Teaching photonic integrated circuits with Jupyter notebooks: design, simulation, fabrication

Wim Bogaerts

Ghent university - IMEC. Photonics Research Group. Department of Information Technology, Ghent, Belgium
Center for Nano- and Biophotonics (NB Photonics), Ghent, Belgium.

ABSTRACT

At Ghent University, we have built a course curriculum on integrated photonics, and in particular silicon photonics, based on interactive Jupyter Notebooks. This has been used in short workshops, specialization courses at PhD level, as well as the M.Sc. Photonics Engineering program at Ghent University and the Free University of Brussels. The course material teaches the concepts of on-chip waveguides, basic building blocks, circuits, the design process, fabrication and measurements. The Jupyter notebook environment provides an interface where static didactic content (text, figures, movies, formulas) is mixed with Python code that the user can modify and execute, and interactive plots and widgets to explore the effect of changes in circuits or components. The Python environment supplies a host of scientific and engineering libraries, while the photonic capabilities are based on IPKISS, a commercial design framework for photonic integrated circuits by Luceda Photonics. The IPKISS framework allows scripting of layout and simulation directly from the Jupyter notebooks, so the teaching modules contain live circuit simulation, as well as integration with electromagnetic solvers. Because this is a complete design framework, students can also use it to tape out a small chip design which is fabricated through a rapid prototyping service and then measured, allowing the students to validate the actual performance of their design against the original simulation. The scripting in Jupyter notebooks also provides a self-documenting design flow, and the use of an established design tool guarantees that the acquired skills can be transferred to larger, real-world design projects.

Keywords: Jupyter Notebooks, Photonic Integrated Circuit, Photonic Circuit Design, Simulation, Course Material

1. INTRODUCTION

Photonic integrated circuits (PIC) is the umbrella term for technologies which integrated many optical functions on the surface of a chip. This comprises multiple material systems based on polymers, metals, dielectrics and semiconductors.[1,2] As with electronic integrated circuits, PICs allow miniaturization of complex functionality, resulting in a smaller footprint, lower power consumption, stability and robustness, and above all functionality that cannot be easily realized with discrete building blocks. PICs have been a well-established technology in fiber-optic communication systems, and are now finding their way in other applications for sensors, spectroscopy, LiDAR, quantum optics etc.

The adoption of PICs has rapidly grown since the advent of *silicon photonics*. This PIC technology makes use of silicon as a base material system, leveraging the technology developments of silicon electronics to enable scaling to large-volume manufacturing. Add to that the fact that the silicon-on-insulator (SOI) material system allows very tight confinement of light, and therefore the integration of a much larger number of optical functions into a single circuit. It is this scaling that is pushing a paradigm shift in the world of PICs.

Up till the last decade, photonic integrated *circuits* contained only a few functional blocks within a circuit, connected by optical waveguides. Design efforts were large focused on geometrical design, optimizing the optical geometries to manipulate the light at level of the electromagnetic fields. But with the capability of larger-scale circuits, the design process is gradually shifting to the circuit level. No longer focusing on geometries, circuit

Further author information: E-mail: wim.bogaerts@ugent.be

design makes abstraction of the actual electromagnetic fields, and describes the behaviour of the light on the chip through signals that are being routed between functional building blocks. As in electronics, circuit abstraction allows the design of much more complex functionality. However, this change in design paradigm is only slowly being adopted in photonics.

1.1 Deployment

At the same time, the rapid adoption of silicon photonics (and PIC technology in general) has led to a shortage of people with design skills for photonic ICs. As a result, people are entering the field in need of basic learning and practical guidelines on how to start designing a chip. We see two types of education and training activities to address this need. On one hand, PIC-oriented summer schools introduce the basic principles behind PIC technology (either generic, or focused to s more narrow field like silicon photonics). These schools, taught by experts in the field, cover the technologies and applications, but usually do not include a strong hands-on component. On the other hand, design tool training courses get new users started on specific PIC design tools. As there are several commercial design tool vendors,[3–6] there is a significant competition in this space, and the many training sessions are often focused on the tools of a single vendor, and the part of the design workflow that is supported by those tools. As most design software is based on a graphical user interface (GUI), exercises often require to mimic the actions of an instructor or a screen recording, or follow a step-by-step printed instruction sheet.

To provide an alternative learning path, we have developed at Ghent University a hands-on course platform to teach the principles of design of photonic integrated circuits. Using Jupyter notebooks, the students are presented with integrated material that combines instructions and theoretical background with an interactive scripting environment that supports all the steps in a photonic integrated circuit design flow. By integrating the design tool with the explanations, distractions are eliminated, allowing the student to focus on the subject at hand. The result is an environment where students can go through the entire design flow of a photonic circuit, send off the design for fabrication, and come back to analyze the measurement data of the fabricated circuit.

In the rest of this paper, we will first elaborate on the teaching objectives, and in particular the design flow we wish our students to master. We then discuss the technical platform and the tools embedded in it, and how it lends itself to a variety of courses.

