Acknowledgements

I would like to give my most heartfelt thanks to the best friend one could ever ask for: Thomas Heuschling for his patience, friendship, guidance and all of the amazing moments we spent throughout our studies. I would also thank Prof. Wim Bogaerts and Prof. Dirk Stroobandt for their time, patience and trust in applying for an FWO proposal to extend this Master Thesis. And finally, I would like to thank Alexandre Bourbeillon for his help and advices for the creation of the grammar, parser and compiler of the PHÔS programming language.

Remark on the master's dissertation and the oral presentation

This master's dissertation is part of an exam. Any comments formulated by the assessment committee during the oral presentation of the master's dissertation are not included in this text.

Abstract

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim aeque doleamus animo, cum corpore dolemus, fieri.

Table of contents

Glossary

DSP	Digital Signal Processor	
FIR	Finite Impulse Response	4
FPGA	Field Programmable Gate Array	1, 3
FPPGA	Field Programmable Photonic Gate Array	3
IIR	Infinite Impulse Response	4
PHÔS	Photonic Hardware Description Language	
PIC	Photonic Integrated Circuit	7, 1, 2, 3, 4
PRG	Photonics Research Group	1
RF	Radio Frequency	1
SPICE	Simulation Program with Integrated Circuit Emphasis	1, 2
Verilog-A	Verilog for Analog, a continuous-time subset of Verilog-AMS	2
Verilog-AMS	Verilog for Analog and Mixed Signal	VI, 2

List of figures

List of tables

List of code fragments

1. Introduction

TODO

1.1 Motivation

This section serves as the motivation for the research and development of appropriate abstractions and tools for the design of photonic circuits, especially as it pertains to the programming of so-called Photonic FPGAs (Field Programmable Gate Array) [1] or Photonic Processors. As with all other types of circuit designs, such as digital electronic circuits, analog electronic circuits and RF circuits, appropriate abstractions can allow the designer and/or engineer to focus on the functionality of their design rather than the implementation. One may draw parallels between the abstraction levels used when designing circuits and the abstractions used when designing software. Most notably the distinction made in the software-engineering world between imperative and declarative programming. The former is concerned with the "how" of the program while the latter is focused on the "what" of the program [2].

At a sufficiently high level of abstraction, the designer is no longer focusing on the implementation details (imperative) of their design, but rather on the functionality and behavioural expectations of their design (declarative) [2]. In turn, this allows the designer to focus on what truly matters to them: the functionality of their design.

Currently, a lot of the work being done, on photonic circuits, is done at a very low level of abstraction, close to the component-level [3]. This leads to several issues for broader access to the fields of photonic circuit design. Firstly, it requires expertise and understanding of the photonic component, their physics and the, sometimes complex, relationship between all of their design parameters. Secondly, it requires a large amount of time and effort to design and simulate a photonic circuit. Physical simulation of photonic circuit is generally slow [3, 4], which has led to efforts to simulate photonic circuit using SPICE (Simulation Program with Integrated Circuit Emphasis) [5]. Finally, the design and implementation of photonic circuit is generally expensive, requiring taping-out of the design and working with a fab for fabrication. This increases the cost, but also increases the time to market for the product [6].

This therefore means, that there is a large interest in constructing new abstractions, method of simulation and design tools for photonic circuit design. Especially for rapid prototyping and iteration of photonic design. This is the reason that research in the field of programmable photonics, and especially recirculating programmable photonics has had growing interest in the past few years. This master's thesis has the singular purpose of finding ways in which the user can easily design their photonic circuit and program them onto those programmable PICs [6]. This interest has culminated in the work being done as part of the PRG (Photonics Research Group).

Additionally, photonic circuits are often not the only component in a system. They are often used in conjunction with other technologies, such as analog electronics, used in driving the photonic components, digital electronic, to provide control and feedback loops and RF (Radio Frequency) to benefit from the high bandwidth, high speed capabilities of photonics [7]. Therefore, it is of interest to the user to co-simulate [3, 8] their photonic circuits with the other components of their systems. This is a problem that is partly addressed using SPICE simulation [5]. However, especially with regards to digital

co-simulation, SPICE tools are often lacking, making the process difficult [9], relying instead on alternative such as Verilog-A (Verilog for Analog, a continuous-time subset of Verilog-AMS).

