UNIVERSITAT POLITÈCNICA DE VALÈNCIA
DEPARTAMENT DE COMUNICACIONS

Multipurpose Programmable Integrated Photonics: Principles and Applications

Ph.D THESIS

Aitor López Hernández

Supervisors:

Daniel Pérez López

José Campany Francoy

Prometheus Das Mahapatra

Universitat Politècnica de València June 2023

Thesis Reviewers:

Francisco José Díaz Otero

Javier Fraile Peláez

Valerio Pruneri

Thesis Commitee:

Pascual Muñoz Muñoz

Guillermo Carpintero del Barrio

Francisco José Díaz Otero

A mí me importa muy poco
que un pájaro en la alameda
se pose de un árbol a otro.

Agradecimientos

Y aquí estamos. Después de tanto tiempo, en la recta final de una larga aventura en la que he tenido el placer de contar con muchas personas muy especiales a las que espero hacer justicia mediante estas modestas líneas.

La entrega de esta tesis cierra al fin una etapa de mi vida que empecé hace ya más de cinco años, cuando decidí mudarme a Valencia para trabajar en la nueva ERC Grant que había recibido José. Gracias a ti, José, no sólo he podido conocer un mundo que pensaba (de veras) que sólo existía en la ciencia ficción, sino que también he podido realizarme como profesional y como persona y descubrir una ciudad en la que me siento muy a gusto.

Pero, si la fama del trabajo de José fue la que me hizo decidirme por venir aquí y trabajar en la UPV, conocer a Dani fue sin duda lo que revolucionó mi camino durante estos últimos años. Siento que podría dedicar un capítulo entero de mi tesis sólo para alabarte y agradecerte por todo lo que he aprendido de ti. Me sentiré siempre muy afortunado por haber tenido la oportunidad de vivir esta experiencia contigo; por las largas, sesudas, instructivas y (¿por qué no decirlo?) divertidas discusiones que hemos tenido a todas horas, por tu continuo afán por impedir que me venciera el desánimo en los momentos más duros y por enseñarme siempre lo mejor que sabes. Has conseguido contagiarnos a todos la pasión, el esfuerzo y el sacrificio que le dedicas a tu trabajo, y es por ello que me conmueve y me enorgullece mucho contemplar todo lo que has conseguido en tan poco tiempo. Estoy convencido de que te esperan muchos más éxitos de aquí en adelante (que espero poder seguir de muy cerca) y te deseo la mejor de las suertes y mucha felicidad junto a tu encantadora familia.

Tengo mucho que agradecerte a ti también, Ana, por muchas razones. Durante todo este tiempo tú has sido lo más parecido que he tenido a una hermana mayor, no sólo preocupándote por que no hiciese demasiado el ridículo en el laboratorio, sino por haber estado también pendiente siempre de mí también fuera de la oficina y haberme permitido tener la experiencia de enseñar en una clase, algo que llevaba queriendo probar toda la vida. Considero que, si puede haber alguna cualidad tan valiosa en cualquier trabajo como ser una buena profesional, es sin duda ser una buena persona. Y puedo decir que tú cumples, de sobra, con ambos calificativos.

Erica, ha sido un enorme placer compartir este camino contigo. Siento que todavía me queda mucho por aprender de tu resiliencia, de tu organización y de tu perseverancia. No me cabe duda de que eres (y te convertirás en) una científica excepcional.

También quiero dar las gracias a todos los miembros del iTEAM y de Photonics Research Labs, por todos estos años llenos de experiencias. Prometheus y Luis, por todos los momentos divertidos que hemos vivido juntos. David y Amparo, por todos los quebraderos de cabeza (que no han sido pocos) que os habré causado cada vez que me he enfrentado a un trámite administrativo.

A la gente de iPronics, y en especial a su división de software. Habéis llevado a vuestro pequeño a cotas inimaginables hace tan solo tres años. Sois un equipo excepcional y muy inspirador. Gracias, Carlos y Paco, por todo el tiempo y la dedicación con los que habéis resuelto mis dudas cuando las he tenido. Sois todo un ejemplo a seguir para mí.

A todos mis amigos y familiares. Ha sido un camino largo, con muchas subidas y bajadas, y aunque las cosas no han salido siempre como yo he esperado, siempre habéis estado ahí para apoyarme cuando os he necesitado, incluso desde la distancia.

Patry. Dziękuję za wiarę we mnie nawet bardziej niż we własne siły. Za motywowanie i inspirację do dążenia zawsze do bycia najlepszą wersją siebie. Za Twoją miłość i bezwarunkowe wsparcie. Przy Tobie czuję, że jestem zdolny do wszystkiego. Wkrótce nadejdzie mój czas, by być z Tobą na Twojej obronie pracy dyplomowej.

Mamá. Hemos pasado a lo largo de toda una vida por un sinfín de recitales de piano, obras escolares de teatro, actos académicos varios… Pero creo que esta será mi última función. Gracias por todo tu esfuerzo y sacrificio para llevarme hasta aquí. Espero que la disfrutes.

Juanan. Me has enseñado a sumar y a restar. A mutiplicar y a dividir. A hacer raíces cuadradas y potencias. Funciones trigonométricas, derivadas e integrales. Me enseñaste a tener la mente abierta y a amar a la ciencia. A ser ambicioso y perseverante. Tampoco estaría aquí sin ti. Has sido el mejor padre que hubiese podido tener. Todavía no puedo creer que ya no estés aquí, con las ganas que tenías de estar en mi defensa, pero tengo el consuelo de que pudiste ver cómo acabé esta tesis. El resto del camino me tocará hacerlo solo, pero teniendo muy presente siempre todo lo que me has enseñado. Te prometo que voy a hacerte sentir orgulloso. Te quiero mucho.

¡Gracias, de todo corazón, a todos vosotros!

Abstract

In recent years, programmable integrated photonics (PIP) has evolved from a promising, new paradigm to deploy photonics to a larger scale to a solid, revolutionary reality, bringing up the attention of numerous research and industry players. Based on the same theoretical foundations than field-programmable gate arrays (FPGAs), this technology relies on common, two-dimensional integrated optical hardware configurations based on the interconnection of programmable unit cells (PUCs), which -by suitable programming of their phase actuators- can implement a variety of functionalities that can be elaborated for basic or more complex operation in many application fields, such as artificial intelligence, deep learning, quantum information systems, 5/6-G telecommunications, switching, data center interconnections, hardware acceleration and sensing, amongst others.

In this work, we will dedicate ourselves to explore several software capabilities of these processors under different chip designs. We explore different cutting-edge approaches based on computational optimization and graph theory to precisely control and configure these devices. One of these, **self-configuration,** deals with the automated synthesis of optical circuit configurations -even in presence of parasitic effects such as nonuniform losses, optical and electrical crosstalk- without any need for prior knowledge about hardware state. There are occasions, though, in which accessing to this information may be of use. **Self-calibration** and **self-characterization** tools allow us to perform a quick check to our photonic processor's status, allowing us to retrieve useful pieces of information such as the electrical current needed to supply to each phase actuator to change its corresponding PUC state arbitrarily or the insertion loss of every unit cell and optical interconnection surrounding the structure. These mechanisms not only allow us to quickly identify any malfunctioning PUCs or chip areas in our design, but also reveal another alternative to program photonic circuits in our design from current pre-sets.

These strategies constitute a gigantic step to unleash all the potential of these devices. They provide solutions to handle with hundreds of variables and simultaneously manage multiple configuration actions, one of the main limitations that prevent this technology to scale up and become disruptive in the years to come.

Resumen

En los últimos años, la fotónica integrada programable ha evolucionado desde considerarse un paradigma nuevo y prometedor para implementar la fotónica a una escala más amplia hacia convertirse una realidad sólida y revolucionaria, capturando la atención de numerosos grupos de investigación e industrias. Basada en el mismo fundamento teórico que las matrices de puertas lógicas programables en campo (o FPGAs, en inglés), esta tecnología se sustenta en la disposición bidimensional de bloques unitarios de lógica programable (en inglés: PUCs) que -mediante una programación adecuada de sus actuadores de fase- pueden implementar una gran variedad de funcionalidades que pueden ser elaboradas para operaciones básicas o más complejas en muchos campos de aplicación como la inteligencia artificial, el aprendizaje profundo, los sistemas de información cuántica, las telecomunicaciones 5/6-G, en redes de conmutación, formando interconexiones en centros de datos, en la aceleración de hardware o en sistemas de detección, entre otros.

En este trabajo, nos dedicaremos a explorar varias aplicaciones software de estos procesadores en diferentes diseños de chips. Exploraremos diferentes enfoques de vanguardia basados en la optimización computacional y la teoría de grafos para controlar y configurar con precisión estos dispositivos. Uno de estos enfoques, la **autoconfiguración**, consiste en la síntesis automática de circuitos ópticos -incluso en presencia de efectos parasitarios como distribuciones de pérdidas no uniformes a lo largo del diseño hardware, o bajo interferencias ópticas y eléctricas- sin conocimiento previo sobre el estado del dispositivo. Hay ocasiones, sin embargo, en las que el acceso a esta información puede ser útil. Las herramientas de **autocalibración** y **autocaracterización** nos permiten realizar una comprobación rápida del estado de nuestro procesador fotónico, lo que nos permite extraer información útil como la corriente eléctrica que suministrar a cada actuador de fase para cambiar el estado de su PUC correspondiente, o las pérdidas de inserción de cada unidad programable y de las interconexiones ópticas que rodean a la estructura. Estos mecanismos no sólo nos permiten identificar rápidamente cualquier PUC o región del chip defectuosa en nuestro diseño, sino que también revelan otra alternativa para programar circuitos fotónicos en nuestro diseño a partir de valores de corriente predefinidos.

Estas estrategias constituyen un paso significativo para aprovechar todo el potencial de estos dispositivos. Proporcionan soluciones para manejar cientos de variables y gestionar simultáneamente múltiples acciones de configuración, una de las principales limitaciones que impiden que esta tecnología se extienda y se convierta en disruptiva en los próximos años.

Resùm

En els darrers anys, la fotònica integrada programable ha evolucionat des de considerarse un paradigma nou i prometedor per implementar la fotònica a una escala més ampla cap a convertir-se en una realitat sòlida i revolucionària, capturant l'atenció de nombrosos grups d'investigació i indústries. Basada en el mateix fonament teòric que les matrius de portes lògiques programable en camp (o FPGAs, en anglès), aquesta tecnología es sustenta en la disposición bidimensional de blocs units lògics programables (en anglès: PUCs) que -mitjançant una programación adequada dels seus actuadors de fase- poden implementar una gran varietat de funcionalitats que poden ser elaborades per a operacions bàsiques o més complexes en molts camps d'aplicació com la intel·ligència artificial, l'aprenentatge profund, els sistemes d'informació quàntica, les telecomunicacions 5/6-G, en xarxes de comutació, formant interconnexions en centres de dades, en l'acceleració de hardware o en sistemes de detecció, entre d'altres.

En aquest treball, ens dedicarem a explorar diverses capatitats de programari d'aquests processadors en diferents dissenys de xips. Explorem diferents enfocaments de vanguardia basats en l'optimització computacional i la teoría de grafs per controlar i configurar amb precisió aquests dispositius. Un d'aquests enfocaments, l'**autoconfiguració**, tracta de la síntesi automática de circuits òptics -fins i tot en presencia d'efectes parasitaris com ara pèrdues no uniformes o crosstalk òptic i elèctric- sense cap coneixement previ sobre l'estat del dispositiu. Tanmateix, hi ha ocasions en les quals l'accés a aquesta información pot ser útil. Les eines d'**autocalibració** i **autocaracterització** ens permeten realizar una comprovació ràpida de l'estat del nostre procesador fotònic, el que ens permet obtener informació útil com la corrent eléctrica necessària per alimentar cada actuador de fase per canviar l'estat del seu PUC corresponent o la pèrdua d'inserció de cada unitat programable i de les interconnexions òptiques que envolten l'estructura. Aquests mecanisms no només ens permeten identificar ràpidament qualsevol PUC o área del xip defectuosa en el nostre disseny , sinó que també ens mostren una altra alternativa per programar circuits fotònics en el nostre disseny a partir de valors de corrent predefinits.

Aquestes estratègies constitueixen un pas gegant per a aprofitar tot el potencial d'aquests dispositius. Proporcionen solucions per a gestionar centenars de variables i alhora administrar múltiples accions de configuració, una de les principals limitacions que impideixen que aquesta tecnología esdevingui disruptiva en els pròxims anys

Contents

List of Figures

List of Tables

List of Acronyms

(P)IC: (Photonic) Integrated Circuit

AS(P)IC: Application-Specific (Photonic) Integrated Circuit

BFS: Breadth-First Search

BJT: Bipolar Junction Transistor

BUD: Basic Unit Delay

BUL: Basic Unit Length

CAGR: Compound Annual Growth Rate

CF: Cost Function

CMOS: Complementary Metal-Oxide-Semiconductor

CROW: Coupled-Resonator Optical Waveguides

DC: Directional Coupler

DFS: Depth-First Search

DRAM: Dynamic Random-Access Memory

DSP: Digital Signal Processing

EDA: Electronic Design Automation

EMC: Electromagnetic Management Coupling

ER: Extinction Ratio

FIR: Finite Impulse Reponse

FP(P)GA: Field-Programmable (Photonic) Gate Array

FSR: Free Spectral Range

GA: Genetic Algorithm

HPB(B): High-Performance Block(s)

IIR: Infinite Impulse Response

InP: Indium Phosphide

IoT: Internet of Things

MEDA: Multi-electronic Driving Array

MEMA: Multi-electronic Monitor Array

MOSFET: Metal-Oxide-Silicon Field-Effect Transistor

MUA: Monitoring Unit Array

MWP: Microwave Photonics

MZI: Mach-Zehnder Interferometer

NRE: Non-Recurring Engineering

O/E: Optoelectronic

ODL: Optical Delay Line

ORR: Optical Ring Resonator

PIP: Programmable Integrated Photonics

PSO: Particle Swarm Optimization

PUC: Programmable Unit Cell

RAMZI: Ring-Assisted Mach Zehnder Interferometer

RCA: Radio Corporation of America

RF: Radiofrequency

SCISSOR: Side-Coupled Integrated Spaced Sequence of Resonators

SiN: Silicon Nitride

SOI: Silicon On Insulator

TSMC: Taiwan Semiconductor Manufacturing Company

VLSI: Very Large Scale Integration

WDM: Wavelength Division Multiplexing

Chapter 1

Introduction and Thesis objectives

1.1 A (not so small) overview of semiconductor industry story

1.1.1 The birth of electronic integration

Dallas, September of 1958. After several exhausting and likely suffocating months of intense work at Texas Instruments Central Research Labs, a newly employed, 35-year-old Jack St. Clair Kilby found himself presenting to company's management the results of his fatidic "summer break".

Around that time, the so-called "tyranny of numbers" was a real warhorse for all computer engineers around the globe. Early computers were including an increasing number of components which were needed to be wired to every other and were typically strung and soldered by hand. In such way, an increase of performance (and hence of the number of required components) would require an excessive volume of wiring, eventually leading to construction and reliability problems.

Inspired by the quiet solitude provided by lab work in summertime, Kilby's approach to overcome that problem consisted of a thin slice of Germanium as a bulk resistor, a single four-port bipolar transistor and wires of gold. And so, by pressing a switch and showing a continuous sine wave through an oscilloscope attached, the first hybrid integrated circuit was born [1].

No one inside that room could possibly imagine the tremendous reach of such modest approach, to the point that it is practically impossible to conceive today's world without electronic integrated circuits. Gazing around, it is hard not to find at least one electronic device containing millions of these little building blocks. Computers, mobile phones, home appliances… it would seem adequate to state that Kilby's legacy left a deep footprint in present-day technology, or even in our own lifestyle.

Only a few months later -and not so many miles away- from Kilby's first IC proof-of-concept, Robert Norton Joyce (co-founder of Fairchild Semiconductor and Intel Corporation and nicknamed "the Mayor of Silicon Valley") gave electronic integration a decisive push by patenting the first, mass-producible, monolithic IC chip made of silicon [2]. This marked the beginning of a technological paradigm shift which would first land into NASA's Apollo Program prior to jump to non-military market in middle 60's.

(a) (b)

Figure 1.1 (a) Jack Kilby's original hybrid integrated circuit prototype, (b) Robert Noyce's original monolithic integrated circuit prototype. Taken from [3].

Commented [DPL1]: Falta fotos.

High-density integrated circuits as we know them today, combining millions of transistors onto a single chip, were brought to life thanks to the invention of metal-oxide-silicon field-effect transistors (MOSFETs) by Mohamed M. Atalla and Dawon Kahng at Bell Labs in 1959 [4]. Regarded by many experts as the "workhorse of electronic industry", these devices showed off from their very conception to bring multiple benefits to the incipient semiconductor ecosystem with respect to state-of-the-art bipolar junction transistors (also known as BJTs) such as less power consumption, much smaller dimension, faster switching speed and relatively simpler processing steps [4].

The next significant milestone of semiconductor history would take place with the creation of Complementary metal-oxide-semiconductor (CMOS) technology by Frank Wanlass and Chih-Tang Sah at Fairchild Semiconductor in 1963 [5]. Before CMOS, PMOS (p-type MOS) and later NMOS (n-type MOS) logic were widely used for implementing logic gates [6]. Although initially NMOS was faster and less expensive, it was eventually overtaken by CMOS in the 80s as the dominant MOSFET fabrication process due to its high noise immunity and low static power consumption, which allowed this device to give off less heat waste. As a result, by 2011, 99% of the semiconductor ICs -including most analog and some high-frequency circuits- were fabricated in CMOS technology [7].