2. TEACHING OBJECTIVES

We position our course material between courses on the fundamental technologies (as often taught in short courses or summer schools), and vendor-specific trainings in design software. The objective is that students understand

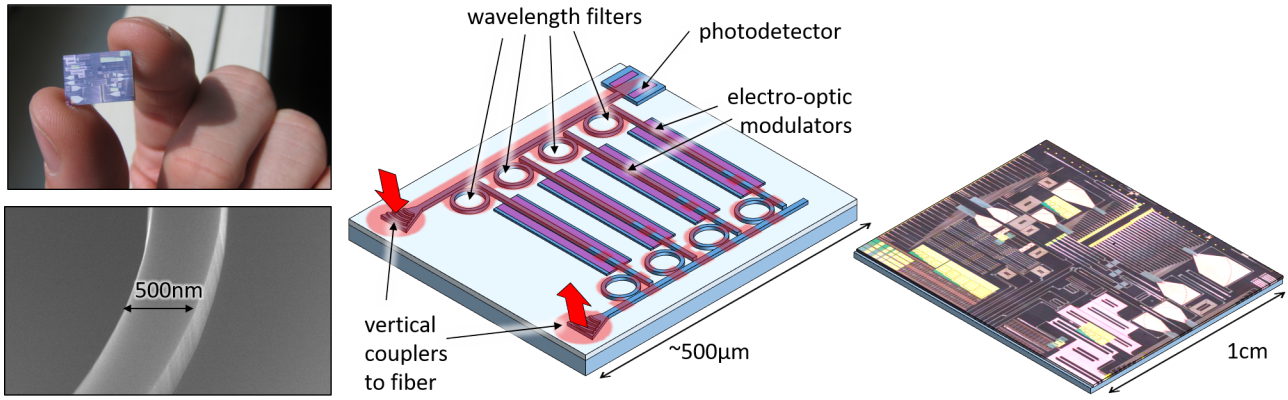


Figure 1. Silicon photonic integrated circuits (PIC). Silicon photonic chips can pack a large number of optical elements on a chip because the high index contrast allows confinement in submicrometer waveguides. By connecting functional elements such as electro-optic modulators, wavelength filters and photodetectors together in a circuit, complex functionality can be realized.

the fundamental steps in a photonic design flow (irrespective of the software tool that they intend to use), can make the distinction between component design and circuit design, and are aware of the many pitfalls in the process of translating a design idea into a working circuit. To realize that, we developed interactive teaching content for every step in the entire design flow, up to the point that the students can design a photonic circuit and have it fabricated and characterized.

The design flow for photonic integrated circuits is described in detail in,[7] and illustrated in Fig. 3. It bears a strong similarity with the design of electronic circuits. The first step is translating a functional idea into a schematic circuit, which consists of abstract building blocks connected with signal lines. These building blocks can be optical, electrical or a electro-optic, such as lasers, modulators or photodetectors. The optical signal lines can either represent direct connections between components, or optical waveguide connections. In photonic circuit design, it is important to make a distinction between waveguides that are used to 'just connect' two optical building blocks, or waveguides that have a deliberate optical function as a delay line. Because optical signals are coherent and carry not just an amplitude but also a phase, two or more optical signals can interfere constructively or destructively depending on their relative phase. Waveguide lines introduce a phase delay that can give rise to such interferences, which will be dependent on the wavelength of the light. Therefore, one of the important steps in the design process it to learn the distinction between waveguides that induce a deliberate delay (e.g. to implement a wavelength filter), and waveguide that merely serve as an interconnect 'wire'. Knowing where to use which in a circuit is an essential skill for a photonic circuit designer.

Simulation of photonic circuits can be done both in time domain and frequency (or wavelength) domain. Time-domain simulation for photonic circuits is similar to electronic simulations, with the distinction that photonic signals are usually modelled as a complex modulation (amplitude and phase) on a carrier wavelength. This is needed because the optical waves oscillate at frequencies around 200 THz, while the information carried on the waves usually covers a bandwidth of about 100 GHz. Simulating the full waveform requires a very small time step and does not generate any additional information. The photonic circuit simulations usually require a dedicated simulator based on a scattered waves formalism,[8,9] but there have also been demonstrations of photonic circuit simulations implemented in existing electronics circuit simulators.[10,11]

It is also quite common to simulate photonic circuits, and especially the linear passive components in the circuit, such as interferometric wavelength filter circuits, in the frequency domain. This generates a scatter matrix (S-matrix) of the circuits, with the complex coupling coefficients between all input-output ports. Often, the critical functionality of a photonic circuit is defined in the passive circuit, and therefore these simulations are essential to optimize a circuit design. The advantage of frequency domain simulations is that generally all the responses between the optical ports are obtained simultaneously. As the technique relies on the linearity of the circuit, all relations can be deduced through superposition. Time-domain simulations, on the other hand,