In this work, the beginning of a solution to these problems will be offered by introducing a new way of designing photonic circuit using code, a novel way of simulating these circuits and a complete workflow for the design and programming of circuit will be presented. Finally, an extension of the simulation paradigm will be introduced, allowing for the co-simulation of the designs with digital electronics, which could, in time, be extended to analog electronics as well.

1.1.a Research questions

The main goal of this work is to design a system to program photonic circuits, this entails the following:

1. How to express the user's intent?
    - What programming languages and paradigms are best suited?
    - What workflows are best suited?
    - How does the user test and verify their design?
2. How to translate that intent into a programmable PIC configuration?
    - What does a compiler need to do?
    - How to support future hardware platforms?
    - What are the unit operations that the hardware platform must support?

The remainder of this work will be structured following these questions until the formulation of thorough answers to them ; followed by a series of examples to demonstrate the potential use of the workflow. Finally, demonstrations based on the prototype of the aforementioned workflow will be presented showing the potential and the capabilities of the simulation system.

2.
Programmable photonics

As previously mentioned in Section 1.1, the primary goal of this thesis is to find which paradigms and languages are best suited for the programming of photonic FPGAs. However, before discussing these topics in detail, it is necessary to start discussing the basic of photonic processors. This chapter will therefore start by discussing what photonic processors are, what niche they fill and how they work. From this, the chapter will then move on to discuss the different types of photonic processors and how they differ from each other. Finally, this chapter will conclude with the first and most important assumption made in all subsequent design decisions.

In this document, the names Photonic FPGA and Photonic Processor are used interchangeably. They are both used to refer to the same thing, that being a programmable photonic device. The difference is that the former predates the latter in its use. Sometimes, they are also called FPPGA (Field Programmable Photonic Gate Array) [10].

2.1 Photonic processors

In essence, a photonic FPGA or photonic processor is the optical analogue to the traditional digital FPGA. It is composed of a certain number of gates connected using waveguides, which can be programmed to perform some functions [11]. However, whereas traditional FPGAs uses electrical current to carry information, photonic processor uses light contained within waveguide.

However, it is interesting to note that, just like traditional FPGAs, there are devices that are more general forms of programmable PIC (Photonic Integrated Circuit)[1] than others. As any PIC that has configurable elements could be considered a programmable PIC, it is reasonable to construct a hierarchy of programmability, where the most general device is the photonic processor, which is of interest for this document, going down to the simplest tunable structures.

Therefore, looking at Fig. 1, one can see that there are four large categories of PIC based on their programmability. The first ones (a) are not programmable at all, they require no tunable elements and are therefore the simplest. The second category (b) contains circuits that have tunable elements but fixed function, the tunable element could be a tunable coupler, modulator, phase shifter, etc. and allows the designed to tweak the properties of their circuit during use. The third kind of PIC is the feedforward architecture (c), which means that the light is expected to travel in a specific direction, it is composed of gates, generally containing tunable couplers and phase shifters. Additionally, external devices such as high speed modulators, amplifiers and other elements can be added. Finally, the most generic kind of programmable PIC is the recirculating mesh (d), which, while also composed of tunable couplers and phase shifters, allows the light to travel in either direction, allowing for more general circuits to be built as explored in Section 2.1.a.

(a) (b) (c) (d)

Fig. 1: A hierachy of programmable PICs (Photonic Integrated Circuit), starting at the non-programmable single function PIC (a), moving then to the tunable PIC (b), the feedfoward architecture (c) and finally to the photonic processor (d).