And the rest of the journey is history. All aforementioned achievements (along with the arrival of new machinery capable of writing smaller lines on the chips, larger wafers and handling and processing equipment, new cooling techniques and packages

to bring out all the connections…) made a solid start for integrated electronics to experience an exponential, unparalleled boost during the following decades, going from the earliest experimental 16-transistor MOS IC designed in 1962 to the most recent microprocessor release of Apple, the 114-billion MOSFETs dual-die M1 Ultra system in 2022. Over the past decades, transistors have been continuously scaled down in size from the 10 μm process provided by leading semiconductor companies of that time such as RCA and Intel to the current 5 nm-technology node currently manufactured by Samsung and TSMC from 2020 [8]. The main reason to make transistors smaller is to pack more and more devices in a given chip area, resulting in more compact architectures, or chips with more added functionality in the same area. Also, smaller ICs allow more chips per wafer, reducing the cost per chip. But, as astonishing as this remarkable evolution in MOSFET size and count per chip may seem today to us, it did not come as a surprise for semiconductor industry luminaries Robert H. Dennard and Gordon E. Moore. Dr. Dennard -also known for being the inventor of dynamic random-access memory (DRAM)- posited on a 1974 paper that, with every technology generation, transistor dimensions could be scaled by -30% and the operating frequency could be increased by 40% while power consumption of each individual transistor would decrease by 50% [9]. This observation went in accordance to a prediction made by Moore eleven years back, affirming that the number of transistors in an integrated circuit doubles about every two years [10]. And time has proven both forecasts considerably accurate -as depicted in Figure 1.2-, to the point that Moore's prediction has been used in industry along the years to guide long-term planning and to set targets for research and development, thus functioning to some extent as a self-fulfilling prophecy.

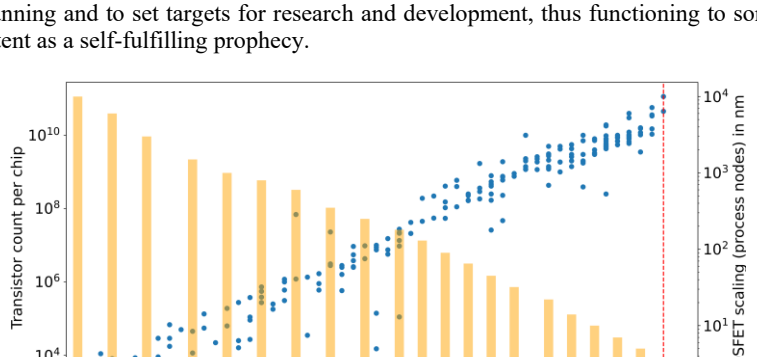


Figure 1.2. Temporal evolution of the transistor count per chip (in blue) and MOSFET scaling (in orange) between years 1971 and 2024 (estimated). Data source: [11].

Based on these facts, there is a good reason to remain optimistic about the electronic semiconductor industry healthiness. In fact, several manufacturers have already

Commented [DPL2]: Falta datos de cada serie.

Commented [ALH3R2]: E de Wikipedia. Lo he citado. Es

planned to jump to 2 nm-production in 2025 [12]. However, -as pointed out by many experts- three interlaced threats lying ahead in the horizon have the potential to slow down -and eventually stop- the journey started by Kilby, Norton and co. more than sixty years ago: heat, power and size [8], [13].

Making processors smaller comes at the inevitable cost of eventually heating it up as a result of tightening up the flow of electrons inside of it. As a result, all today's most powerful chips incorporate mechanisms to cool them down and hence prevent a meltdown. And the amount of required power -and hence, the cost- of these cooling systems can become prohibitively large when moving to bigger infrastructures such as Facebook, Yahoo or Google massive data centers, to cite a few.

Physics sets also a natural limit on how much we can shrink transistors. Silicon has a lattice parameter of 0.543 nm, meaning that the 2 nm-production forecast of semiconductor manufacturers would be less than four times bigger than the characteristic spacing of silicon atoms! And even if we actually had the technology to make transistors of the size of an atom, we would still need to face side effects such as quantum tunnelling, with electrons unexpectedly sliding off their paths.

Two proposed approaches to breathe some life into the eventual demise of Moore's law are the use of multi-core processors and three-dimensional semiconductors. As the name implies, multi-core processors consist of two or more separate processing units reading and executing program instructions on a single chip. To date, they are widely used across many application domains such as general-purpose, embedded, network, digital signal processing (DSP) and graphics. However, developing software for multicore systems (in tasks so that smaller portions of a problem can be executed simultaneously inside each core) can sometimes become a challenge. And, in addition, this new architecture is not free from the same heat and power consumption concerns appearing in single-core systems [14].

The jump to three-dimensional semiconductors is not also unchallenging. Monolithic 3D technology enables multiple transistor layers above a single substrate by providing vertical interconnects with physical dimensions similar to conventional metal vias. However, power and thermal challenges are the biggest hurdles for many 3D IC projects. It is not sufficient to perform power and thermal analysis of the individual dies in isolation using electronic design automation (EDA) tools, since the top ones get their power from the lower ones, so those must be therefore capable to dissipate much more heat than traditional 2D chips. In addition, it must be ensured during fabrication that the reliability of devices within the bottom layer can be degraded by the temperature and processing time required to fabricate high quality transistors at upper layers [15].

1.1.2 Photonics to the rescue: a paradigm shift

A very cunning -and unexpected- solution would come from electronics' younger sibling: photonics. Unlike electronics, where -as the name implies- we deal with the emission, behavior and effects of electrons, photonics does the same with photons. And the advantages of bringing photons into the semiconductor arena are enormous.

Optical waveguides are the primary components of integrated photonics [16]. These elements are essential for bidirectional light guidance in integrated circuits, enabling efficient energy transmission in both directions with minimal loss. Unlike metal traces or copper wires (and such as with discrete components like fiber optics), they do not suffer heating -which, in turn, causes energy and information losses- due to resistance. In addition, unlike with electrons, photons move at the speed of light with almost no interference with other photons. As a result, the data transfer rate and speed of photonic circuits can be several orders of magnitude greater than that from conventional electronic circuits [17].

Following the wake of electronics, the origins of integrated optics date back to the late 60s with the demonstration of the first 2-D waveguides on planar surfaces using lithium niobate ($LiNbO_3$) [18]. In the mid-70s, operational three-dimensional waveguides were demonstrated in a wide variety of materials, from glasses to crystals and semiconductors. But it was not until mid-80s when optical waveguides were successfully implemented in silicon for the first time [19], raising up the curtain of a frenetic race towards the implementation and standardization of high-volume manufacturing silicon photonic chips to reduce cost, making use of past decades of research and manufacturing experience gained from the microelectronics industry.

Along the years, many other key components (including active components) in optical communication have also been progressively miniaturized. Optical modulation -the ability to imprint information onto light prior to transmission- can be achieved in silicon platform either by altering the device's refractive index (and consequently the speed of light to enable interference between two light paths) or its absorption coefficient (a measure of its light blocking ability). To date, both of these approaches are able to provide 100 Gb/s operation with impressive modulation efficiencies [20].

The integration of lasers, however, remains as one of the most challenging aspects of silicon photonics. Silicon does not lase, as it does not give off photons when driven with electrons because of its indirect band gap structure. To date, such limitation has been addressed either by using an external laser coupled to the silicon photonics chip or using hybrid integration by bonding a III-V laser.

External lasers are mature optical components with a wide choice of suppliers. In addition, their physical separation from the chip allows an easier temperature

handling. Because of this, research on fiber-to-chip interconnects has been a permanent matter of interest to scientific community [21], [22]. On the other hand, many efforts from silicon photonics players are being made towards hybrid or heterogeneous integration for the sake of compactness. This process involves constructing a photonic integrated circuit using two or more materials, and is performed at the packaging stage, after fabrication through butt coupling or flip-chip strategies, amongst others [23], [24].

The same issue with lasers applies to photodetectors. Again, as an indirect band gap semiconductor, silicon is not an efficient light absorber at standard telecommunication wavelengths (1310 nm and 1550 nm). To overcome such issue, many experts have turned their attention to the use of Germanium (Ge) thanks its large absorption coefficient at near-infrared frequencies. Thanks to its CMOS compatibility, relatively high responsivity, low size requirements and high speed, germanium-on-silicon photodetectors have emerged as the preferred solution by chipmakers to build photodetectors on silicon [25], [26].

In a world that thirsts for data delivered at high speed, global interest in photonics is, naturally, growing. Sustained by these fruitful technology achievements, silicon photonics market is predicted to reach a value of $1.49 billion in 2022 at a compound annual growth rate (CAGR) of 24.13% and $3.44 billion at a CAGR of 23.23% in 2026 [27].

To meet the current market demands for silicon photonics manufacturing, a variety of open-access platforms is offered by CMOS pilot lines, R&D institutes and commercial foundries not only for silicon-on-insulator (SOI) platform, but also for other prominent material flavors such as Indium Phosphide (InP) and Silicon Nitride (SiN), amongst others [28].

Indium Phosphide is a binary semiconductor composed of indium and phosphorus [29]. To date, this is the only technology capable of the monolithic integration of active and passive photonic components, providing the most complete list of available components for integration. InP can realize lasers emitting in the common telecommunications bands between 1.26-1.625 μm, and high-performance modulators and photodetectors.

Silicon Nitride is a chemical compound of the elements silicon and nitrogen [30]. Today this integration platform allows the fabrication of single-layer waveguides, multi-layer waveguides (comprising both symmetric and antisymmetric double stripe and box-shaped) and buried waveguides, offering ultra-low optical attenuation from the visible to beyond the infrared —a range not accessible through other platforms. The main disadvantage of this technology, though, is that it is full passive. Consequently, no optical sources, detectors, amplifiers, or modulators are available for integration without resorting to hybrid or heterogeneous integration with III-V

gain materials. However, its relatively low index contrast (around 3.5 to 4.5 less than that of submicrometer SOI) leads to reduced loss and an order of magnitude weaker backscattering, apart from 5 times lower temperature sensitivity [31].

While this degree of material diversity (in contrast to integrated electronics, where CMOS in its own variety of nodes is by far the most dominant technology) may be seen as a severe drawback in the pursue of a single standardized technology platform, it also helps developing a sense of competition and even of cooperation between foundries and technologies that allows integrated photonics to target more specific markets.

1.2 Programmable Integrated Photonics

1.2.1 From specificity to large-scale production

Picking up our discussion from we left off, it would seem that the future of integrated photonics is at its brightest. Indeed, year after year, the ecosystem has seen more and more players enter the marketplace with new products and acquisitions. However, at present this is not entirely true. The scaling of silicon photonics is lagging with respect to that of electronics, and we can only see sizeable silicon photonics product volumes in datacenter transceiver markets.

A great responsible for this outcome is no other than the current state of photonic circuit design process (Figure 1.3). While the design flow for analog electronics enables first-time-right design, tools and practices enabling this for photonic circuit design are not yet established in the overall PIC community. A full-custom photonic circuit often needs several costly fabrication iterations before it performs its function within specifications, because the design tools and foundry design kits do not yet support good models and predictive variability modelling. This immaturity and uncertainty in the development cycle for new photonic chips is proving to be a major obstacle for rapid adoption [32].

Today, virtually all PICs are application specific: they are designed to perform a single or a few functions, targeted to the needs of a specific application. As already mentioned, these application specific PICs (ASPICs) are costly and risky to develop, as it can take two or three design-fabrication-test cycles, each taking up to a year, to get the circuit working to specification [33], [34]. This slow development cycle is detrimental for testing the viability (both technical and economic) of a new product and its market potential and demands for solutions to cut it short.

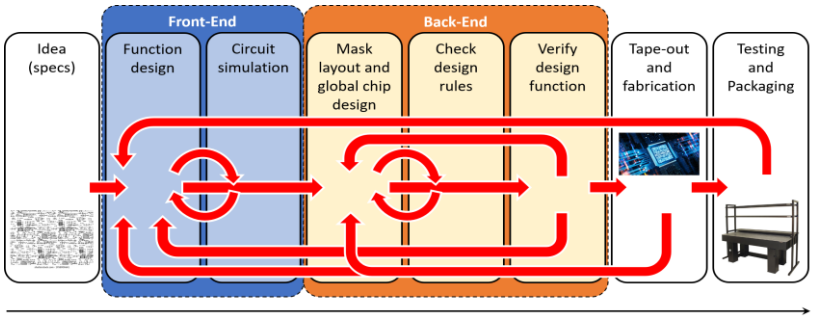


Figure 1.3. Photonic circuit design flow. The horizontal axis indicates the sequence of design steps.

At this point, it should not come as a surprise to the reader where to look again for an answer for this challenge. Back in 1984, the ingenuity of two American electrical engineers named Ross Freeman and Bernard von der Schmitt (co-founders of Xilinx) gave birth to a revolutionizing invention that would shake the world of reconfigurable electronic computing: the Field-Programmable Gate Array (FPGA) [35]. FPGAs are electronic integrated circuits designed to be configured by a customer or a designer after manufacturing. They contain an array of programmable unit blocks, and a hierarchy of reconfigurable interconnects allowing blocks to be wired together. Logic blocks can be configured to perform complex combinational functions, or act as simple logic gates like AND and XOR. These devices can be reprogrammed to implement different logic functions, allowing flexible reconfigurable computing as performed in computer software.

Unlike application-specific integrated circuits (ASICs), FPGAs do not require custom mask tooling, with the subsequent saving in risk and up-front non-recurring engineering (NRE) costs. These savings ensure FPGAs to be more cost effective than ASICs at some volume, making that solution more affordable to end users. For high volumes, the ASIC would feature a lower overall cost since the FPGA would consume a larger chip footprint and require more driver electronics and available ports for many applications [35]. However, only few PIC-based products really require such production volumes, as measured by the standards of silicon foundries [36], [37].

So, the question we must ask to ourselves is: can we come up with any photonic hardware design to perform a wide variety of functions for different applications using off-the-shelf components such as FPGAs do? The answer, as could not have been otherwise, is yes. And so, the concept of Field-Programmable Photonic Gate Arrays (FPPGAs) was born [38].

1.2.2 Reconfigurable waveguide meshes and programmable unit cells

Such as for FPGAs, the working principle of FPPGAs lies on the interconnection of 2-input, 2-output optical analogue gates -hereinafter referred to as *programmable unit cells* or simply PUCs- acting as tunable couplers in a two-dimensional mesh network (or optical core) [39]. These devices, each working as a photonic processing unit, should allow an independent tuning of both power splitting and relative phase delay either through electro-optic [40], opto-mechanic [41] or thermo-optic [42] effects.

The most common on-chip implementation following last approach - the one to be considered throughout this Thesis- is with a balanced Mach-Zehnder interferometer (MZI) loaded with a heater on each arm such as the one shown in Figure 1.4. Its basic unit length (BUL) is determined by the addition of the overall length of access waveguides and the length of the tunable coupler. Other alternative implementations of PUCs based on dual-drive directional couplers have also been explored in the literature [43].

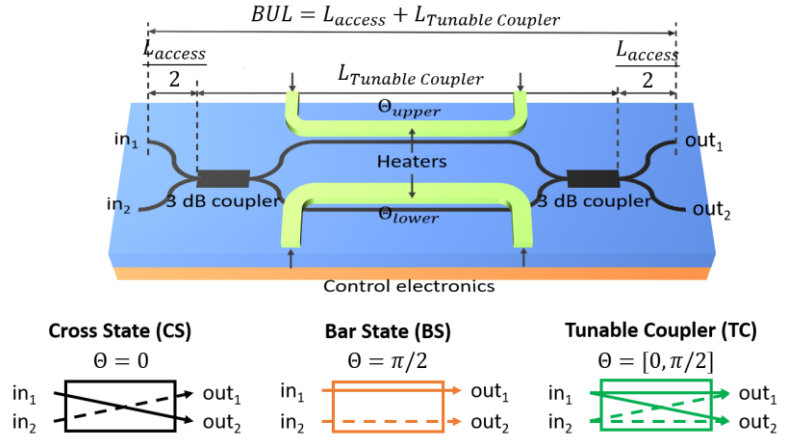


Figure 1.4. Schematic of an integrated balanced MZI-based tunable coupler acting as the programmable unit cell of the waveguide mesh and signal flow for the different PUC operation regimes [44].

Commented [DPL4]: Esta del OPEX 2015 y/o la de squar

We can configure the splitting ratio of these devices by increasing the effective index in either its upper or lower arm through Joule effect, producing a ϕ_{upper} or ϕ_{lower} phase shift, respectively. A common phase shift can be then delivered by applying a common drive in both heaters. By doing so, a programmable unit cell can work under three operation regimes: *cross state switch* (with light travelling from in_1 to out_2 and

from in_2 to out_1), *bar state switch* (connecting in_1 to out_1 and in_2 to out_2) and *tunable splitter*. And, as we are about to see, this brings the potential to synthesize many given photonic integrated circuit topologies -such as optical delay lines or wavelength filters- inside our optical core.

Figure 1.5 presents several demonstrated waveguide mesh arrangements architectures. The initial concept showed in Figure 1.5 (a) was pioneered by Reck et al. back in 1994 under the name of *universal multiport photonic interferometer* [45]. This architecture was later expanded by Miller to provide self-reconfiguration capabilities and to facilitate the use of non-ideal components [46]. And, in [47], Clements et al. proposed a new design based on a rectangular arrangement requiring half of the optical depth than Reck's original design, hence minimizing optical losses and fabrication resources.

