies 4, 179–188, 2015.
[3] M. Smit et al. “An introduction to InP-based generic integration technology,” Semicond. Sci. Technol. 29, 083001, 2014.
[4] W. Bogaerts and L. Chrostowski, “Silicon Photonics Circuit Design: Methods, Tools and Challenges,” Laser & Photonics Reviews, 12, 1700237, 2018.
[5] “JePPIX Roadmap”, http://www.jeppix.eu/vision.
[6] “Nazca Design”, https://nazca-design.org.
[7] “KLayout Layout Viewer And Editor”, https://www.klayout.de.
[8] “Python port of xsection script for KLayout,” https://github.com/dimapu/klayout_pyxs.
[9] “KLayout standalone Python package,” https://pypi.org/project/klayout/.