In this work, the focus will be on the fourth kind of tunability, the most generic. However, the work can also apply to photonic circuit design in general and is not limited to photonic processors. As will be discussed in Section 2.1.a, the recirculating mesh is the most general kind of programmable PIC, but also the most difficult to represent with a logic flow of operation due to the fact that the light can travel in either direction. Therefore, the following question may be asked:

At a sufficiently high level of abstraction, can a photonic processor be considered to be equivalent to a feedforward architecture?

This will be answered in Section 2.2.b.

2.1.a Feedforward and recirculating mesh

Both architecture rely on the same components [1], those being 2x2 tunable couplers, optical phase shifters and optical waveguides. These elements are combined in all-optical gates which can be tuned to achieve the user's intent. Additionally, to provide more functionality, the meshed gates can also be connected to other devices, such as high speed modulators, amplifiers, etc. [1, 11, 12]

The primary difference between a feedfoward architecture and a recirculating architecture, is the ability for the designer to make light travel both ways in one waveguide. As is known [13], in a waveguide light can travel in two direction with little to no interactions. This means that, without any additional waveguides or hardware complexity, a photonic circuit can be made to support two guiding modes, one in each direction. This property can be used for more efficient routing cite along with the creation of more structures.

As it has been shown[12], recirculating meshes offer the ability to create more advanced structures such as IIR (Infinite Impulse Response) elements, whereas feedforward architectures are limited to FIR (Finite Impulse Response) systems. This is due to inherent infinite impulse response of the ring resonator cell, while in a feedforward architecture, the Mach-Zehnder interferometers have a finite impulse response. But, not only does the recirculating mesh allow the creation of IIR cells, it still allows the designed to create FIR cells if needed.

2.1.b Potential use cases of photonic processors

2.1.c Embedding of photonic processor in a larger system

2.2 Circuit representation

2.2.a Bi-directional systems

2.2.b Feedforward approximation

It has been shown that, given a sufficient level of abstraction, any bi-directional photonic circuit can be represented by an equivalent, higher-level, feedforward circuit. This result is crucial for the formulation of the requirements for the programming interface of such a photonic processor. And is the basis on which the rest of this document is built.

2.3 Initial design requirements

2.3.a Interfacing

2.3.b Programming

2.3.c Reconfigurability

2.3.d Tunability

3.
Programming of photonic processors

3.1 Analysis of existing software ecosystems

3.1.a Compiler and runime

3.1.b Code editors

3.1.c Formatting

3.1.d Linting

3.1.e Testing

3.1.f Debugging

With the previous sections, it can be seen that creating a user-friendly ecosystem revolves around the creation of tools to aid in development. The compiler and language cannot be created in isolation, and the ecosystem as a whole has to be considered to achieve the broadest possible adoption.

3.2 Analysis of programming paradigms

3.2.a Imperative programming

3.2.b Functional programming

3.2.c Object-oriented programming

3.2.d Logic programming

3.2.e Dataflow programming

This section has outlined different approaches for the selection of a programming paradigm. However, it is clear, and will be further demonstrated in Section 6.1 that dataflow programming is the most suitable for the development circuits for photonic processors.

4.
Translation of intent

5.
The PHÔS programming language

6.
Examples of photonic circuit programming

6.1 Using traditional programming languages

7.
Extending PHÔS to generic circuit design

8.

Simulation in PHÔS

8.1 Co-simulation with digital electronic

8.2 Towards co-simulation with analog electronic

9.
Future work

10.
Conclusion

Bibliography

[1] W. Bogaerts, D. Pérez, et al., "Programmable photonic circuits," Nature, vol. 586, no. 7828, pp. 207–216, Oct. 2020, doi: 10.1038/s41586-020-2764-0. Accessed: Mar. 10, 2023. [Online]. Available: https://www.nature.com/articles/s41586-020-2764-0

[2] "Imperative programming: Overview of the oldest programming paradigm," 2020. Accessed: Mar. 27, 2023. [Online]. Available: https://www.ionos.com/digitalguide/websites/web-development/imperative-programming/