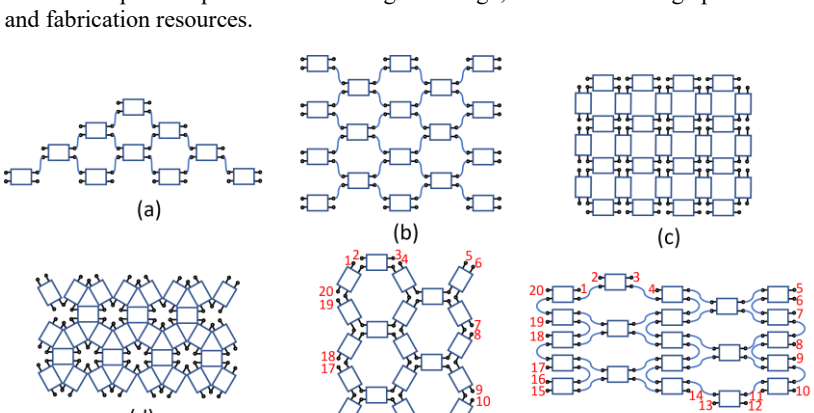


Figure 1.5. Summary of state-of-the art waveguide mesh architectures: (a) Triangular unitary architecture, (b) rectangular unitary architecture, (c) rectangular recirculating mesh, (d) triangular recirculating mesh, (e) hexagonal recirculating mesh, (f) flattened hexagonal recirculating mesh.

One limitation of both Reck and Clements architectures is that they are forward-only, which means that both arrangements only allowed one-directional propagation of light from one side to the mesh to another, hence separating ports into a set of inputs and outputs. As a result, a second category of meshes -*recirculating meshes*- was born in 2015 and 2016 by the hand of Zhuang et al. [44], and Pérez et al. [48]. By contrast to forward-only meshes, recirculating meshes are particularly attractive because they allow users to program a full scattering matrix between all the mesh ports. In addition, these arrangements unlocked the synthesis of any kind of optical

Commented [DPL5]: El co
que he editado un poco esta par

Commented [DPL6R5]: E
cuadradas y hexagonales y trian
Para que nos hagamos una idea
mio de 2014 sobre un reconfigu
que le faltaba es el core y ya di

core circuit topology, including finite and infinite response filters with tunable sampling periods by appropriate switching along the beamsplitter-based mesh.

Throughout last years, many different circuit topologies have been proposed for recirculating waveguide meshes, including the square one proposed by Zhuang, triangular [43], [48] and hexagonal [49]. As demonstrated in [48], this last arrangement is especially appropriate for the design of programmable photonic processors as it features improved performance in terms of spatial tuning reconfiguration step, reconfiguration performance, switching elements per unit area and losses per spatial resolution. In addition, it is the only one where all ports can be used as inputs or outputs. The work done in [50] introduced a new design for this kind of meshes aiming to improve its integration density, one of its main scalability limitations.

1.2.3 Towards programmable photonic processors: technology stack and applications

Both forward-only and recirculating waveguide meshes can form the core of a working programmable photonic processor such as the one in Figure 1.6 [51]. This final assembly bears some similarity to that of programable electronics, but including additional photonic building blocks such as:

- A set of input/output *optical signal ports* to couple light to inside/outside the chip, respectively. Depending on the waveguide architecture laying at the processor's core, these ports can be arranged surrounding the structure or only at its left- and right-hand sides.
- Dedicated peripheral *high-performance blocks,* such as high-quality filters, long delay lines or high-speed modulators, to expand the FPPGA capabilities and include higher-level functionality fixed into the chip.
- One of the biggest setbacks preventing the synthesis of complex photonic structures on waveguide meshes usually is the accumulated optical insertion loss of these circuits. Hence, the insertion of *optical amplifiers* (preferably at the mesh perimeter to avoid breaking down symmetry) becomes increasingly necessary for certain applications as the number of PUCs in the design increases.
- *Optical power monitors* and *control loops* at strategic points of the circuit to monitor the flow of light throughout the mesh and enable dynamic feedback operations.
- A *software layer* at the end user's command available to reconfigure the processors' internal subsystems by means of electronic control signals. Depending on the software capabilities for a targeted functionality, the

processor might be able to configure itself to optimally perform the application.

- An *electrical layer* bridging the software layer with the optical core by handling all PUC phase shifter actuators, radiofrequency components and packaging using microcontrollers, FPGAs or digital signal processors (DSPs).

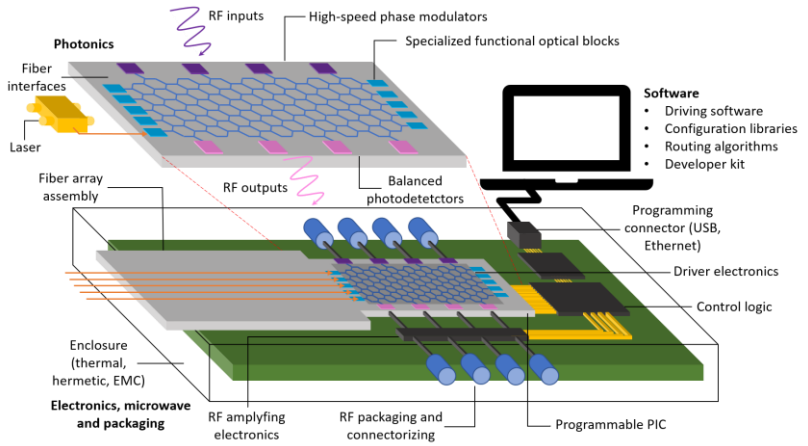


Figure 1.6. Representation of a programmable photonic circuit, including a waveguide mesh acting as optical core driven by control electronics and connected to high-speed phase modulators and detectors. Modified from [51].

Of course, this flexibility and programmability are only useful if they can be put into practice for more than one setting. Indeed, programmable photonics can find applications in a myriad of areas, such as [52]:

- *Microwave photonics* (MWP), the interdisciplinary link between radiofrequency engineering and opto-electronics, has attracted great interest from both the research community and the commercial sector over the past 30 years. Microwave photonic links offer significantly reduced size, weight, low and frequency-independent propagation loss, immunity to electromagnetic waves and high capacity for broadband signals [53], [54]. To date, several MWP signal processing functionalities including signal filtering, temporal differentiation, time delay, beamforming and spectral shaping have been demonstrated on a programmable processor with microdisk resonators working as PUCs [55].

Commented [DPL7]: Si es citarlo aquí. Si está copiada hab hagas tu propia imagen puesto pertenecen al journal en cuestió

Commented [ALH8R7]: L misma pero está girada, he pue

- *Optical switching* and *routing* inside metro and long-haul communication networks and in short-reach communication networks within data centers and inside the processors of high-performance multicore computers, replacing partially or completely conventional electronic switches to eliminate costly and inefficient optoelectronic and electrooptic conversions [56]–[58].
- *Adaptive sensing*, which might lead to a generic class of programmable measuring devices to be successfully integrated as a building block in the future Internet of Things (IoT) [59] or for biosensing applications [60], [61].
- *Quantum information processing,* with the optical core operating as a universal linear interferometer on input photons encoded into quantum states [62], [63].
- *Neuromorphic computing,* the use of very large scale integration (VLSI) systems containing analog circuits to simulate the neuro-biological architectures present in human brain and nervous system [64]. Fully optical neural networks based on multiport interferometers are emerging as captivating approaches to implement these architectures because of their high bandwidth, low propagation losses and low latency [65]–[69].

1.3 Objective and Thesis Structure

The scientific work presented within the framework of this Ph.D. Thesis has been developed in the Photonics Research Labs, a part of iTEAM Research Institute at Universitat Politècnica de València. The scope of this Thesis is to provide a sufficiently robust and flexible software framework to achieve full and agile automation in high-density, complex integrated programmable photonic circuits.

In Chapter 2 we present a software tool to configure optical circuits on a programmable photonic processor based on computational optimization. We will first delve into the foundations that sustain the optimization algorithm of our choice (particle swarm optimization) and then present its application to the synthesis of optical structures of multiple flavors such as optical interconnects, beamsplitters or waveguide-selective filters.

Chapter 3 presents an alternative approach for the synthesis of photonic structures based on graph theory. First, we explore the parallelism between photonic waveguide meshes and graph structures. On top of this, we develop a path finding algorithm that allows the automatic search of inner connections inside the mesh. Finally, we will show many software applications that can be accessed with the help of this novel capability.

The application of all these automated protocols on a real programmable photonic processors is left for Chapter 4. We cover chip designs, software execution and its application to relevant microwave photonics functionalities.

As a wrap-up of this Thesis, the objective of Chapter 5 is to address future work required for software-defined photonic processors to spread the market of photonics and to envision their role in the new horizon of multipurpose programmable photonics.

Commented [DPL9]: Creo de la tesis. He quitado este com 1) el trabajo realizado es muy b apuntar demasiado al future wo 2) Es un comentario que aporta lectura yendo más al grano.

Annex A provides pseudocode and a clear insight about all the software routines and algorithms described in chapters 2 and 3.

1.3.1 Original contributions of this Thesis

- By combining computational optimization and photonics, the self-configuration software tool presented in Chapter 2 constitutes a big step towards the realization of high-density and complex integrated programmable photonics, enabling the automated synthesis or arbitrary interferometric structures on programmable photonic processors.
- The work presented in Chapter 3 based on graph searching algorithms allows to unlock multiple unprecedented software features. They include two novel self-calibration and self-characterization routines that provide a robust check about the photonic processor status, helping the end user to decide whether it is safe to use certain areas of the chip or not. These routines also supply information to facilitate the device operation in the form of coupling factor versus required driving current for every single photonic actuator of the PIC. In addition, we will rely on this approach to present a novel software tool to simulate photonic circuits in any arbitrary waveguide mesh. This method reduces development costs, speeds up the growth of new circuit designs and has the potential to become a fundamental tool for the development of programmable photonic libraries.
- All previously introduced software features are experimentally demonstrated in Chapter 4 using a real photonic processor. Such processor, consisting of a 17-cell layout integrated on silicon and designed in the frame of the ERC-ADG2016-741415 UMWP-CHIP Advanced Grant, will be also used to showcase several unprecedented MWP functionalities on reconfigurable waveguide meshes.

Chapter 2

Self-configuration of photonic circuits in programmable photonic processors

2.1 Introduction

As brought out in the prior chapter, programmable integrated photonic circuits rely on a common hardware that can provide multiple functionalities by suitable programming of control signals. However, scalability —which is essential for increasing functional complexity and integration density— is severely limited by the need to precisely control and configure several hundreds of variables and simultaneously manage multiple control actions. The following two chapters will present two different strategies towards management automation in programmable photonic circuits, enabling simultaneous handling of circuit self-characterization, auto-routing, self-configuration, and optimization by bringing up together computational optimization, graph theory and photonics.

In this chapter, we will delve into circuit programming based on computational optimization routines that obtain iteratively the optimum driving configurations to achieve a targeted functionality [70]. This approach does not require any prior information about the inner workings of the photonic core (such as dynamic tuning crosstalk between actuators, optical crosstalk due to imperfect design and fabrication, nonuniform loss distributions over the circuit…) to operate. Instead, we will treat it as a "black box" returning the scattering matrix datasheet as a function of the chip's passive and dynamic conditions and the electrical inputs. Then, the optical readout system extracts a portion of this scattering matrix and closes the feedback loop of the optimization system, as shown in Figure 2.1.

Commented [DPL10]: JST

This portion of the extracted matrix is employed to compute a cost function to be minimized when a targeted application (signal filtering, optical beamsplitting, point-to-point interconnection…) is achieved. Such customized function serves as a metric

describing the processor's performance for the specific job we demand it to do; however, its definition is not a straightforward task. As a matter of fact, different cost functions can be employed for a same application, thereby achieving different convergence rates, and ultimately compromising the success of the algorithm. Provided that we are observing N different parameters/metrics, the cost functions used throughout this work will take the general form:

$$CF = c_1 f_1 + c_2 f_2 + \cdots + c_N f_N \tag{1}$$

with $f_1, \ldots, f_N$ representing the specific values of each parameter, weighted by their respective coefficients $c_1, \ldots, c_N$ in case we want to prioritize certain parameters over others in the overall cost evaluation.

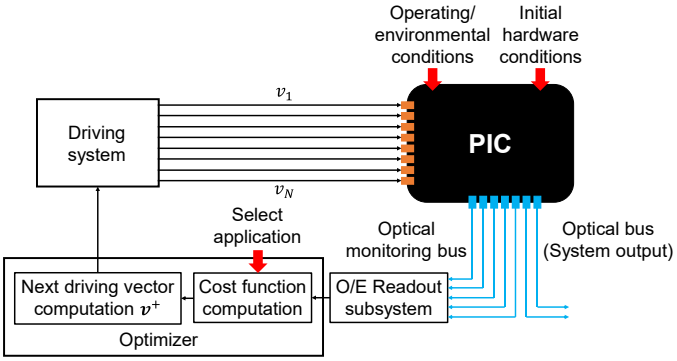


Figure 2.1. Optimization system diagram and its application to self-configuring performance of optical processors. PIC: Photonic Integrated Circuit, v: vector defining the configuration variables of the system for the integrated actuators. The full cycle defines a single operation. Although a real system has an amplitude and phase response, we will employ the overall amplitude response in our application examples.

While the purpose of every optimization method is basically the same -minimizing the cost function by systematically choosing input values from within an allowed set and computing the value of the function-, they can be classified according to different criteria in:

- Global and local search algorithms, depending on their suitability to find the global maximum or their ability to converge rapidly to their closest minimum point.
- Derivative and non-derivative methods, depending on whether the computation of the gradient is employed for each iteration or not.
- Deterministic or stochastic methods, depending on the inclusion of random variables during the procedure.

- Individual or population approaches, depending on the number of search points employed during each iteration.

Throughout the chapter, we will be showing the results obtained using Particle swarm optimization (PSO) algorithm, a global-search, population-based algorithm which will be briefly introduced to the reader in the following section.

2.2 Particle swarm optimization

Commented [ALH11]: A l
fusionarse en una.

Particle swarm optimization maintains at each iteration a swarm of particles (set of points) with a velocity vector associated with each particle [71]. Each of these particles represents a possible solution to the optimization problem, and they move around during the execution of the algorithm in search of the fittest one. In our case, each particle's position will be modelled by the driving vector $\boldsymbol{v}$, which includes the electrical current delivered to each of the processor PUCs' photonic actuators. An illustration example showing this procedure can be observed in Figure 2.2.

At each iteration, this algorithm generates a new set of particles from the previous swarm combining random and inherited parameters defined by the following adjustable settings (also known as hyperparameters):

- The **number of particles** at the swarm corresponds to the total number of possible solutions provided by the algorithm at any given time. Having more particles gives a better chance of finding the best solution, but it will also take more time and computing power.
- The **inertia** coefficient controls how much each particle is influenced by its own previous movement. If this value is high, the particles will keep moving in the same direction for a longer time. If it is low, the particles will change direction more quickly.
- The **cognitive** factor represents how much each particle is influenced by the best solution it has found so far. It represents the particle's own memory of its past experiences and what it has learned from them, trying to improve upon its own best solutions.
- The **social** factor, on the other hand, models how much each particle is influenced by the best solution found by the other particles in the swarm. It represents the particle's ability to learn from the experiences of others and how is trying to improve upon the best solutions found by other particles in the swarm.

Apart from a fine tuning of these coefficients, PSO algorithm performance can be enhanced using mechanisms such as velocity clamping. This is used to bound the maximum velocity of particles, preventing them from leaving the search space and avoiding large oscillations, converging faster [72].

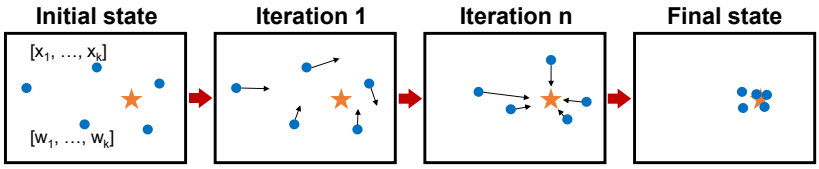


Figure 2.2. An illustration of Particle Swarm Optimization (PSO) algorithm.

Once we have gone through the basics of computational optimization and how we can specifically utilize it in our system, we are ready to move on to the next section, in which we will apply these foundations to the synthesis of three different configuration examples: an all-cross PUC interconnection, a 1x8 optical beamsplitter and the automatic definition of optical filters based on spectral masks [73].

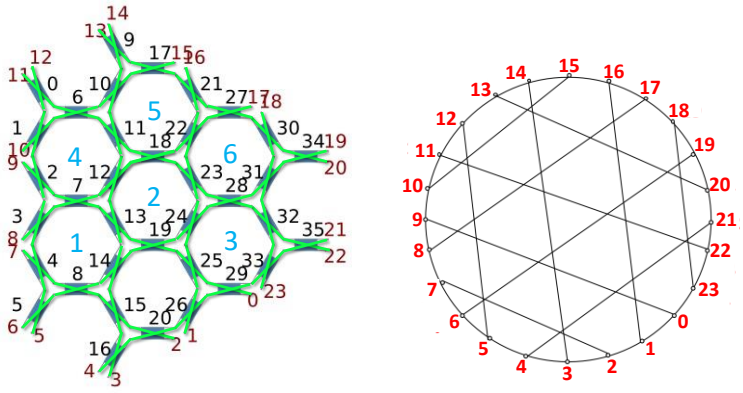
Commented [DPL12]: Au
yo metería referencia al paper d
más detalles y para que se sepa

2.3 Circuit programming

2.3.1 Self-configuration of 'all-cross' waveguide meshes

This configuration aims to drive every phase actuator so that all PUCs in the waveguide mesh are in cross-state. This capability is particularly interesting for both calibration and characterization, as well as for setting an optimized initial point to start the self-configuration of a myriad of structures, improving the speed at which optimization finds a good solution. It is just one example of a broad range of applications handling the flow of optical signals between different ports.