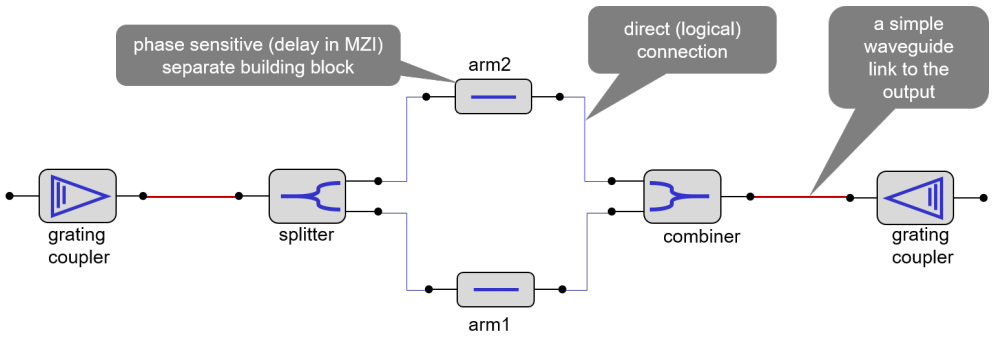


Figure 2. the different roles of waveguides in a photonic integrated circuit. When used in an interferometric structure, the optical phase delay of the waveguide is critical. Therefore, it is important to represent the waveguide as a dedicated component, directly connected to the splitter/combiner. On the other hand, for just connecting this interferometer to the input/output structures, the phase delay in the waveguide is not critical. In such cases it is convenient to represent the waveguide as a simple connection in the schematic.

can also model the nonlinear and time-varying behavior of a circuit, but require a separate simulation for every combination of stimuli.

Translating the schematic into a layout requires placement, routing and selecting layout parameters for the individual building blocks. The dual function of waveguides in a circuit also shows that it is not straightforward to decouple the abstract schematic from the physical layout of the circuit: actual waveguide lengths have a real impact on the circuit function, and should therefore also be considered ad the schematic level. Therefore, a design flow which integrates the layout and schematic functionality in a single tool provides a genuine advantage, allowing the designer to go back and forth between the schematic and the layout.

The resulting layout should then be verified on different levels. First, a design rule check (DRC) is performed to ascertain that the layout does not contain features that cannot be fabricated, such as sharp corners or narrow lines that cannot be resolved by the lithography process.[12] In a next step, the layout connectivity is analyzed, and compared to the original schematic. Here, again, using a tool that tightly integrates both aspects of the design, speeds up the design process. The final level to evaluate the circuit design is that of variability. Photonic waveguides, and especially high-contrast silicon waveguides, are extremely sensitive to nanometer-scale geometry variations. Therefore, it is important to assess how the circuit will perform in the presence of stochastic fluctuations in the fabrication parameters. Statistical simulations which take into account the layout-dependent fluctuations, and can help design the circuit to be tolerant to fabrication deviations, is becoming an essential step in the photonic circuit design flow.[13]

The result of a design flow is a chip that can be fabricated. For this we were inspired by silicon photonics courses organized by prof. Lukas Chrostowski at the University of British Columbia.[14] The students can use the learning environment to actually design their own circuit that is then fabricated and tested. For the fabrication, we make use of diverse channels, depending on the type of course, the size and complexity of the design, and the overall timeline. For small circuits, the use of a rapid prototyping service using e-beam technology is by far the most attractive. We have already made use of the services of Australian Silicon Photonics and the e-beam multi-project wafer (MPW) runs organized by the Si-EPIC program.[15] These services offer design-to-chip turn-around times of a few weeks. Alternatively, more complex circuits can be taped out on MPW runs by full-scale manufacturing services such as Europractice,[16] which have a turn-around time of a few months. It is even possible to use the design kits of different fabs, and have the same circuit fabricated in different places, to compare the relative performance.

The final step in the process is characterization of the chips. We measure the circuit response using an automated optical probe station which records the wavelength-dependent fiber-to-fiber transmission between the

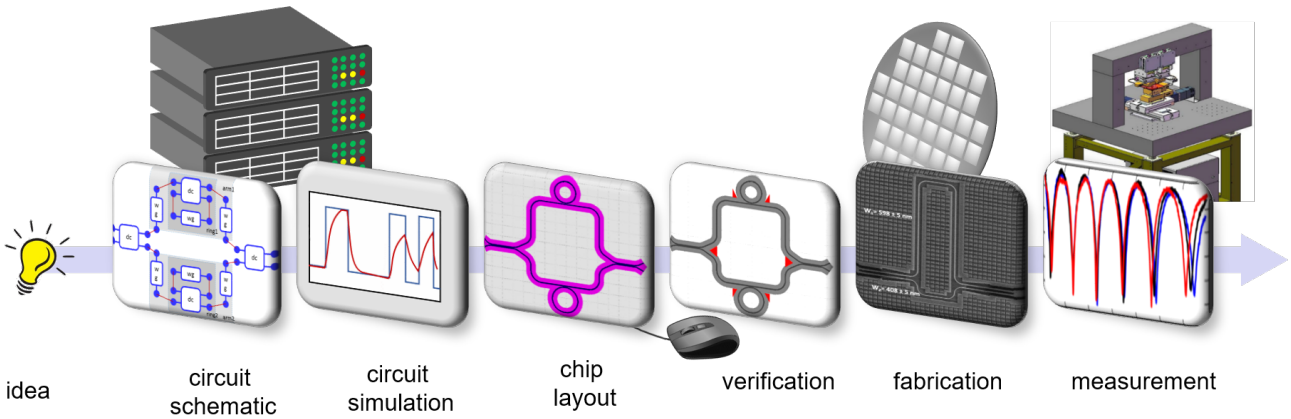


Figure 3. Design flow for photonic integrated circuits. First, the functional idea is captured in an abstract schematic, which is refined through circuit simulation. This schematic is translated into a layout, and further optimized taking into account the specific effects of the layout. After verifying whether the layout can be fabricated and corresponds to the original schematic, it is fabricated and tested.

input and output ports. This can be done using individual fibers, or fiber arrays. The resulting data consists of optical power transmissions, which the students need to analyze for performance, variations, and other circuit parameters.