[3] W. Bogaerts, and L. Chrostowski, "Silicon Photonics Circuit Design: Methods, Tools and Challenges," Laser & Photon. Reviews, vol. 12, no. 4, p. 1700237, Apr. 2018, doi: 10.1002/lpor.201700237. Accessed: Mar. 27, 2023. [Online]. Available: https://onlinelibrary.wiley.com/doi/10.1002/lpor.201700237

[4] E. Alerstam, T. Svensson, and S. Andersson-Engels, "Parallel computing with graphics processing units for high-speed Monte Carlo simulation of photon migration," J. Biomed. Opt., vol. 13, no. 6, p. 60504, 2008, doi: 10.1117/1.3041496. Accessed: Mar. 27, 2023. [Online]. Available: http://biomedicaloptics.spiedigitallibrary.org/article.aspx?doi=10.1117/1.3041496

[5] Y. Ye, T. Ullrick, W. Bogaerts, T. Dhaene, and D. Spina, "SPICE-Compatible Equivalent Circuit Models for Accurate Time-Domain Simulations of Passive Photonic Integrated Circuits," J. Lightw. Technol., vol. 40, no. 24, pp. 7856–7868, Dec. 2022, doi: 10.1109/JLT.2022.3206818. Accessed: Mar. 10, 2023. [Online]. Available: https://ieeexplore.ieee.org/document/9893343/

[6] W. Bogaerts, and A. Rahim, "Programmable Photonics: An Opportunity for an Accessible Large-Volume PIC Ecosystem," IEEE J. Sel. Topics Quantum Electronics, vol. 26, no. 5, pp. 1–17, Sep. 2020, doi: 10.1109/JSTQE.2020.2982980. Accessed: Mar. 10, 2023. [Online]. Available: https://ieeexplore.ieee.org/document/9049105/

[7] D. Marpaung, J. Yao, and J. Capmany, "Integrated microwave photonics," Nature Photon., vol. 13, no. 2, pp. 80–90, Feb. 2019, doi: 10.1038/s41566-018-0310-5. Accessed: Mar. 10, 2023. [Online]. Available: https://www.nature.com/articles/s41566-018-0310-5

[8] C. Sorace-Agaskar, J. Leu, M. R. Watts, and V. Stojanovic, "Electro-optical co-simulation for integrated CMOS photonic circuits with VerilogA," Opt. Electron., vol. 23, no. 21, p. 27180, Oct. 2015, doi: 10.1364/OE.23.027180. Accessed: Mar. 27, 2023. [Online]. Available: https://opg.optica.org/abstract.cfm?URI=oe-23-21-27180

[9] A. M. Smith, J. Mayo, et al., "Digital/analog cosimulation using cocotb and xyce.," , 2018, doi: 10.2172/1488489. [Online]. Available: https://www.osti.gov/biblio/1488489

[10] D. Pérez-López, A. López, P. DasMahapatra, and J. Capmany, "Multipurpose self-configuration of programmable photonic circuits," Nature Commun., vol. 11, no. 1, p. 6359, Dec. 2020, doi: 10.1038/s41467-020-19608-w. Accessed: Mar. 10, 2023. [Online]. Available: https://www.nature.com/articles/s41467-020-19608-w

[11] J. Capmany, I. Gasulla, and D. Pérez, "The programmable processor," Nature Photon., vol. 10, no. 1, pp. 6–8, Jan. 2016, doi: 10.1038/nphoton.2015.254. Accessed: Mar. 10, 2023. [Online]. Available: http://www.nature.com/articles/nphoton.2015.254

[12] D. Perez, I. Gasulla, and J. Capmany, "Programmable Multifuctional Photonics ICs," arXiv, 2019. Accessed: Mar. 28, 2023. [Online]. Available: http://arxiv.org/abs/1903.04602

[13] A. Ghatak, and K. Thyagarajan, Introduction to Fiber Optics, Cambridge: Cambridge University Press, 1998. Accessed: Mar. 28, 2023. [Online]. Available: http://ebooks.cambridge.org/ref/id/CBO9781139174770