In the example shown in Figure 2.3, the optical channels are identified by the following port pairs: 11-22, 13-20, 9-0, 7-2, 12-5, 14-3, 16-1, 18-23, and 10-15, 8-17, 6-19 and 4-21. It can be noted that last four pairs are redundant and can be removed from cost function definition, as they do not include additional PUCs. Extra features —such as monitoring channels' spectral response to consider the ripples observed or observing every remaining port for each input to prevent the formation of spurious paths (power leakage)— can be added to cost function definition to enhance algorithm performance and/or reduce the number of iterations; however, this could require additional measurements. Although these features have not been incorporated in the cost function definition for this example, they shall be considered for future large-scale waveguide mesh arrangements.



(a) (b)

Figure 2.3. (a) Labelled schematic of the waveguide mesh arrangement under test. (b) Black box system with the targeted performance. Routing between channels defined by port pairs 11-22, 13-20, 9-0, 7-2, 12-5, 14-3, 16-1, 18-23, 10-15, 8-17, 6-19 and 4-21 representing direct connections, without crossings or splitting.

Without further ado, we define the cost function as follows:

$$CF_{all-cross} = \begin{cases} c_1 = -1 \\ f_1 = \dfrac{1}{N}\displaystyle\sum_{1}^{chs}(\log(|H_{chs}|)) \end{cases} \tag{2}$$

where N represents the number of optical interconnections incorporated in the optimization process, while $|H_{chs}|$ takes the maximum absolute value of the measured electric field for each optical interconnection before listed. To perform a qualitative analysis of the achieved performance, we monitor and define the output feature 1 as the average power transmission response in the targeted optical channels defining the all-cross operation. It is expressed in logarithmic units.

To test the performance and to find the best hyperparameter ranges for this configuration problem, we first perform a grid search tuning the hyperparameters and running the algorithm for 320 trials employing the defined cost function. The hyperparameters are selected according to the ranges specified in Table 2-I. These are wide enough to ensure the exploration of different combinations. Those featuring the best performance appear in Table 2-II.

Table 2-I. Summary of particle swarm optimization algorithm hyperparameter values under grid search in our experiments.

Number of particles per variable	Inertia	Cognitive	Social
300 %	2	2	2
200 %	1	1	1
100 %	0.5	0.5	0.5
50 %	0.1	0.1	0.1
20 %			

Table 2-II. Selection of best performance hyperparameters for the synthesis of all-cross configuration using $CF_{all-cross}$ in a 36-PUC waveguide mesh.

$CF_{all-cross}$	Best performance ranges
Number of particles per variable (%)	150 %
Inertia coefficient	0.5
Cognitive coefficient	0.8
Social coefficient	1.0

From the trials, it resulted that only 7.28% of the samples have succeeded in the self-configuration task, confirming the dependency of our algorithm on the hyperparameters and cost function selected. The sensitivity to hyperparameter selection can be mitigated using adaptive values allowing the scheduling of the exploration and exploitation capabilities of the self-configuration task. It results in a robust statistical behavior, once we are close to the optical set of hyperparameters that configure the algorithm. As shown in Figure 2.4, we repeated 100 times the self-configuration routine and obtained an average error better than 3 dB after 4000 operations in the 76.66% of cases. The PSO hyperparameters configuration included an adaptive inertia from 0.9 to 0.35, 108 particles, and cognitive and social coefficients of 0.7 and 1.8, respectively.

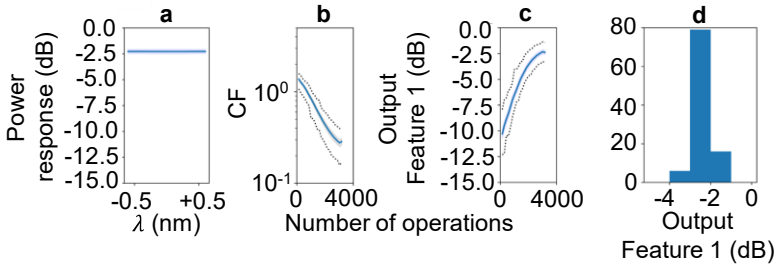


Figure 2.4. Numerical results for the self-configuring of an all-cross function in a 36-PUC hexagonal waveguide mesh using particle swarm algorithm. (a) Spectral response versus wavelength normalized to the basic unit delay (BUD), (b) evolution of the average (solid), maximum and minimum (dotted) and standard deviation (shaded) cost function and (c) output feature (OF), (d) histogram of the last operation (OF1: average of normalized output channels power of the beamsplitters, the datasheet is composed of 100 independent experiments with different arbitrary waveguide mesh initial conditions.

2.3.2 Self-configuration of a 1x8 optical beamsplitter

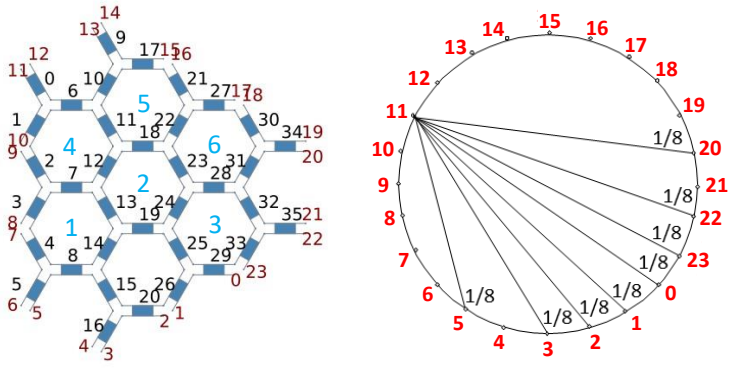
In this example, we aim to set up a group of phase actuators in our waveguide mesh arrangement in a way that allows us to use it as a beamsplitter with one input port and eight output ports, so that the input optical power is divided equally among them. This capability is useful for a variety of signal processing systems and subsystems, such as finite impulse response filters and beamforming networks [74]–[76].

Two possible cost functions $CF_{1,2}^{1\times 8}$ are presented in equations (3) and (4) for comparison. Both consider two features of the performance of the beamsplitter: the difference between the average optical power budget of the eight channels and the expected value (f_1), and the average ripple in the targeted channels (f_2):

$$CF_1^{1\times 8} = \begin{cases} c_1 = \frac{1}{8}, \quad c_2 = \frac{1}{8} \\ f_1 = \sum_{op} \left(10\log_{10}\left(\left|H_{op,6}\right|^2\right) + 10\right)^2 \\ f_2 = \sum_{op} \begin{pmatrix} \max\left(10\log_{10}\left(\left|H_{op,6}\right|^2\right)\right) - \\ \min\left(10\log_{10}\left(\left|H_{op,6}\right|^2\right)\right) \end{pmatrix} \end{cases} \tag{3}$$

$$CF_2^{1\times 8} = \begin{cases} c_1 = -1, \ c_2 = -1 \\ f_1 = \dfrac{20}{N}\displaystyle\sum_{1}^{chs}\left(\log\left(1 - \frac{||H_{chs}| - 0.31|}{0.69}\right)\right) \\ f_2 = 20\log_{10}\left(1 - \dfrac{1}{N}\displaystyle\sum_{1}^{chs}(\max|H_{chs}| - \min|H_{chs}|)\right) \end{cases} \quad (4)$$

where op references the optical ports under use by this configuration. Note that, in this case, we are not employing the information coming from the signals from the non-targeted points to reduce the number of reads by a practical read-out system. In both cases, we consider the average signal in the targeted optical channels and the ripple at the channel. This is an example of a cost function that employs spectral information, meaning that if low-speed diodes are employed, a laser sweeping multiple wavelengths would be required in a real system implementation. Alternatively, a filtered WDM spectrum could be photodetected at each spectral channel, increasing the complexity of the system. The use of extra features and the consideration of non-used or secondary ports is particularly interesting for larger-scale waveguide meshes. Figure 2.5 (a) and (b) both show the labelled waveguide mesh employed in this example together with a schematic view of the power splitting from port 11 to ports 5, 3, 2, 1, 0, 23, 22 and 20.



(a) (b)

Figure 2.5. (a) Labelled schematic of the waveguide mesh arrangement under test, (b) black box system with the targeted performance. Routing between channels defined by port pairs with input 11 and outputs 20, 22, 23, 0, 1, 2, 3 and 5.

The cost function defined in equation (4) was already studied in [50], so here we will focus on analysing $CF_2^{1\times 8}$. To do so, we define and keep track during the process of

two output features at each iteration. The first one, referred as Output Feature 1 (OF_1) computes the mean optical loss at each channel and is normalized to -10 dB. The second one, OF_2, captures the main ripple in dB at the 8 targeted channels.

To start, we perform again a grid-search tuning the hyperparameters from Table 2-I and running the algorithm for 320 trials for the proposed cost function $CF_2^{1\times8}$. Those featuring the best performance appear in Table 2-III.

Table 2-III. Selection of best performance hyperparameters for the synthesis of a 1x8 beamsplitter using $CF_2^{1\times8}$ in a 36-PUC waveguide mesh.

CF_{Filter}	Best performance ranges
Number of particles per variable (%)	200 %
Inertia coefficient	0.5
Cognitive coefficient	0.5-1
Social coefficient	1-2

Once this "optimum" set of hyperparameters has been located, we launch 100 trials using it and changing the initial phase offsets of the PUCs at each trial, for which we limit the number of operations (i.e., the number of single configurations of the waveguide mesh and the extraction of the amplitude scattering matrix at the eight output channels) to 3000. Results can be observed in Figure 2.6.

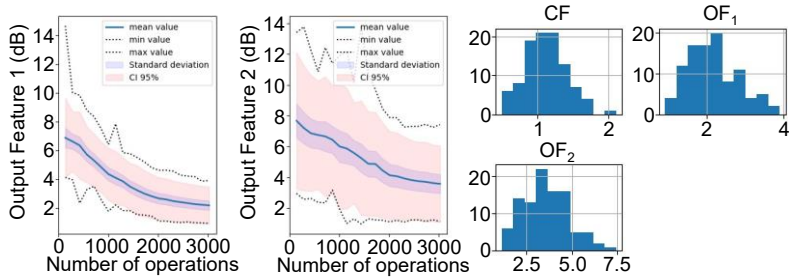


Figure 2.6. Optical beamsplitter statistical results for fixed hyperparameter selection with PSO algorithm for $CF_2^{1\times8}$: evolution of the output features (OF_1: mean of normalized output channels power of the beamsplitters, OF_2: mean ripple at the output channels). Progress (left) and histogram at last iteration (right). The datasheet is composed of 100 independent experiments with different waveguide mesh initial conditions.

From this statistical analysis, we can see that 89% of the trials have obtained an OF_1 better than 3 dB. Maintaining this performance, a 31% of the trials achieve an OF_2 better than 3 dB. The trends suggest that a larger number of operations would improve the statistical result. However, we see that the improvement rate is reduced for OF_2,

suggesting that some of the samples might be close to a local minimum. The efficiency of this algorithm (i.e., better convergence speed and less sensitivity to hyperparameters) could be improved if parameters like the inertia are configured adaptively to decrease during optimization process.

2.3.3 Self-configuration of optical filters

2.3.3.1 Filter self-configuration on mesh arrangements with non-ideal components

The design and configuration of optical filters as application-specific photonic integrated circuits has been discussed in multiple papers and books [74]. This usually involves choosing a specific architecture design and building it using optical splitters, combiners, and waveguides. In this subsection, we will demonstrate a way to automatically configure optical filters using a general-purpose photonic processor. The goal of this application is to suppress a specific range of wavelengths while keeping losses as low as possible in the passband. While our waveguide mesh arrangement can be set up to work as a filter using pre-determined settings and routines, having a method to configure filters on demand is a powerful capability. Additionally, letting an automated function to choose from thousands of parameters is a good solution for scalable systems, and can help deal with issues such as optical crosstalk, tuning crosstalk, power savings, logical footprint savings and optical loss improvements.

When defining our cost function, we can consider different features such as the filter passband insertion loss, the extinction ratio, its roll-off, spurious optical power at the non-targeted ports, and so on. This is shown in Figure 2.7. We can also specify the targeted spectral mask of the filter and create a cost function that measures the difference between the obtained spectral trace and the mask at each iteration. In this example, we use only the mean square error (MSE), but it is possible to add other objectives to the cost function definition to improve performance.

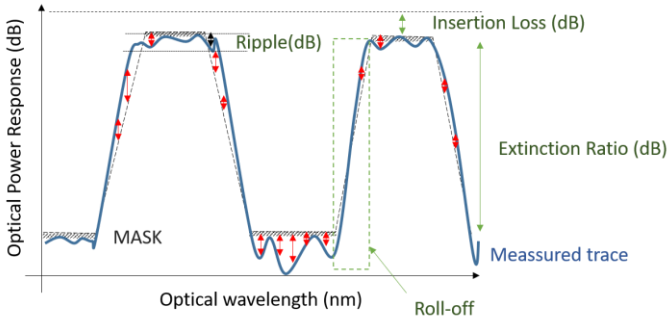


Figure 2.7. Optical filter performance scheme for the construction of the cost function CF_1^{Filter}.

The following equation describes the cost function under use:

$$CF_{Filter}\begin{cases} c_1 = 1 \\ f_1 = \dfrac{1}{N_\lambda}\sum_{\lambda}(M_\lambda - 20\log_{10}(|S_{4,1}(\lambda)|))^2 \end{cases} \tag{5}$$

where N_λ represents the number of wavelength points, M_λ is the value of the spectral mask at each wavelength point and $S_{4,1}(\lambda)$ is the value of the scattering matrix at the optical channel defined between ports 4 and 1 at a given wavelength. Note that this operation is equivalent to the average of the distance between the mask and the measured/simulated spectral trace. For the comparison between methods and future cost functions, we also analyze the evolution of two output features, dealing with the insertion loss in the passband (OF_1) and with the extinction ratio of the filter (OF_2).

We can perform a statistical robustness test of our model from the fixed set of "optimum" hyperparameters shown in Table 2-IV. Again, this set was retrieved from a preliminary broad, grid search tuning the hyperparameters from Table 2-I and running the algorithm for 320 trials using the proposed cost function. In this case, all the hyperparameters are fixed; however, a better convergence speed and less hyperparameter sensitivity would be achieved again if parameters like inertia are configured to decrease during the optimization process. We marked as valid those samples accomplishing OF_1 better than 4.5 and OF_2 better than 18 dB.

Table 2-IV. Selection of best performance hyperparameters for the synthesis of optical filters using CF_{Filter} in a 36-PUC waveguide mesh.

CF_{Filter}	Best performance ranges
Number of particles per variable (%)	200 %
Inertia coefficient	0.5
Cognitive coefficient	0.5-1
Social coefficient	1-2

Then, we test the method 100 times with variable initial conditions for the waveguide mesh arrangement. We limited the number of operations to 3000 for every trial, so slower convergence samples will be considered as failed. It is worth noting that each operation implies a single configuration of the waveguide mesh and the extraction of the amplitude scattering matrix at only one output channel.

Results plotted in Figure 2.8 and Figure 2.9 illustrate that 71% of the samples achieved targeted performance. In addition, the trends suggest that all trials are close to global optimum and a large number of operations would meet the requirements.

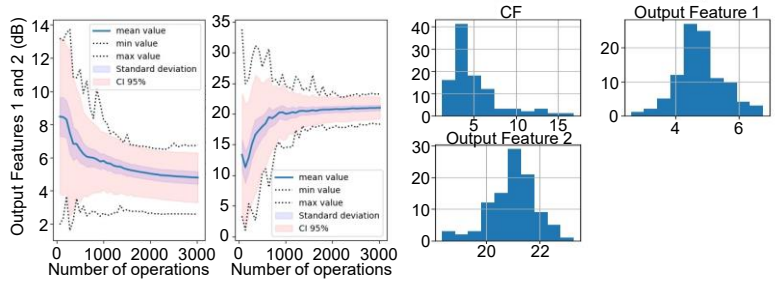


Figure 2.8. Optical filter function statistical results for fixed hyperparameter selection with PSO algorithm for CF_1^{Filter}: evolution of the output features (output feature 1: insertion loss of the passband in dB, output feature 2: extinction ratio in dB). Progress (left) and histogram at last iteration (right). The datasheet is composed of 100 independent experiments with different waveguide mesh initial conditions.

We can then verify the spectral response statistically considering the final configuration for each trial. As observed in Figure 2.9, the mask is accomplished notably, and it shows that the device finds challenging achieving the targeted insertion loss while maintaining the conditions fixed for the stopband and the passband in terms of flatness and bandwidth. We believe that a large-scale waveguide mesh architecture will have more freedom to combine and split the light and thus achieve more challenging optical filter design demands.

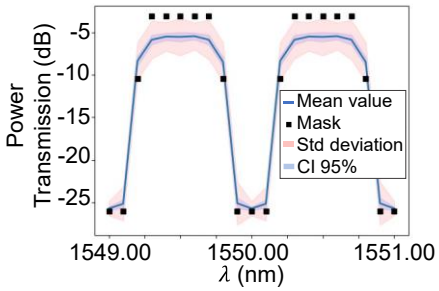


Figure 2.9. Optical filter function statistical results for fixed hyperparameter selection with PSO algorithm for CF_1^{Filter}: statistical results considering the spectral response after the self-configuration of the filter. The datasheet is composed of 100 independent experiments with different waveguide mesh initial conditions.