3. THE TECHNICAL PLATFORM

3.1 Jupyter notebooks

When we set out to build a hands-on teaching platform for PIC design, we took into account several considerations. First of all, we needed a teaching workflow that could keep students focused on the material at hand, and where they could work at their own pace, and personally experience the result of each action in the design flow. The workflow should be reproducible, and not too dependent on the ecat state of the student's computer and software configurations. It should support all the essential steps in the design flow, and allow us to extend it with new functionality. We wanted to use it in different settings, including remote and online education.

This brought us to the Jupyter notebook environment.[17] Originally constructed around the interactive IPython kernel, it gives the user a Python (and now also other languages) scripting environment inside a standard web browser. It can be run both locally or remotely, requiring not special installations on the user's computer.

But the beauty of Jupyter notebooks is that is allows the combination of textbook material, LaTeX formulas, figures and videos with interactive Python code that can generate interactive data plots and visualizations. This makes it possible to combine the background information and step-by-step instructions with the actual design environment, giving the student a single view on all the information that is essential to a particular learning module or exercise. the notebooks allow for a linear learning flow that takes the student through subsequent steps, but also permits back-and-forth navigation and experimentation in the same environment.

The core language is Python. While Jupyter notebooks have expanded to a multitude of languages, we find Python to be ideally suited to our purpose. It has become a well-established language in engineering and

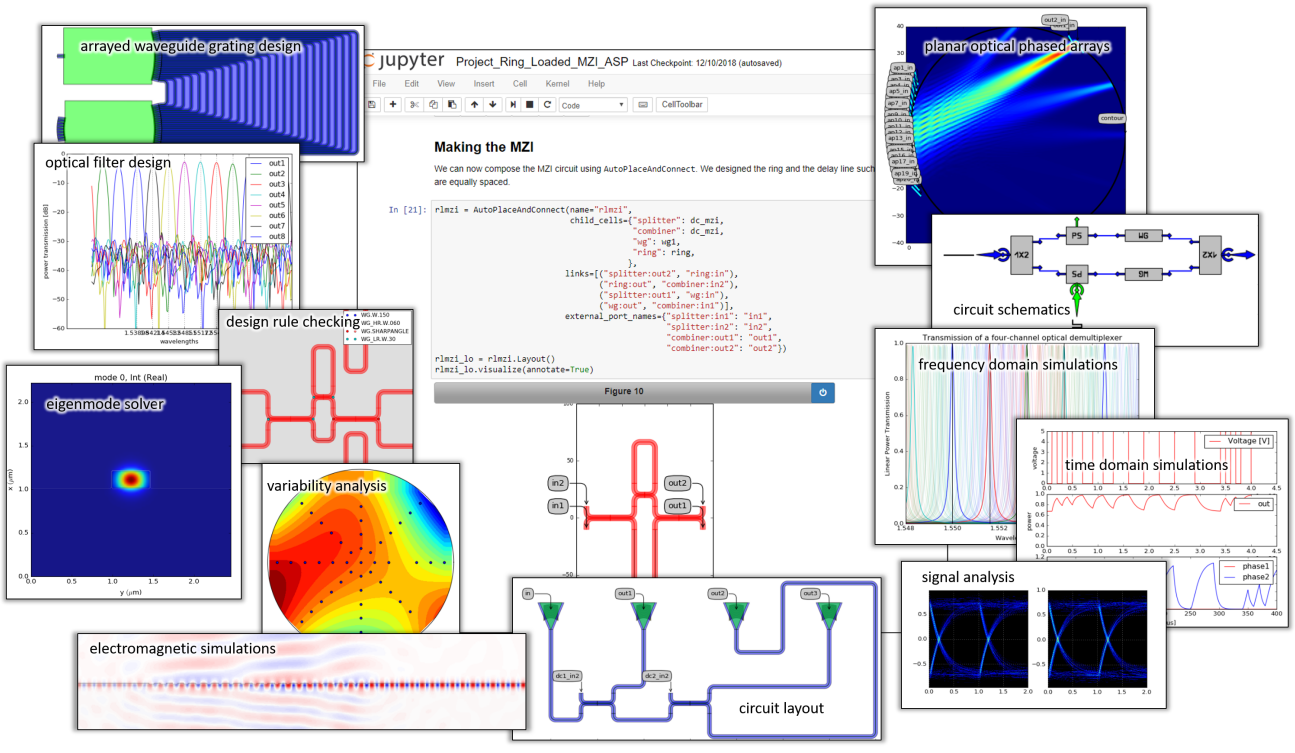


Figure 4. Example fragments of Jupyter notebooks for teaching PIC design concepts. All elements of the entire design flow, from schematic capture to verification and data analysis, can be executed within the same environment.

education (at Ghent University it is the language of choice to teach programming to engineering students). It is concise, human readable, and supported by a wide community and a host of powerful scientific and engineering libraries.

3.2 IPKISS

Within these Jupyter notebooks, we run the IPKISS photonics design framework. Originally developed as a photonic circuit design tool in our lab at Ghent University and imec,[18] IPKISS has evolved into a commercial PIC design environment by Luceda Photonics.[19] At its core it has a python scripting framework that allows the user to define parametric cells that contain design information on different views: layout, connectivity (netlist) or circuit models. This allows for a parametric design of complex hierarchical circuits. This is supported by a built-in powerful optical circuit simulator that supports both time-domain and frequency-domain simulations. It has various extensions for the design of optical filters, integrations with third-party simulation tools as well as interfaces to electronic-design automation tools. Extensions that we use for our course include a design toolbox for wavelength filters, including complex arrayed waveguide gratings, the REME eigenmode solver developed by RMIT University, and an interface to invoke Lumerical FDTD[20] and CST Studio[21] for electromagnetic simulations.

We have customized and extended the IPKISS framework with several functions specific to our course material. In some places we simplified the standard IPKISS design flow to make students focus on the problem at hand, while in other situations we removed certain automated shortcuts, making design choices more explicit to create awareness of their impact. We also wrote Python interfaces to third-party tools for verification and versioning of design data, to make sure that students could go through the entire design flow within the Jupyter notebook environment.

On top of the off-the-shelf design framework, we created our own Monte-Carlo analysis framework that allows the student to assess the impact of fabrication fluctuations on their design.[13] Figure 4 shows a collection of examples of different design and simulation results that are generated as the students progress through the course material.

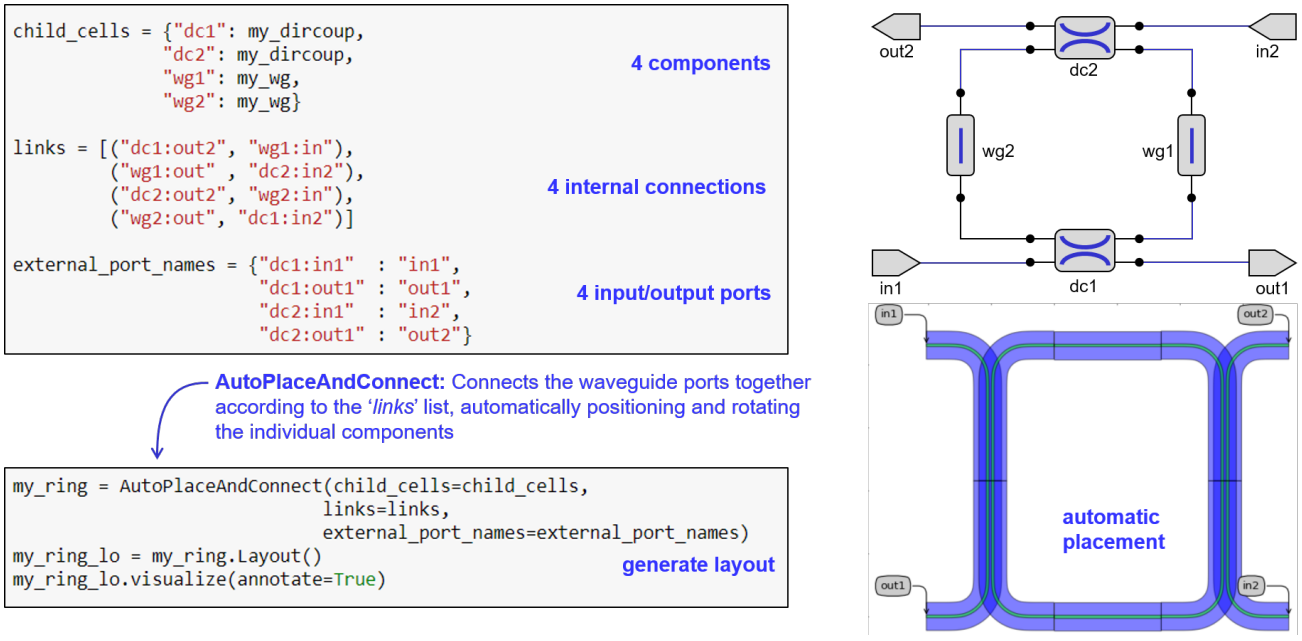


Figure 5. Connecting a circuit of a ring resonator with phase-sensitive waveguides. The waveguides are treated as components in the circuit, and connected with zero-length logical connections. The circuit can be fully specifies by a list of components, connections between ports, and the connections to the outside world.