The flexibility of the self-configuration approach is demonstrated with the following application examples. In all of them, we maintained the hyperparameters employed in the previous example. For each figure, the datasheet employed is specified. Every trial is statistically independent from each other, and we consider random an unknown offset value for each PUC. In all of them, ports 1 and 4 are employed.

First, Figure 2.10 includes the spectral responses and progress for different spectral masks. The first three examples maintain the targeted free spectral range, corresponding to a 2-PUC difference or cavity. The resolution (understood as the number of samples of the spectral response) is low. We can see how the modification of the four points at -10 dB from the first, second and third subfigure, modify the passband and stopband responses. The four example targets a different free spectral range, related to a 4-PUC difference or cavity length. It finds the challenge of spectral filters when trying to optimize the extinction ratio in a single frequency while maintaining a flat passband with low insertion loss. In this case, a mean ripple of less than 2 dB is produced to achieve a narrow stopband.

A great advantage of multipurpose meshes is that we can configure additional spectral masks with different spectra range while maintaining the same inputs and outputs as in the previous example. The following examples, illustrated in Figure 2.11, configure different masks while maintaining a free-spectral-range associated to 6-PUC interferometric lengths.

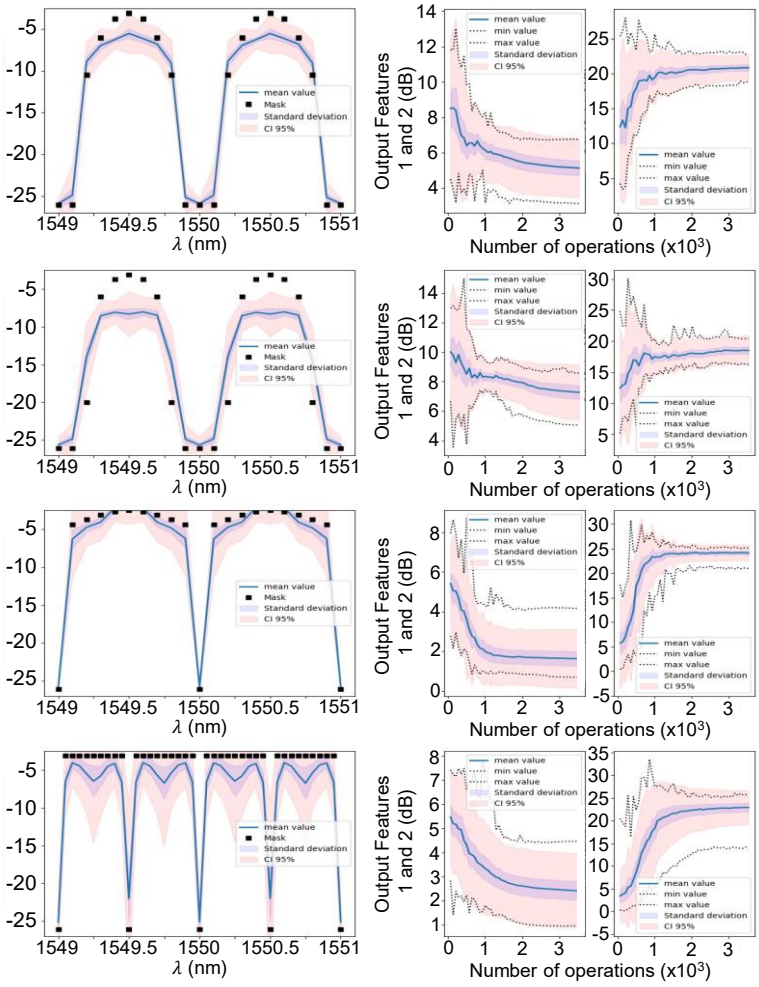


Figure 2.10. Optical filter function statistical results for fixed hyperparameter selection with PSO algorithm for CF_1^{Filter}: spectral response, evolution of the output features (output feature 1: insertion loss of the passband in dB, output feature 2: extinction ratio in dB). Each datasheet is composed of 20 independent experiments with different waveguide mesh initial conditions.

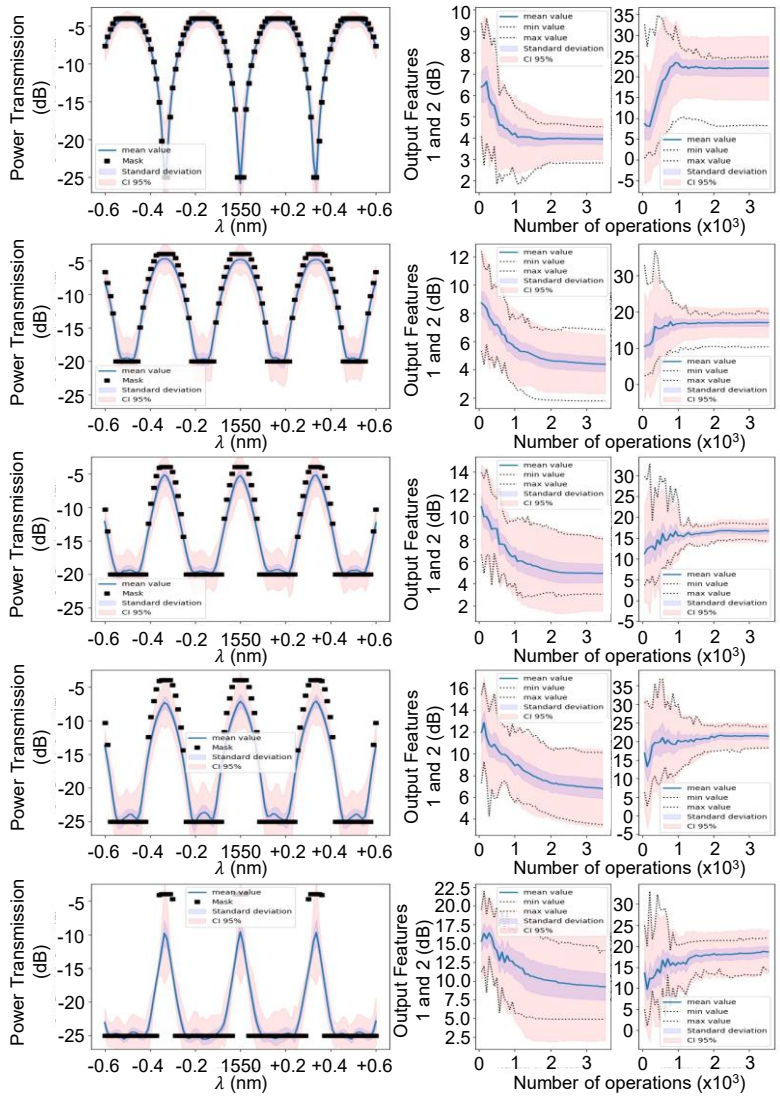


Figure 2.11. Optical filter function statistical results for fixed hyperparameter selection with PSO algorithm for CF_1^{Filter}: spectral response, evolution of the output features (output feature 1: insertion loss of the passband in dB, output feature 2: extinction ratio

in dB). Each datasheet is composed of 20 independent experiments with different waveguide mesh initial conditions.

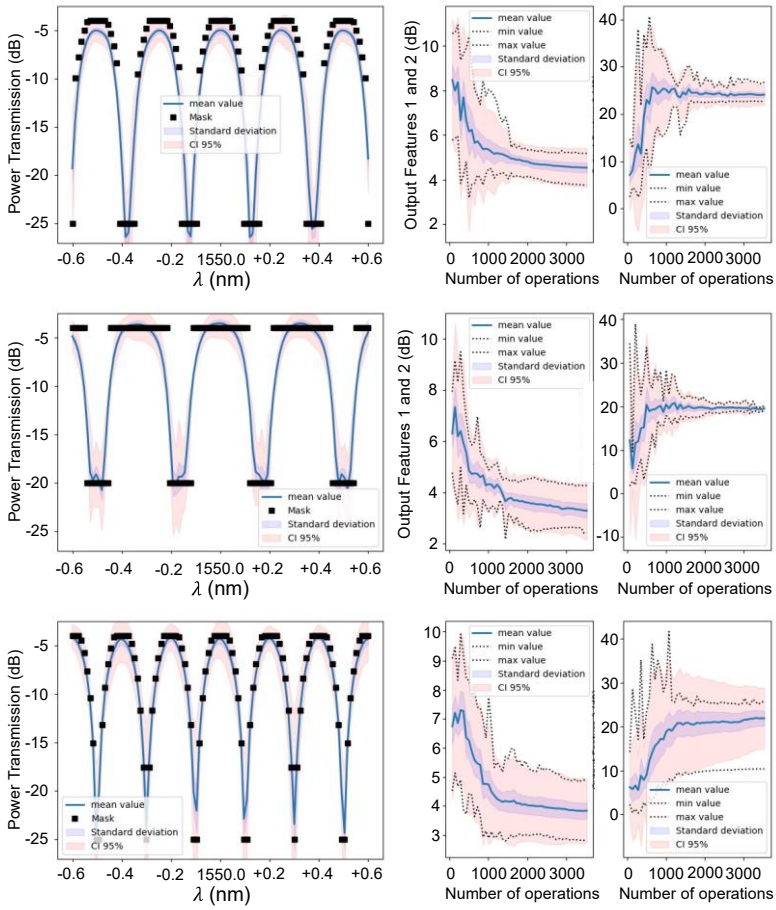


Figure 2.12. Optical filter function statistical results for fixed hyperparameter selection with PSO algorithm for CF_1^{Filter}: spectral response, evolution of the output features (output feature 1: insertion loss of the passband in dB, output feature 2: extinction ratio in dB). Each datasheet is composed of 20 independent experiments with different waveguide mesh initial conditions.

The first example, illustrated in Figure 2.11 (a), is a mask that resembles the reflection response of a ring resonator with the notch centered at a normalized frequency equal to 0 (1550 nm). We observe how this filter achieves an average extinction ratio of 22 dB and an average insertion loss of 4 dB after the self-configuration process. The spectral mask is achieved notably. The second example, shown in Figure 2.11 (b), targets a more selective filter. In this case, the mask reduces the passband region and increases the stopband range. The extinction ratio specifications are reduced to 16 dB. The remaining examples continue both reducing the passband range and increasing the stopband ratio to produce more selective filters. To meet the specifications, the self-configuration routine returns a filter with higher insertion loss. In the last two cases, we observe how the trends of each process suggest that a larger number of iterations would improve the system performance in both output features.

As shown in Figure 2.12, we can also change the mask to alternative free spectral ranges including 8, 6, and 10-BUL interferometric paths, respectively. When more periods are obtained, it is necessary to either increase the number of wavelength points or reduce the frequency range where the mask is evaluated to ensure that we have enough resolution to resolve the spectral response.

2.3.3.2 Filter self-configuration on mesh arrangements with non-ideal components

The scalability of multipurpose waveguide mesh arrangement to higher integration densities is currently limited by several factors related to physical hardware constraints and the precise control and configuration of several hundreds of variables.

However, one of the benefits of programmable multipurpose waveguide meshes is their ability to automatically handle non-ideal fabrication and design defects in the circuit such as the uneven distribution of optical loss all over the circuit, the optical crosstalk and parasitic effects and dynamic crosstalk coming from the undesired tuning mechanism effects in nearby photonic components.

While some of these issues can be also addressed by using pre-characterization routines and modelling the effects to counter-act them during the configuration stage, a full characterization of some of them –such as the tuning crosstalk– can be time-consuming and difficult, especially in large-scale circuits. As an example, determining the tuning crosstalk matrix that computes the crosstalk coefficient between N phase actuators all over the circuit requires the calculation of N^2 coefficients. Additionally, it is not clear whether such coefficients would change depending on the number of phase actuators in use at a given time, and if other parts of the circuit would be somehow affected by this issue.

As mentioned in the introduction, the self-configuration method proposed in this work considers all non-ideal effects as part of the system behavior during the

optimization process. The following examples show the application of this routine to non-ideal circuits providing fault-tolerant, self-healing and error-mitigation capabilities.

Tuning crosstalk

When tuning one phase actuator (such as a thermo-optic actuator), the physical effect causing the tuning in the desired waveguide can spread to the neighboring waveguides, producing an undesired tuning effect. This effect, known as tuning crosstalk, can be appreciated even at distances larger than 10 mm [49].

We model this effect through a constant that reflects the percentage of phase shift occurred in a non-targeted waveguide compared to the experienced by the target waveguide. Simulations and experimental works result in a crosstalk coefficient between 1 and 3% at several hundreds of micrometers. When applied to a system with many phase shifters, this model can be represented by a system of equations relating the effective phase shifts with the phase shifts set by the algorithm or manually by the user.

$$\Delta\Phi_{effective} = \begin{pmatrix} 1 & CT_{12} & \cdots & \cdots & CT_{1N} \\ CT_{21} & 1 & & & CT_{2N} \\ CT_{31} & & 1 & & \vdots \\ \vdots & & & \ddots & CT_{1N-1} \\ CT_{N1} & CT_{N2} & \cdots & \cdots & 1 \end{pmatrix} \begin{pmatrix} \Delta\Phi_1 \\ \Delta\Phi_2 \\ \vdots \\ \Delta\Phi_N \end{pmatrix} \quad (6)$$

We evaluate our algorithm by performing two statistical tests. The first one does not consider thermal crosstalk, while the second one does. In the latter, we load to the performance model [77] a severe crosstalk matrix whose crosstalk coefficients are obtained from a uniform distribution between 0 and 5%. This crosstalk matrix does not contemplate the mitigation of crosstalk with distance, making overall a more challenging configuration scenario. As shown in Figure 2.13, the differences are not noticeable, as self-configuration routine considers dynamic tuning crosstalk effect during optimization.

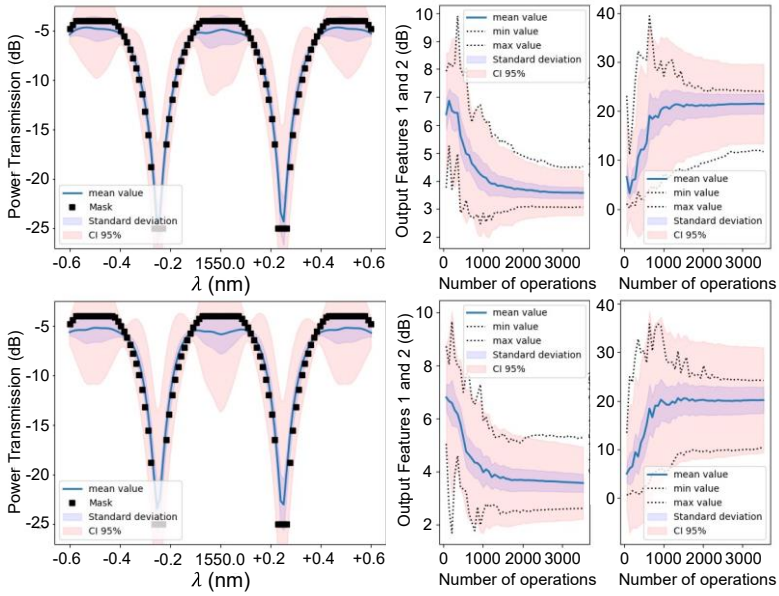


Figure 2.13. Optical filter function statistical results for fixed hyperparameter selection with PSO algorithm for CF_1^{Filter}. From left to right: Spectral response, evolution of Output Feature 1 (Passband insertion loss in dB) and Output Feature 2 (Extinction ratio in dB). The datasheet is composed of 20 independent experiments with different waveguide mesh initial conditions. Upper example: no crosstalk, lower example: 5% crosstalk.

Let us repeat this same experiment on a different structure. This time, we change the reference mask and perform the test without crosstalk, with a 5% crosstalk and with a 10% crosstalk. Even though these scenarios are much more challenging that what would be experienced in a real system, the average performance of self-configuration routine is remarkably good. These results (illustrated in Figure 2.14) open the possibility for employing waveguide mesh arrangements with much higher integration density in our optical core, reducing the distance between components.

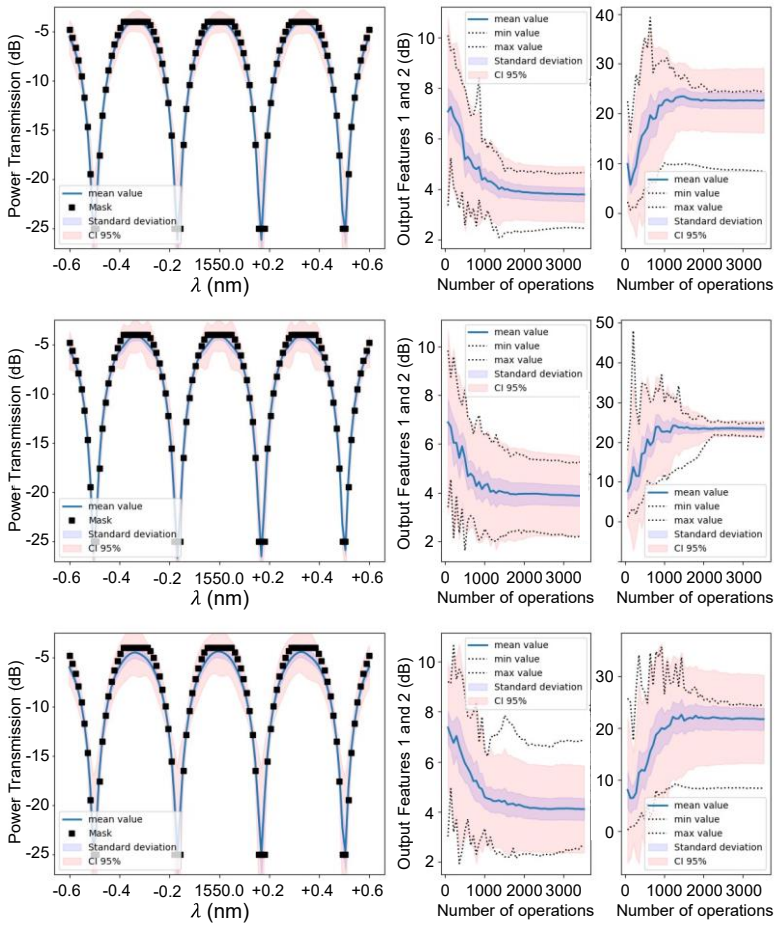


Figure 2.14. Optical filter function statistical results for fixed hyperparameter selection with PSO algorithm for CF_1^{Filter}. From left to right: Spectral response, evolution of Output Feature 1 (Passband insertion loss in dB) and Output Feature 2 (Extinction ratio in dB). The datasheet is composed of 20 independent experiments with different waveguide mesh initial conditions. From up to bottom: No crosstalk, 5% crosstalk and 10% crosstalk.