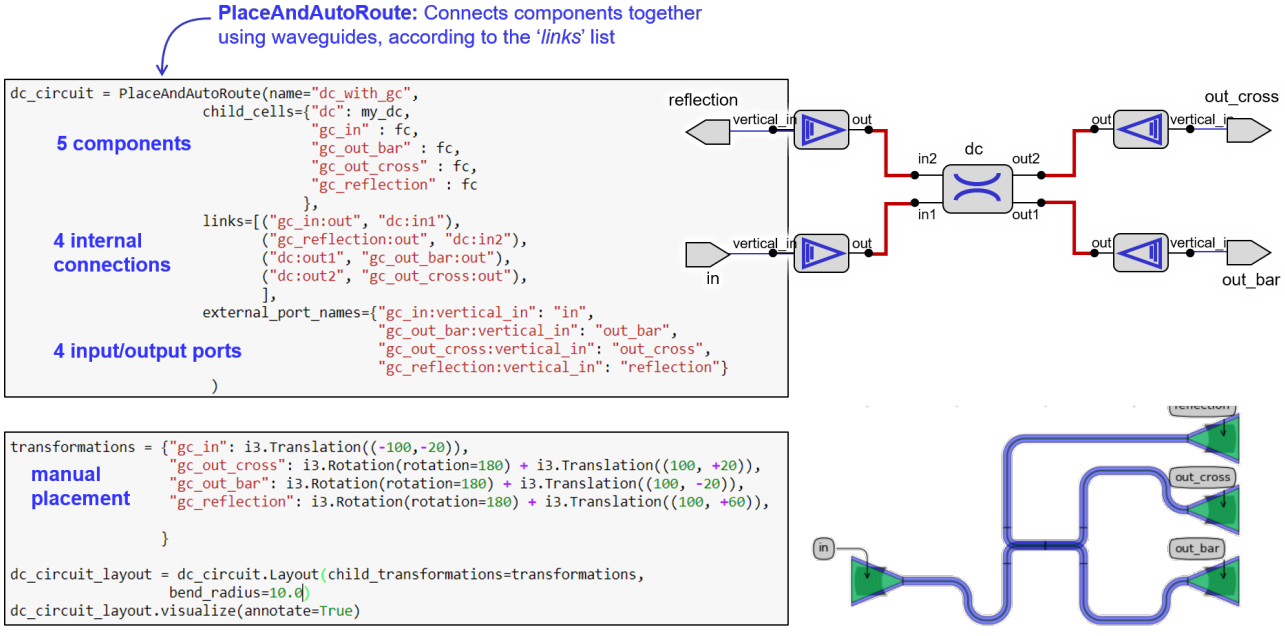
One of the key aspects that we try to convey in our course is the correct use of waveguides, as discussed earlier. We encourage the student to think critically about the use of their waveguides, and separate the circuits into parts where waveguides have a phase-sensitive function, and subcircuits where the waveguides just serve as interconnects. These two paradigms are depicted in Fig. 5 and 6, respectively. In both cases, a circuit can be described by a list of components, the connections between the ports of these components, and the connections to the outside world. For phase sensitive circuits, the components are connected together directly, and therefore the placement information is deduced automatically: all waveguide ports should snap in place. If this is not the case (e.g. because components were improperly dimensioned and they don;t fit together), the unmatched ports are highlighted in the layout. For circuits where the waveguides serves as simple connections (Fig. 6), the designer has to provide placement information, and the waveguides are then connected together by automatic or manual waveguide routes.

The IPKISS framework keeps all the aspect of the circuits and its components synchronized. So once a circuit is connected, it can also be simulated, taking into account the properties of the actual layout. For instance, in a circuit where the waveguides do not have a phase-sensitive function, the actual phase delay will still be taken into account in the simulation, so possible side-effects can be caught at the design stage.

3.3 Deployment

Rather than distributing the software to all students, we deployed a JupyterHub server. This removes a multitude of installation problems, as the only requirement for the students is to have a browser and an internet connection. All simulations and calculations are executed on the server, or dispatched to dedicated simulation machines to distribute the workload, as illustrated in Fig. 7. The course material is accessible on Windows, Linux and Mac environment. Within the server environment, the students have their own user account, and the state of the notebooks is maintained between sessions. The server environment also facilitates license management of the commercial software tools.

The course material itself is managed in a Mercurial (Hg) version control system, using a script to deploy it from the repository to the student accounts. Depending on the course, the learning modules are deployed all at




```
dc_circuit = PlaceAndAutoRoute(name="dc_with_gc",
                    child_cells={"dc": my_dc,
                                 "gc_in" : fc,
                                 "gc_out_bar" : fc,
                                 "gc_out_cross" : fc,
                                 "gc_reflection" : fc
                                },
                    links=[("gc_in:out", "dc:in1"),
                           ("gc_reflection:out", "dc:in2"),
                           ("dc:out1", "gc_out_bar:out"),
                           ("dc:out2", "gc_out_cross:out"),
                           ],
                    external_port_names={"gc_in:vertical_in": "in",
                                         "gc_out_bar:vertical_in": "out_bar",
                                         "gc_out_cross:vertical_in": "out_cross",
                                         "gc_reflection:vertical_in": "reflection"}
                     )
```

```
transformations = {"gc_in": i3.Translation((-100,-20)),
                   "gc_out_cross": i3.Rotation(rotation=180) + i3.Translation((100, +20)),
                   "gc_out_bar": i3.Rotation(rotation=180) + i3.Translation((100, -20)),
                   "gc_reflection": i3.Rotation(rotation=180) + i3.Translation((100, +60)),

                  }

dc_circuit_layout = dc_circuit.Layout(child_transformations=transformations,
                    bend_radius=10.0)
dc_circuit_layout.visualize(annotate=True)
```

Figure 6. Connecting a circuit with waveguide connections that are not phase-critical. The circuit is defined by a list of components, the connections between the ports and the outside world. In the layout, the designer is responsible for placing the components, and if necessary, overruling the automatic waveguide routes.

once, at fixed times, or based on the progress of the student. The version control system makes it easy to roll out new versions of the notebooks, or fix errors on very short notice.