Fault-tolerant and self-healing effects

Defects during fabrication or degradation over time can decrease the performance of the components integrated in the circuit and even destroy some of its sections. Whereas these issues would render an entire die useless in traditional ASPICs, mesh arrangements offer potential fault-tolerant and self-healing capabilities, as their architecture relies on the repetition and interconnection of simple components. The availability of spare components and sections in the circuit enable the use of alternative circuits when some parts of the optical core are damaged. Figure 2.15 showcases a demonstration example. In this case, we configure the filter specified by the spectral mask illustrated in the figure. Given the demanded FSR, it is likely that cells 1, 2, 3 and 4 from Figure 2.3 (a) and Figure 2.5 (a) will perform as a coupled cavity, defining the targeted mask. In the next example, we decrease the performance of PUCs 13, 14 and 19 by imposing 30 dB of insertion loss on each one. After running the statistical test again, self-configuration process provides alternative structures. Indeed, we can see that it is able to maintain the demanded response, probably employing cavity 5 and less likely 6.

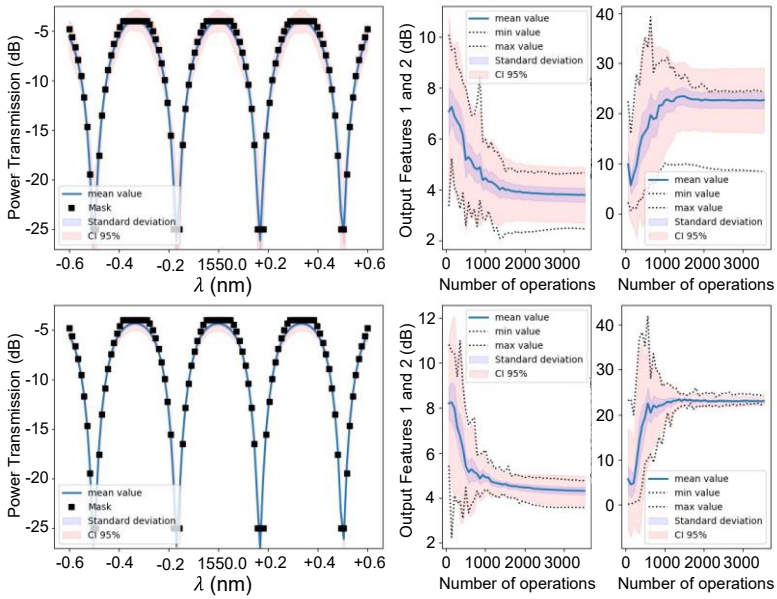


Figure 2.15. Optical filter function statistical results for fixed hyperparameter selection with PSO algorithm for CF_1^{Filter}. From left to right: Spectral response, evolution of Output Feature 1 (Passband insertion loss in dB) and Output Feature 2 (Extinction ratio in dB). The datasheet is composed of 20 independent experiments with different

waveguide mesh initial conditions. Upper example: mesh with good performance, lower example: mesh with PUCs 14, 15 and 20 featuring additional 30 dB insertion loss.

Chapter 3

Applications of graph-based algorithms in programmable photonic processors

3.1 Introduction

In mathematics, graph theory is the study of graphs, which are mathematical structures used to model pairwise relations between objects [78]. Throughout the years, this discipline is becoming increasingly significant as it is actively used in diverse fields. To cite an example, it sustains the operation of the upcoming intelligent transportation systems (ITS), which use location data collected from smartphones and self-driving cars to alleviate traffic congestion and accidents more efficiently by routing cars or aircrafts [79]. However, it is also present in other areas of knowledge such as in social sciences -linking social media users, exploring the impact of rumor spreading… [80]-, biochemistry -constructing molecular structures and lattices, predicting drug-target interactions… [81]- and computer science [82], amongst others.

Commented [DPL14]: Re

Commented [ALH15R14]

In this chapter, we present circuit programming based on pre-sets and global algorithms using graph theory. Unlike for previous section, in which we explored the use of advanced optimization methods for the self-configuration of optical circuits (requiring several iterations until convergence into the desired functionality) the set of protocols presented in this section require a prior knowledge of data regarding the mesh architecture and full bias calibration information of every PUC. The information involving the architecture itself includes the interconnection scheme and the physical features defining the PUCs, such as their basic unit length (BUL) and basic unit delay (BUD). The information regarding the full control of every PUC includes the computation of the non-ideal passive phase offsets of each PUC, the calibration curve (tuning response) of every phase shifter, their estimated insertion

loss, power consumption and the tuning crosstalk matrix that characterizes the undesired coupled effects of neighboring phase actuators.

To obtain all these figures of merit, we also present in this chapter several self-calibration and characterization routines based on iterative maximization and minimization methods and regression-based approximations. With the information gathered, we will be able to present a specific algorithm targeting the automatic search of optimum optical paths or interconnections between any two connections of a photonic processor -or the definition of circuits or interconnections between programmed components and external high-performance blocks (HPPBs). In next subsections, we will use this algorithm to provide several valuable software features to our photonic processor.

3.2 Pathfinder algorithm

3.2.1 Graph construction

Before discussing the algorithm in more depth, let us define some key concepts in graph theory and their adaptation to waveguide mesh-based photonic integrated circuits:

- *Graphs*, the fundamental objects in graph theory, are systems of *nodes* connected in pairs by *edges*. In this work, the nodes are the physical optical ports of the PUCs and the edges represent the connections between them.
- *Weights* are numerical values assigned to each graph edge. The overall weight of any path inside the graph (i.e., the traversed route, with or without repeated nodes —and consequently, edges) will be given by the sum/multiplication of the weights of the edges within such path. In this work, weights are defined as the figure of merit to be optimized during the creation of the optical connections.

Any interferometric structure can be displayed by its reciprocal graph representation. To do so, we model one PUC as a set of two-input, two-output vertices (corresponding to its input and output optical ports) and four edges reproducing the device's internal connections. Other works [83], published in parallel of to this Thesis, propose alternative graph representations of a PUC using auxiliary internal nodes with negative weights or even model the PUCs themselves as standalone nodes. This last case, however, would not be applicable to meshes with feedback loops. In any case, these structures can be arranged to mirror the original mesh topology, as illustrated in Figure 3.1.

To develop, apply and illustrate the performance of our algorithm, we need to define and name our graph nodes and edges. To do so, we denote every graph vertex (optical

connection) by a set of two indices: a first one referring to the closed polygon to which it belongs inside the mesh and a second one used to identify its position inside of it (oriented clockwise). To identify all input/output optical ports at the mesh perimeter, we define an additional set of 'imaginary' cells surrounding the original structure. In all cases (actual and imaginary, separately) cell numeration follows the same criteria: from top to bottom and from left to right.

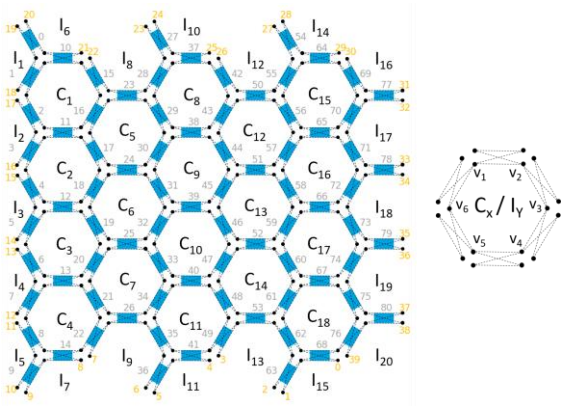


Figure 3.1. Joint portrayal of an 81-PUC waveguide mesh along with its graph representation. We denote each of its vertices by referring to its constituting cell index ('Cx/Iy', both numbered separately from top to bottom and from left to right) and to its position inside of it, numbered clockwise. Some examples: upper left port from PUC 51 would be referred to as '$C_{12}v_5$', while ports 27 and 28 of the waveguide mesh would be '$I_{12}v_2$' and '$I_{14}v_6$'.

Next, we must define a weight or *transmission distance* (TD) for every edge connecting two optical nodes in the arrangement. As we will see through the following subsections, we can define such distance according to several figures of merit, for instance:

- The number of traversed PUCs through the whole mesh arrangement. In such case, all edges' TD would share a value of 1.
- The accumulated delay time taken by an optical signal to traverse all the PUCs arrangement.
- Insertion loss.
- Power consumption.
- etc.

Moreover, nothing would prevent us -using appropriate normalization rules and coefficients- to come up with new, ad-hoc TDs formed by the addition of two or more figures of merit.

3.2.2 Algorithm description

As we introduced, the fundamentals of shortest-path evaluations consist of searching the shortest route between two nodes through a weighted graph with the purpose of finding the route that accumulates the least weight. After having defined such weights (or TDs) for every edge (optical connection) inside the graph, a shortest path with the input node as root propagates through the remaining ones by accumulating each TD prior to reach destination port, as in classical tree-search algorithm implementations. However, the proposed graph search includes a couple of additional constraints, addressed to physical violations:

- Light cannot propagate through the same PUC twice consecutively, as this is not physically possible without being recirculated by an external element.
- Paths are discarded if force any of its constituting PUCs to be simultaneously in bar/cross transmission states (not TC) during the synthesis of any specific optical connection (in fact, this statement is a generalization of previous one, since the only way for a path to propagate the same PUC twice consecutively is to do so under two different transmission states).

Figure 3.2 illustrates these prohibitions with a bunch of examples. Red edges represent an impossible path, as going from graph node $I_{10}v_5$ to C_8v_1 through $I_{10}v_4$ would go against both constraints specified above in PUC 37 (see Figure 3.1). Instead, the shortest path between both edges would be the one appearing in green. It is allowed, however, to re-use PUCs (i.e., to traverse them multiple times) it they remain in the same transmission state. Such is the case of the path in navy blue synthesized between ports $I_{13}v_4$ and $I_{15}v_2$, in which the PUC between cells I_{15} and C_{18} is set to cross state and crossed twice, from node $I_{15}v_1$ to $C_{18}v_4$ and from $C_{18}v_5$ to $I_{15}v_2$ (or the other way around). At the same time, the remaining colored paths at bottom left side of the figure reveal other possible paths provided by the algorithm between ports I_4v_2 and $I_{11}v_6$.

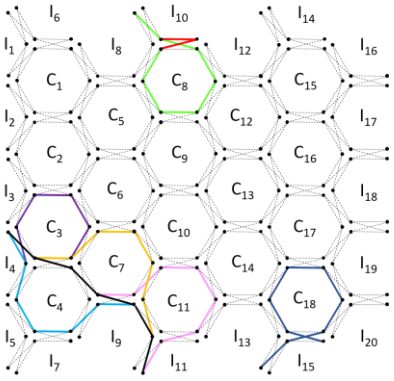


Figure 3.2. Synthesis of several (valid and invalid) optical paths within the graph representation of an 81-PUC waveguide mesh.

The list of available mechanisms to explore all possible paths through the waveguide mesh is very extensive [84]. In this work, we will employ Breadth-First Search (BFS) algorithm, consisting of exploring all nodes at given depth prior to moving on to the nodes at next depth level. Child nodes that are encountered but not yet explored are stored in the form of a queue. However, this algorithm can be slow compared to other alternatives such as Depth-First Search (DFS) algorithm for deeper graphs in which destination nodes lie far from the origin.

The proposed method should avoid a brute-force search of every possibility, ensuring scalability. Therefore, we propose in this work the use of bidirectional search algorithm, in which both origin and destination nodes are used as input/output of our optical paths. At each step, our search algorithm would propagate from both graph nodes, ultimately forming a path any time each pair of branches crosses. Such approach, of complexity $O(b^{d/2} + b^{d/2})$ –being b the branching factor[1] and d the distance between source and destination nodes– can reduce dramatically the time required by the algorithm for the obtention of paths using BFS (of complexity $O(b^d)$).

Figure 3.3 illustrates a case example using a photonic processor. There, we aim to find all possible optical paths between ports 37 and 4 in a 198-PUC waveguide mesh. These nodes lie at 24 PUCs at least and, consequently, any usual BFS evaluation technique would need such number of steps to supply the first results (paths). If the mesh from the figure grew vertically, more candidate paths would propagate in such

[1] In computing, tree data structures and game theory, the branching factor is the number of children at each node. If this value is not uniform, an average branching factor can be calculated.

direction during the execution of the algorithm, enlarging the elapsed time significantly. By contrast, using bidirectional search would allow us to find our first paths at step 12, helping us skipping those PUCs that are far enough from destination.

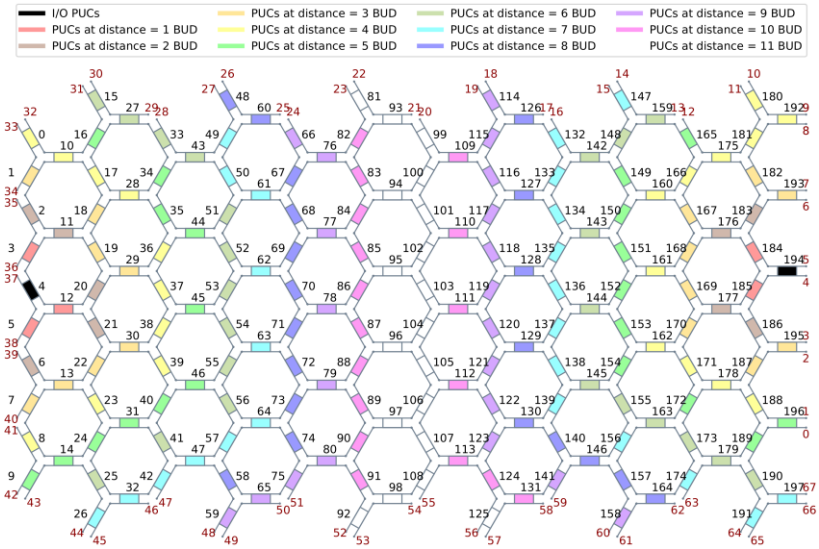


Figure 3.3. Use of pathfinder algorithm (based on bidirectional search) in a 198-PUC waveguide mesh to provide all possible paths between optical ports 4 and 37.

3.3 Applications

3.3.1 Self-calibration of photonic waveguide meshes

We discussed back in previous chapters about the variability introduced by fabrication process in photonic integrated circuits, and how it affects to some PUC key parameters such as its insertion loss or passive phase (i.e., in absence of applied bias). In this subsection, we introduce a software protocol that provides us the passive state of all PUCs in the waveguide mesh as well as the calibration curves of each of their phase actuators, which offer us information about the variation of every PUC coupling factors with the supplied current via thermo-optic effect. Note how this approach does not require any additional photonic elements acting as reference such as the one recently proposed in [85], reducing the design complexity and yield.

A huge advantage of this method is that it only requires one input optical port to be run. Consequently, it is free from any manual operation aside from optical alignment, hence becoming a fully automatic process. The number of output ports required is variable. In meshes with access PUCs (such as the one from this example), we shall need at least one extra port by access PUC to calibrate them.

To kick off this algorithm, we need to get the graph representation of our waveguide mesh such as we did in previous subsection. After this, we set one of our mesh ports as the input node and run the pathfinder algorithm described before between such node and its adjacent one, regarded as output. With this routine, we accumulate as many optical paths as possible in a given timeframe (set to 25 seconds in our experiment) going through the graph representation of our mesh. Each of these paths contains information about the traversed PUCs to connect both nodes, as well as their respective transmission states (bar or cross). When we are done, we save all these paths inside a list before moving to next output port.

Our algorithm iterates then over this list of paths, taking the following steps:

1. For a specific path from the list, we check whether there are any remaining optical phase actuators to be calibrated. If that is not the case, we choose then the following path from the list. Otherwise, we jump into the next stage of the algorithm, involving path power optimization (such as we explained in previous chapter) and reduction of optical leakage.
2. We locate all surrounding PUCs to our optical path at a given distance (set to 1 in our examples). We label those PUCs as neighbor PUCs, which we will later use to prevent optical power leakage from coming through the path. If any of these PUCs has been already calibrated (both phase shifters) in previous iterations of the algorithm, we can directly set to cross state (the reason behind doing this will become clearer to the reader at the end of the algorithm description).
3. Then, we perform a power maximization using the phase shifters from our optical path. Such as in previous step, if we have any PUC whose phase shifters have already been previously calibrated, we can directly set it to either bar or cross state as it may correspond, hence reducing the variable space of our optimization algorithm and therefore easing this task in great deal.
4. After this power maximization, we tune each PUC of the constituting path to reduce by 2 dB for each the overall measured optical power. Then, we carry out a power minimization between the same input/output optical ports using neighbor PUCs phase shifters. We do this initial reduction prior to power maximization to avoid the optimization algorithm to get stuck from the beginning, since previous path maximization from step 3 may result in no optical power going through neighbor PUCs at all.