4. CODE VS. GUI

By embedding the entire design flow into Jupyter notebooks, we force the student to implement his designs in code. This may seem far removed from the more common practice of designing circuits in a graphical user interface (GUI) where components are dragged onto a canvas and wired together with point-and-click operations. The preference of designing in a GUI or in code is a topic of much debate.

The graphical approach to circuit design is quite common, and with an intuitive, well-designed GUI novice designers can get started quite quickly. However, it is a workflow which is inherently limited in terms of scalability and reproducibility. It requires repetitive actions by the designer which are hard to register and/or reproduce, and in many cases it requires the designer to perform calculations offline and transfer the actual component or circuit parameters into the GUI. Automating the design of many variations of the same component or circuit also becomes tedious and error-prone.

From the point of view of teaching, using a GUI presents its own set of problems. As already mentioned, describing a workflow for students in a GUI often boils down to a screencast which they have to reproduce, or a set of instructions which have to be printed (unless the students have more than one screen at their disposal to keep the instructions and GUI visible at the same time. For the teacher, finding a problem in a student's project can be hard if the exact actions of the student in the GUI cannot be reproduced.

Designing in code removes most of these obstacles. Code is in most cases self-explanatory (given that it is properly written), and its execution is reproducible. Parameter variations and design changes are inherently possible, with constructs such as for-loops. In an environment as Jupyter notebooks, where the core can be combined with explanations on the same web page, the problem of combining instructions and the user interface is also addressed. The main obstacle with code-based circuit design is that not everyone is comfortable with a programming environment, and that a new programming language can present an almost insurmountable barrier. That is why the use of a standard language is so important, as it creates a community of support.

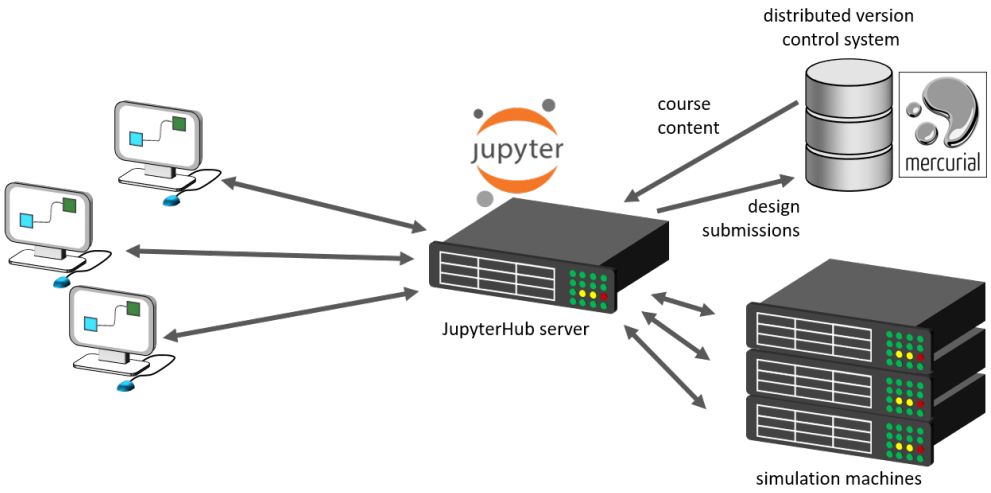


Figure 7. Basic breakdown of the computer infrastructure for the PIC course material. The course content is managed on a version control server (Mercurial), and deployed on a JupyterHub server with the IPKISS design software. Users interact with the server through a web browser, and all the design and simulation code is evaluated on the JupyterHub server, of dispatched to dedicated simulation machines.

4.1 Current use

Today, we use this course material in different ways. It is core to an elective course on Photonic Integrated Circuits in the M.Sc. program in Photonics Engineering at Ghent University. In this course, which runs over a semester 3 hours per week, students start very rapidly with the basics of circuit design, taping out their own design in the fourth week. While then diving into the deeper background an theory, the design is fabricated in time for them to analyse the measurements at the end of the semester.

A more specialized course, more focused on the principles of Photonic circuit design, is targeted at doctoral students, as well as professionals who want to get a deep understanding of the fundamentals of PIC design. This course is taught as a one-week intensive course, after which the students get 2 weeks to submit a circuit design for fabrication. When the chip is fabricated, the measurement data is provided to them through the Jupyter notebook environment.

Because the teaching material is modular, select parts are also used for focused half-day or full-day workshops, for instance at the annual Optical Fiber Communication conference (OFC).

Finally, the web-based infrastructure also makes it possible to make the learning material accessible to a wider audience. For this, Ghent University is currently developing a massive open online course (MOOC) on silicon photonics where these Jupyter notebooks will play an essential role.

5. SUMMARY

We have developed a learning platform to teach the principles of photonic integrated circuit design in a hands-on way, positioned between a theoretical background and a specialized software tool training. Using Jupyter notebooks, we integrate the textbook material with interactive Python code that covers the entire circuit design flow from schematic design over layout to verification and tape-out. This allows us to let students design their own circuit within the on-line course environment, and have it fabricated by a rapid prototyping service.