Commented [DPL16]: Ser

5. Coming after leakage reduction takes place a second power maximization using again path PUCs. This maximization uses the results from first power maximization as seed, meaning that the resulting retrieved power will be equal than then, at least. This path power optimization process is represented in Figure 3.4 and Figure 3.5.

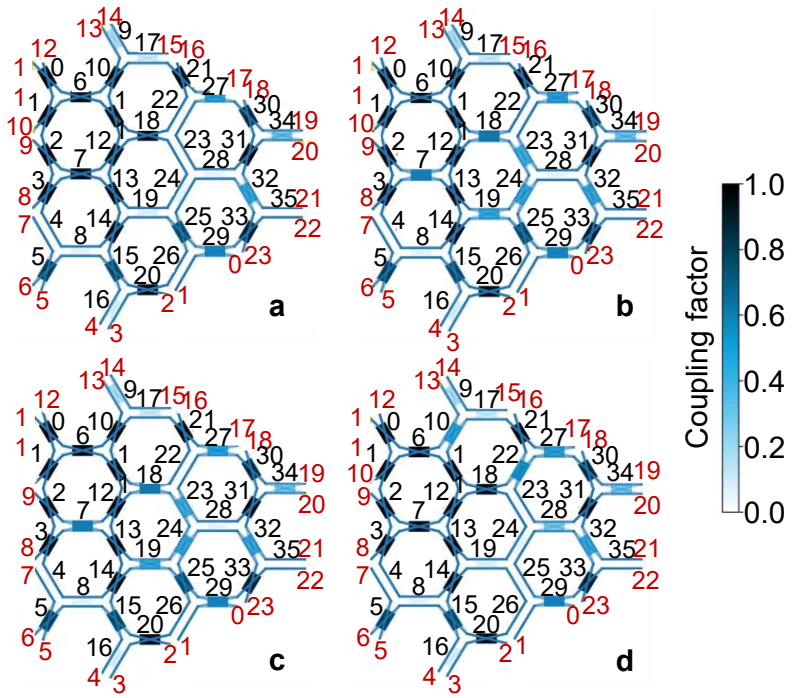


Figure 3.4. Path optimization cycle for the synthesis of path 3 from Table 3-I (PUCs 0, 6, 11, 18, 23, 24, 19, 13, 7 and 2). (a) First path maximization. (b) 2-dB reduction of path PUCs to favor leakage minimization. (c) Leakage minimization using neighbor PUCs at distance 1 (PUCs 1, 3, 10, 12, 14, 22, 25 and 28). (d) Second path maximization.

Commented [DPL17]: Fal

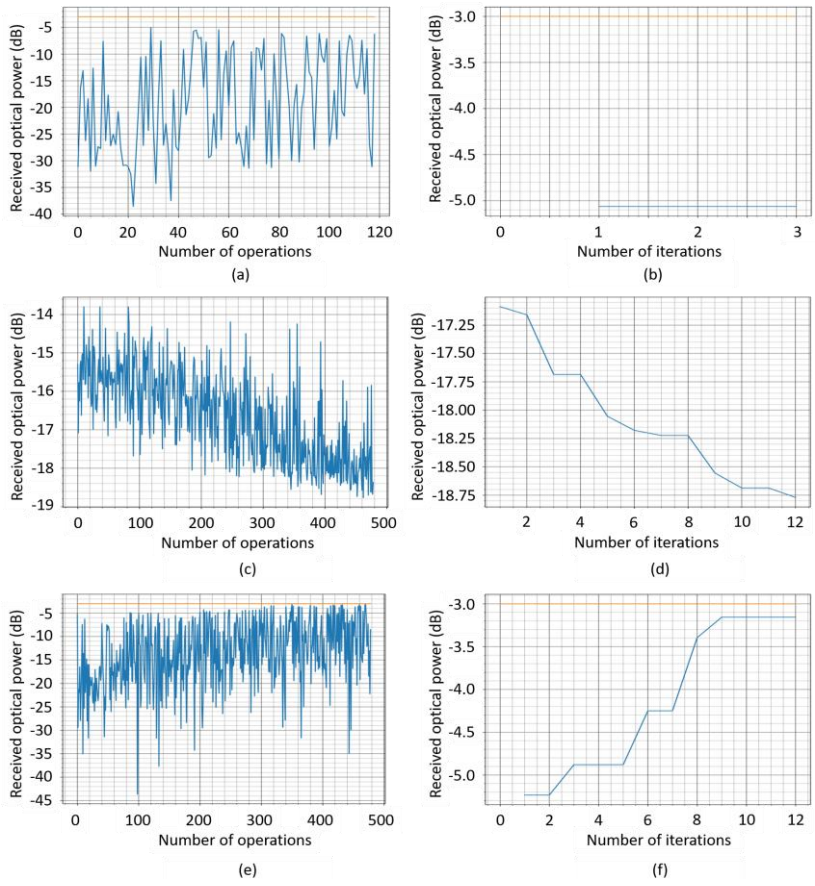


Figure 3.5. Variation of the received optical power with the number of operations (a, c, e) and iterations (b, d, f) between ports 11 and 9 in an 36-PUC waveguide mesh during path 4 optimization from Table 3-I. (a, b) First path maximization, (c, d) leakage minimization, (e, f) second path maximization.

6. Steps 3, 4 and 5 assist us assuring the path found by our algorithm (and no other) has been successfully maximized. We can proceed now to calibrate each PUC phase shifter individually. To do so, we start from first path PUC and set one of its two phase shifters to zero mA. We then perform a fine current sweep to the other one (maintaining the rest of the path in its optimum configuration) and observe its impact on overall power response as observed in Figure 3.6.

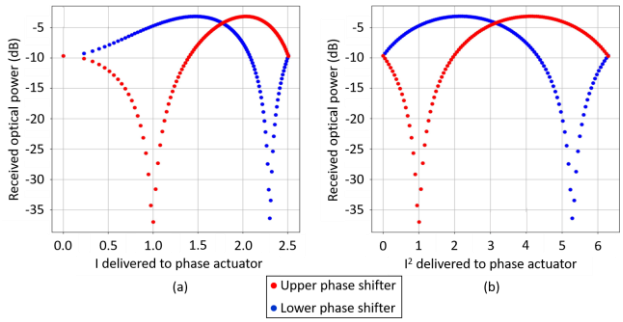


Figure 3.6. PUC 23 upper and lower phase actuator current (a) and current square sweeps after path 4 optimization.

7. After doing this, we are ready to move to the next path from our list. Should we run out of them, we move to the next adjacent output port, re-run pathfinder algorithm to accumulate a new batch of paths and start over.
8. Eventually, we will have at our disposal all phase shifters calibration curves from our mesh. The last two steps of our self-calibration algorithm involve the curve fitting of our experimental results to smooth our prediction as depicted in Figure 3.7 (a) and a normalization and interpolation of these results to provide us the curve showed in Figure 3.7 (b), featuring the variation of the PUC coupling factor with the current difference between both phase shifters. With these curves, we have the information to set any arbitrary coupling factor to every mesh PUC. Note how these steps also provide us information about the passive state of each PUC, a very powerful asset to reduce power consumption in many applications such as [66].

Commented [ALH18]: Do information here?

Commented [DPL19R18]: trabajo realizado.

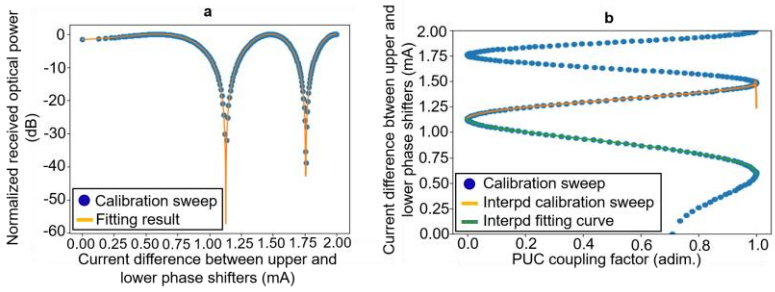


Figure 3.7 (a) Curve fitting of the experimental results of PUC 15 upper phase shifter. (b) Normalization and interpolation of the results in (a) to produce the PUC coupling factor evolution with the current difference between both PUC phase shifters. Orange interpolation curve uses the experimental results, while green interpolation curve does so

from the fitting curve. Due to the smoothness of this latter one, interpolation results are much less noisy.

This tool appears described in the form of pseudocode in Annex A. Table 3-I shows the simulated results in a 36-PUC waveguide mesh, in which all PUCs had been successfully calibrated using 14 paths and 7 output ports only.

Table 3-I. Summary of the auto-calibration results of a 36-PUC waveguide mesh.

Index	Output port	Path	Calibrated PUCs from this path	Remaining PUCs to calibrate
1	10	0, 1	0, 1	34
2	9	0, 6, 11, 12, 7, 2	2, 6, 11	31
3	9	0, 6, 11, 12, 13, 14, 8, 4, 3, 2	3, 4, 8, 12, 13, 14	25
4	9	0, 6, 11, 18, 23, 24, 19, 13, 7, 2	7, 18, 19, 23	21
5	9	0, 6, 11, 18, 23, 24, 19, 14, 8, 4, 3, 2	24	20
6	9	0, 6, 10, 17, 21, 22, 18, 12, 13, 14, 8, 4, 3, 2	10, 17, 21, 22	16
7	9	0, 6, 11, 18, 23, 24, 25, 26, 20, 15, 8, 4, 3, 2	15, 20, 25, 26	12
8	9	0, 6, 11, 18, 22, 27, 30, 31, 28, 24, 19, 13, 7, 2	27, 28, 30, 31	8
9	9	0, 6, 11, 18, 23, 28, 32, 33, 29, 25, 19, 13, 7, 2	29, 32, 33	5
10	6	0, 1, 2, 3, 4, 5	5	4
11	13	0, 6, 10, 9	9	3
12	4	0, 1, 2, 3, 4, 5, 8, 15, 16	16	2
13	19	0, 6, 10, 17, 21, 27, 30, 34	34	1
14	21	0, 6, 11, 18, 23, 28, 32, 35	35	0

Figure 3.8 represents the synthesis of four of the optical paths presented in Table 3-I. Starting between ports 11 (input) and 10 (output), it can be observed how more PUCs are progressively being calibrated during the execution of the algorithm. The routine

finishes with PUCs 5, 9, 16, 34 and 35, which are used as access to the waveguide mesh and hence require switching output port before retrieving paths back again. We can also check from Figure 3.8 (b) how PUC 7 was not fully calibrated at first time (appearing in path 2), but the algorithm keeps running until achieving so at second attempt.

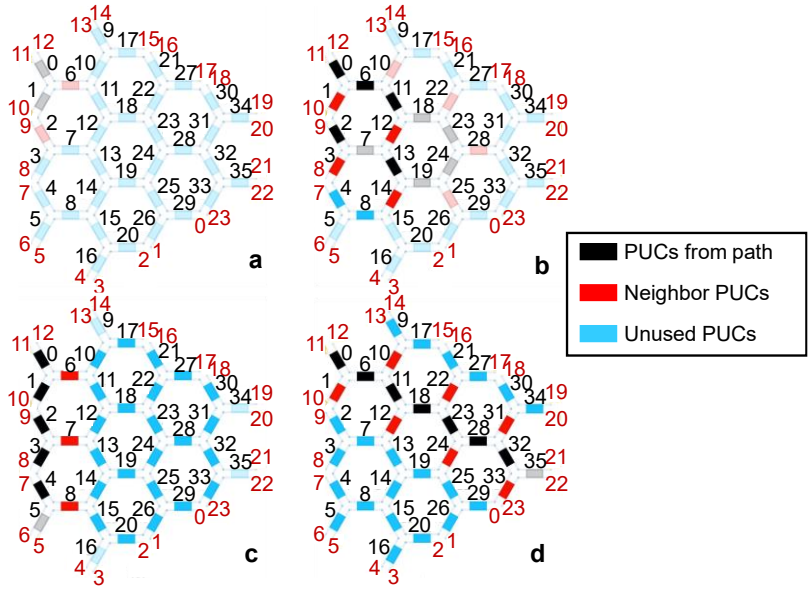


Figure 3.8. Synthesis of paths 1 (a), 3 (b), 8 (c) and 15 (d) from Table 3-I. More saturated colors correspond to PUCs calibrated (both phase shifters) from previous paths.

To study the impact of leakage in our model, we can consider the synthesis of a long optical path in our waveguide mesh going between ports 11 and 9 and traversing through PUCs 0, 6, 11, 18, 23, 24, 19, 13, 7 and 2. If the passive state of PUC 12 is close to bar, we take the risk of synthesizing by accident the path going through PUCs 0, 6, 11, 12, 7 and 2 during power maximization, since it is expected to feature a lower accumulated insertion loss as it traverses through a lower number of PUCs. Note that, in such case, PUCs 11 and 7 would no longer be operating in cross state, but in bar one. Consequently, our software tool would wrongly assume bar states as the ones that maximize the power rather than cross ones while performing the current sweep illustrated in Figure 3.6 (recall that, while storing each path, the algorithm

keeps information about the transmission state at which each individual PUC should be).

There is one important restriction on which paths are eligible to run this algorithm: we must discard those including re-used PUCs. To cite an example, if we observe Figure 3.8 back again, we note how the only available path between ports 11 and 10 is the one using PUCs 0 and 1. Any other one, such as for instance the one going through PUCs 0, 6, 11, 18, 22, 21, 17, 10, 6 and 1 would fall out from this category (as it reuses PUC 8). Also, because of this, we cannot find any available path between ports 11 and 12. The reason is simple: if we pay attention to any of such paths, we can observe how their accumulated insertion loss does not vary if the re-used PUC changes its transmission state from cross to bar (or vice-versa). Consequently, we would not be able again to identify such states with clarity while performing the current sweep of its individual phase shifters as we showed in Figure 3.6.

3.3.2 Self-characterization of photonic waveguide meshes

3.3.2.1. Self-characterization of optical ports insertion losses

In this subsection, we will deal with the obtention of the insertion losses of all optical ports surrounding the waveguide mesh. This tool provides useful information about whether any I/O access (fiber to chip or access path) to the mesh is damaged, hence preventing us from using it from then on.

To do so, we proceed as follows:

1. Starting from the same graph representation of our waveguide mesh created for self-calibration routine, this time we set one fixed port as input and run again our pathfinder tool iteratively between such port and regarding the remaining optical ports as outputs during a set timeframe. For a mesh with N optical ports, this provides us $N-1$ groups of paths belonging to all these combinations between set input port and outputs.
2. Next, we measure the received power for each of these retrieved paths. We can synthesize them by setting their respective constituent PUCs to bar/cross state using the calibration results from self-calibration routine. Note how, unlike then, this time we can reuse PUCs to form additional paths.
3. Once done, we perform a linear regression for each pair of input-output ports using all their accumulated paths. This can be observed in Figure 3.9 and Table 3-II. The intercept of the resulting regression lines will correspond to the combined coupling loss between input and output ports. By doing this, we obtain $N-1$ intercept values. We may store all their slopes for future use in next subsection.

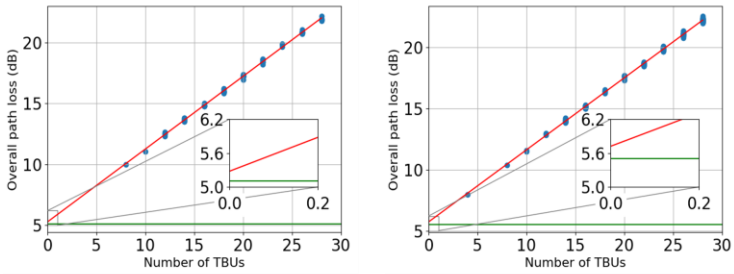


Figure 3.9. Linear fittings (in red) to estimate the accumulated coupling losses between two different pairs of optical ports in a simulated 36-PUC photonic processor. The straight, green line represents the real (simulated) coupling loss of the device. Figure insets provide a zoomed overview of the fitting accuracy.

Commented [DPL20]: Esp
Menciona que es el inset en el
Menciona que es la linea verde

Commented [ALH21R20]

Table 3-II. Summary of paths retrieved between optical ports 11 and 12 in a simulated 36-PUC photonic processor, along with their respective measured optical powers.