ACKNOWLEDGMENTS

Many thanks go to the people of Luceda Photonics, for technical support and making the IPKISS software available for educational purposes. Also thanks to Antonio Ribeiro, Lukas Van Iseghem and Umar Khan for technical support in keeping this learning platform operational. We are also grateful to prof. Lukas Chrostowski at the University of British Columbia for his pioneering example in photonic integrated circuit education, and making his SiEPIC chip fabrication platform available to my students. Likewise, many thanks to Arnan Mitchell, Thach Nguyen and Guanghui Ren of the RMIT university in Melbourne, for the fabrication services of Australian Silicon Photonics, and the use of the mode-solver REME in the course material.

REFERENCES

[1] Chen, X., Milosevic, M. M., Stankovic, S., Reynolds, S., Bucio, T. D., Li, K., Thomson, D. J., Gardes, F., and Reed, G. T., "The Emergence of Silicon Photonics as a Flexible Technology Platform," *Proceedings of the IEEE* **106**(12), 2101–2116 (2018).

[2] Nagarajan, R. and Kato, M., "InP photonic integrated circuits," *Selected Topics in ...* **16**(5), 1113–1125 (2010).

[3] Luceda Photonics, "IPKISS parametric design framework." http://www.lucedaphotonics.com.

[4] Lumerical Solutions, Inc., "InterConnect." http://www.lumerical.com/tcad-products/interconnect/.

[5] Lamant, G., Flueckiger, J., Villa, F., Farsaei, A., Wang, B., Stoffer, R., Wang, X., Pond, J., and Korthorst, T., "Schematic driven simulation and layout of complex photonic IC's," in [*IEEE International Conference on Group IV Photonics GFP*], **2016-Novem**, 92–93 (8 2016).

[6] Synopsys, "OptoDesigner Photonic Chip and Mask Layout."

[7] Bogaerts, W. and Chrostowski, L., "Silicon Photonics Circuit Design: Methods, Tools and Challenges," *Laser and Photonics Reviews* **1700237**, 1–29 (2018).

[8] Arellano, C., Mingaleev, S., Koltchanov, I., Richter, A., Pomplun, J., Burger, S., and Schmidt, F., "Efficient design of photonic integrated circuits (PICs) by combining device- and circuit- level simulation tools," in [*Proc. SPIE, Integrated Optics: Devices, Materials, and Technologies XVII*], **8627**, 862711, International Society for Optics and Photonics (2013).

[9] Fiers, M., Van Vaerenbergh, T., Caluwaerts, K., Vande Ginste, D., Schrauwen, B., Dambre, J., and Bienstman, P., "Time-domain and frequency-domain modeling of nonlinear optical components at the circuit-level using a node-based approach," *Journal of the Optical Society of America B* **29**(5), 896 (2012).

[10] Sorace-Agaskar, C., Leu, J., Watts, M. R., and Stojanovic, V., "Electro-optical co-simulation for integrated CMOS photonic circuits with VerilogA.," *Optics express* **23**, 27180–203 (10 2015).

[11] Martin, P., Gays, F., Grellier, E., Myko, A., and Menezo, S., "Modeling of silicon photonics devices with Verilog-A," in [*Proceedings of the International Conference on Microelectronics, ICM*], 209–212 (5 2014).

[12] Cao, R., Ferguson, J., Bakker, A., Korthorst, T., Arriordaz, A., and O'Connor, I., "Photonic designs with EDA approach: A novel methodology for curve design validation," in [*2015 IEEE Summer Topicals Meeting Series, SUM 2015*], 29–30, IEEE (2015).

[13] Bogaerts, W., Xing, Y., and Khan, M. U., "Layout-Aware Variability Analysis, Yield Prediction and Optimization in Photonic Integrated Circuits," *IEEE Journal of Selected Topics in Quantum Electronics* (2019).

[14] Chrostowski, L., Rouger, N., Deptuck, D., and Jaeger, N. A., "Silicon nanophotonics fabrication: An innovative graduate course," *ICT 2010: 2010 17th International Conference on Telecommunications* , 544–551 (2010).

[15] Chrostowski, L., "The Si-EPIC program."

[16] Europractice, "Silicon Photonics Multi Project Wafer Runs : Pricing." http://www.europractice-ic.com/SiPhotonics_pricing.php.

[17] Kluyver, T., Ragan-Kelley, B., Pérez, F., Granger, B. E., Bussonnier, M., Frederic, J., Kelley, K., Hamrick, J. B., Grout, J., Corlay, S., and others, "Jupyter Notebooks-a publishing format for reproducible computational workflows.," in [*ELPUB*], 87–90 (2016).

[18] Fiers, M., Lambert, E., Pathak, S., Maes, B., Bienstman, P., Bogaerts, W., and Dumon, P., "Improving the design cycle for nanophotonic components," *Journal of Computational Science* **4**(5), 313–324 (2013).

[19] The IPKISS Parametric Design Framework, "www.ipkiss.org."

[20] Lumerical Solutions, Inc., "{FDTD} Solutions." http://www.lumerical.com/tcad-products/fdtd/.

[21] Simulia, D. S., "CST Studio Suite."