Path Index	Path	Traversed PUCs	Path IL (incl. ports IL)
1	I_1v_2, I_4v_5, C_1v_2, C_1v_3, C_1v_4, C_1v_5, C_1v_6, I_1v_3, I_4v_6	0, 6, 11, 12, 7, 2, 1, 0	10.46
2	I_1v_2, I_4v_5, C_1v_2, C_3v_5, C_4v_2, C_4v_3, C_4v_4, C_4v_5, C_2v_2, C_1v_5, C_1v_6, I_1v_3, I_4v_6	0, 6, 11, 18, 23, 24, 19, 13 7, 2, 1, 0	12.79
3	I_1v_2, I_4v_5, C_1v_2, C_1v_3, C_4v_6, C_2v_3, C_2v_4, C_2v_5, C_2v_6, I_2v_3, C_1v_6, I_1v_3, I_4v_6	0, 6, 11, 12, 13, 14, 8, 4, 3, 2, 1, 0	12.84
…			
41	I_1v_2, I_4v_5, I_4v_4, C_3v_1, C_3v_2, C_3v_3, C_3v_4, C_3v_5, C_3v_6, C_3v_1, C_3v_2, I_8v_5, C_6v_2, C_6v_3, C_6v_4, C_7v_1, C_4v_4, C_5v_1, C_2v_4, C_2v_5, C_2v_6, I_2v_3, C_1v_6, I_1v_3, I_4v_6	0, 6, 10, 17, 21, 22, 18, 11, 10, 17, 21, 27, 30, 31, 28, 24, 19, 14, 8, 4, 3, 2, 1, 0	20.31
42	I_1v_2, I_4v_5, I_4v_4, C_3v_1, C_3v_2, C_3v_3, C_3v_4, C_4v_1, C_4v_6, C_4v_5, C_5v_2, C_7v_5, C_7v_4, C_7v_3, C_7v_2, C_7v_1, C_4v_4, C_5v_1, C_2v_4, C_2v_5, C_2v_6, I_2v_3, C_1v_6, I_1v_3, I_4v_6	0, 6, 10, 17, 21, 22, 18, 12, 13, 19, 25, 29, 33, 32, 28, 24, 19, 14, 8, 4, 3, 2, 1, 0	20.11

43	I_1V_2, I_4V_5, I_4V_4, C_3V_1, C_3V_2, C_3V_3, C_3V_4, C_4V_1, C_4V_6, C_4V_5, C_5V_2, C_5V_3, C_5V_4, C_5V_5, I_5V_2, C_2V_5, C_2V_6, I_2V_3, C_1V_6, I_1V_3, I_4V_6	0, 6, 10, 17, 21, 22, 18, 12, 13, 19, 25, 26, 20, 15, 8, 4, 3, 2, 1, 0	17.65

4. To estimate the insertion loss values of the N mesh ports without changing the input port, we shall need an extra relation between any pair of ports. To do so, we get the median value M of the $N-1$ intercepts (or the closest one for an odd value of N). It will correspond to a specific pair of ports, say (in, out_1). If we now find the closest intercept value to M (excluding it) corresponding to ports (in, out_2), say M', we can then retrieve an extra relation between ports out_1 and out_2 through the mean value between M and M'.
5. After this, we can now solve the system of N equations and obtain the coupling loss per optical port. Such system can be represented by the matrix appearing in equation (6), of dimension $N \times N$. As explained before, each of this matrix's columns represents an optical port, while each row is representing a different intercept. At the same time, all intercept values retrieved (those coming from the linear regressions performed in step 3 plus the extra relation deduced in step 4) can be arranged in a $N \times 1$ column vector.

Commented [DPL22]: En
ports

$$H_{i/o\ ports} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & \dots & 0 \\ 1 & 0 & 1 & 0 & 0 & \dots & 0 \\ 1 & 0 & 0 & 1 & 0 & \dots & 0 \\ & & & \dots & & & \\ 1 & 0 & 0 & 0 & 0 & \dots & 1 \\ 0 & 1 & 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & 1 & & \dots & 0 \\ & & & \dots & & & \\ 0 & 1 & 0 & 0 & 0 & \dots & 1 \\ & & & \dots & & & \\ 0 & 0 & 0 & 0 & \dots & 1 & 1 \end{bmatrix} \quad (2)$$

6. The error in the estimation of the coupling losses will propagate to the estimation of the PUCs insertion loss. To avoid this, we proceed to isolate suspiciously characterized ports (i.e., those which would require a second self-characterization round). These are those featuring an estimated insertion loss twice larger than median.

Results from Figure 3.10 reveal an interesting observation: the error in the estimation of ports IL does not vary with their absolute value, but with the variation of the mesh PUCs insertion loss variance. And the reason behind this can be deduced from the fittings shown in Figure 3.9. Looking at them, we can infer how a larger variance in the PUC IL would result in having 'more spread' scatter points representing our measured path optical powers, resulting in a worse fitting. Indeed, if all PUC ILs were the same, the fitting would be perfect and therefore the intercept would match the accumulated coupling loss between input and output ports exactly. Note how changing the concrete coupling loss value of any individual port would not have any effect in their estimation, as such change would affect all paths equally and thus the regression line slope would not be modified. That explains the aspect from the contour plots in Figure 3.10 (a) arranged in horizontal frames. If we were able to use all ports as inputs rather than figuring out a way to estimate the last row of $H_{i/o\ ports}$ through the intercepts median, that would be also the case of Figure 3.10 (b) for the estimation of port insertion losses.

Commented [ALH23]: @

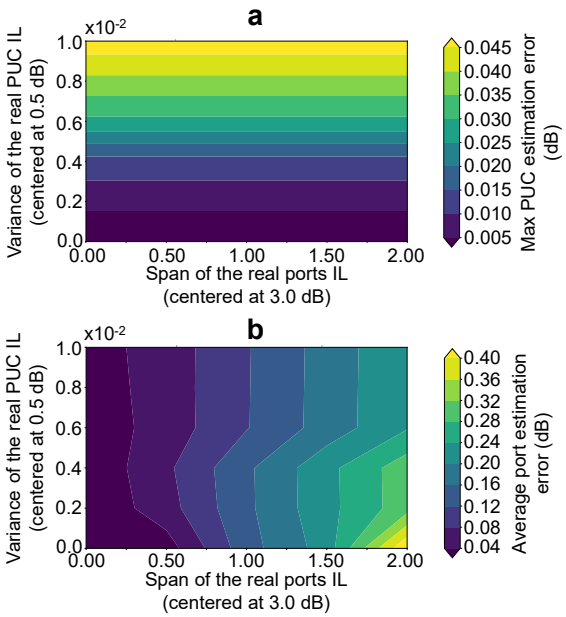


Figure 3.10. Simulated self-characerization of several 36-PUC photonic processors with different PUC IL variances and port IL spans featuring an average IL of 0.5 dB. (a) Contour plot of the PUCs IL estimation errors, (b) contour plot of the ports IL estimation errors.

3.3.2.2. Self-characterization of PUC insertion losses

Once we have been provided with the insertion losses of the optical ports surrounding the structure, next comes the obtention of the insertion losses of the PUCs constituting it themselves. This can be extremely useful, for instance, in case that any of such devices suffers from any malfunction. In such case, pathfinder algorithm would improve its efficiency by discarding any path/s traversing any of these components in future uses.

The process relies on fundamental algebra operations. To do so, we build a second system of equations represented by the matrix H_{PUCs} from equation 7, formed by all paths retrieved during previous process. This matrix is therefore of dimension $P' \times p$, where P' represents the overall number of retrieved paths using all output ports (excluding the ones discarded in step 6 of output ports self-characterization process) and p is the number of mesh PUCs. Therefore, the non-zero elements of each row of H_{PUCs} indicate the PUCs taking part of that specific path (being equal to 1 if the PUC is traversed once only and 2 if it is reused).

$$H_{PUCs} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & \dots & 0 \\ 1 & 0 & 1 & 0 & 0 & \dots & 0 \\ 1 & 0 & 0 & 1 & 0 & \dots & 0 \\ & & & \dots & & & \\ 1 & 0 & 0 & 0 & 0 & \dots & 1 \\ 0 & 1 & 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & 1 & & \dots & 0 \\ & & & \dots & & & \\ 0 & 1 & 0 & 0 & 0 & \dots & 1 \\ & & & \dots & & & \\ 0 & 0 & 0 & 0 & \dots & 1 & 1 \end{bmatrix} \quad (7)$$

At the same time, all measured path insertion losses (minus the corresponding coupling loss estimated by input/output optical ports obtained in previous subsection) can be arranged in a column vector of dimension $P' \times 1$. Note that the accuracy in the estimation of PUCs insertion losses will be therefore strongly dependent on our previous estimation of the optical ports' losses. A poor estimation of the coupling losses will result on a wrong calculation of IL_{paths}'s components, leading us to a wrong characterization of our PUCs insertion losses.

After solving this system of equations, we may now list and isolate 'suspiciously characterized PUCs' such as we did for the optical ports in previous subsection. PUCs in need of re-characterization will be those featuring an estimated insertion loss

falling outside the interval [average of all linear fitting slopes – 0.2, average of all linear fitting slopes + 0.2], where we are using the regression line slopes stored after step 3 from previous subsection.

The following figures illustrate several application scenarios in which we use our self-characterization tool to unveil the insertion losses of the PUCs and the optical ports surrounding a 36-PUC waveguide mesh. We start from the case from Figure 3.11 (a), in which the insertion losses of all programmable unit cells have been modelled through gaussian random variables centered at 0.4 dB and featuring variances of 0.05 dB. The insertion loss of PUC 15 has been set to 20 dB, thus corresponding to a PUC featuring bad functioning (attenuating light by a factor of 100 as it crosses it). At the same time, optical ports were modelled through uniform random variables centered at 2.75 dB and a span of 2 dB.

As previously detailed, first would come the estimation of the optical ports' insertion loss, illustrated in Figure 3.11 (b). And here we see something curious -as all optical paths connecting ports 0 and 2 must necessarily come across PUC 15, the linear regression corresponding to such pair of ports provides us a very large (and wrong) value for the IL of port 2, as observed in the figure. Such error makes perfect sense and is related to the position of PUC 15 in the waveguide mesh.

As this insertion loss is more than twice as large as the median of the remaining estimated ones, we mark our estimated IL value for port 2 as 'suspicious' and remove all paths (rows) reaching it from H_{PUCs} matrix, solving the resulting system of equations using all remaining ones (Figure 3.11 (c)). With this correct estimation of PUC 15 IL, we are now able to re-estimate the insertion loss of port 2, reaching to the results presented in Figure 3.11 (d) and confirming there was nothing wrong about it from start.

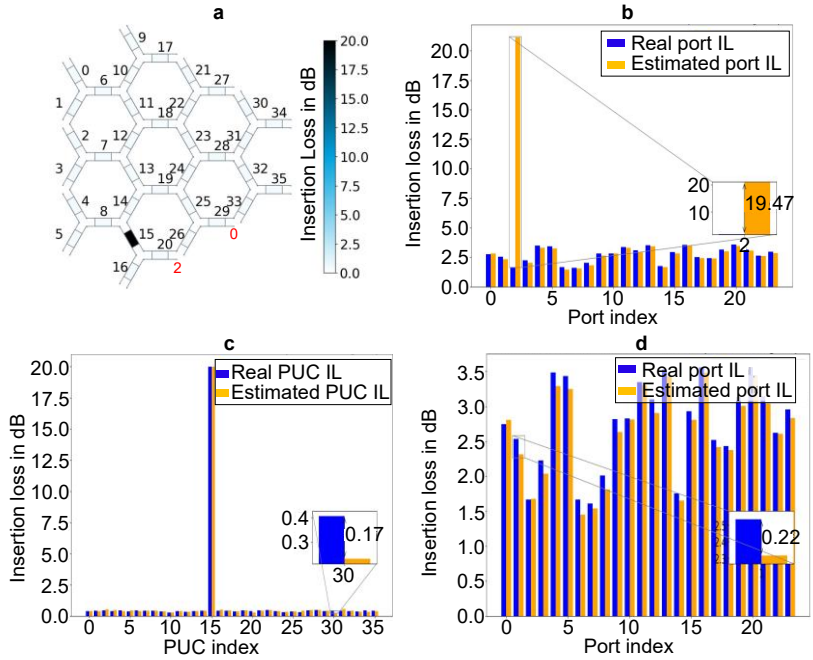


Figure 3.11 (a) Schematic of our 36-PUC simulated waveguide mesh, whose unit cells IL are modelled by a gaussian random variable centered at 0.4 dB and featuring a 0.05 dB variance. PUC 15 IL is set to 20 dB, (b) comparison between estimated and real optical ports IL after first self-characterization round, (c) comparison between estimated and real PUCs IL after first self-characterization round, (d) comparison between estimated and real optical ports IL after second self-characterization round.

As explained, the need of going through this second self-characterization round for the optical ports was given by the specific position of PUC 15 inside the mesh. The exact same issue would arise, for instance, with PUC 10 and optical port 15. But there are other 'more benevolent' cases, such as the one depicted in Figure 3.12, for which this second round would no longer be needed. Here it is PUC 18 the one suffering a malfunctioning of 20 dB again, while the remaining ones are operating correctly. But in this case, as can be observed in Figure 3.12 (a), there are many alternatives to connect port 0 to all remaining ones without traversing PUC 18. Hence, during linear regression, all paths going through such PUC would be treated as outliers and discarded from the process and, as a result, all optical ports IL would be correctly estimated at first as shown in Figure 3.12 (b).

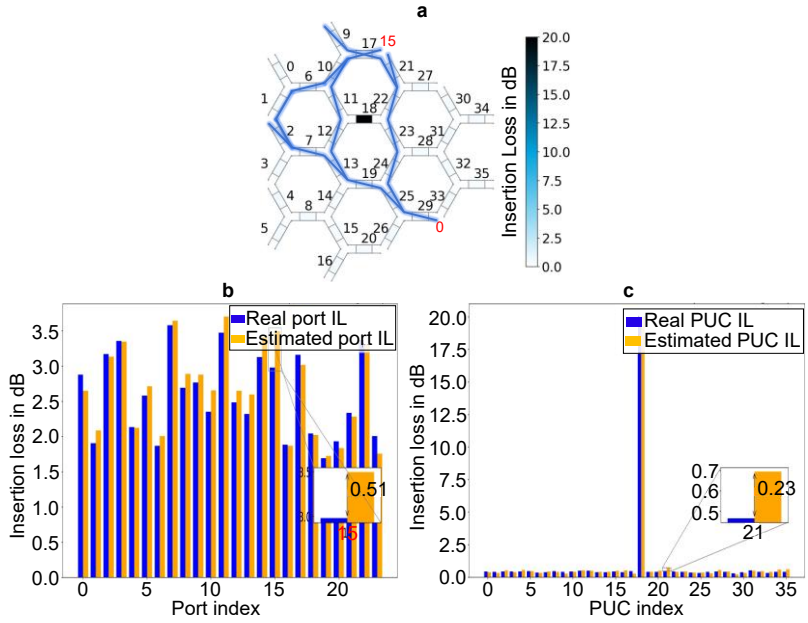


Figure 3.12 (a) Schematic of our 36-PUC simulated waveguide mesh, whose unit cells IL are modelled by a gaussian random variable centered at 0.4 dB and featuring a 0.05 dB variance. PUC 18 IL is set to 20 dB, (b) comparison between estimated and real optical ports IL after first self-characterization round, (c) comparison between estimated and real PUCs IL after first self-characterization round, (d) comparison between estimated and real optical ports IL after second self-characterization round.

Figure 3.13 represents two previous scenarios combined, with now both PUCs 15 and 18 featuring 20 dB of insertion loss. As can be observed, our tool deals with this new scenario treating both cases separately. During first PUC IL self-characterization round, as detailed in Figure 3.13 (c), both malfunctioning PUCs are successfully identified without any major issue and during next step (Figure 3.13 (d)) it deals with the re-estimation of port 2 insertion loss such as it also did in Figure 3.11 (d).

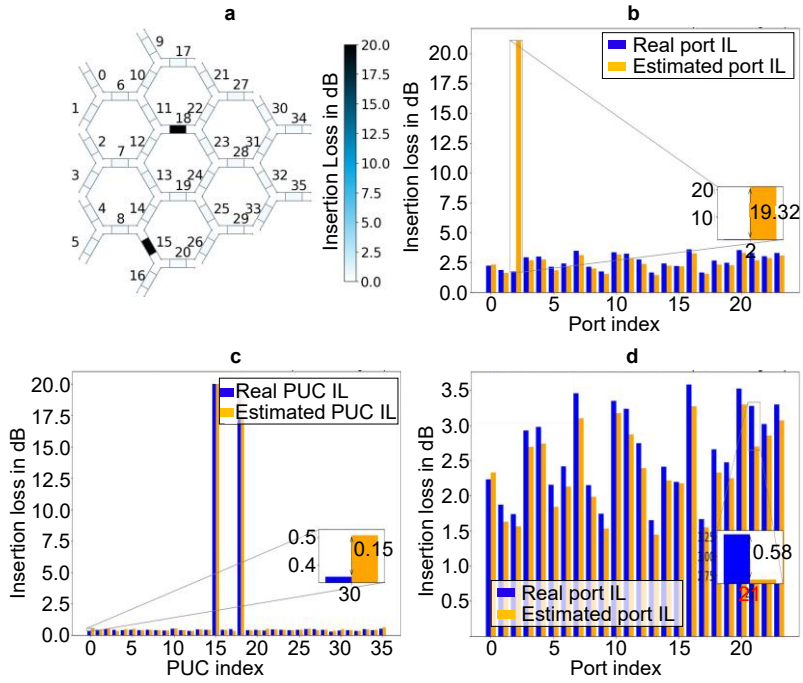


Figure 3.13 (a) Schematic of our 36-PUC simulated waveguide mesh, whose unit cells IL are modelled by a gaussian random variable centered at 0.4 dB and featuring a 0.05 dB variance. PUCs 15 and 18 insertion losses are set to 20 dB, (b) comparison between estimated and real optical ports IL after first self-characterization round, (c) comparison between estimated and real PUCs IL after first self-characterization round, (d) comparison between estimated and real optical ports IL after second self-characterization round.

Finally, we deal with the scenario depicted in Figure 3.14 (a) in which both adjacent PUCs 15 and 16 are seemingly broken. If we now pay attention to the first optical ports' self-characterization results appearing in Figure 3.14 (b), we can see how optical ports 2, 3 and 4 have been marked as broken by our algorithm. This should not come as a surprise after having observed previous examples -port 2 is again involved because of the malfunctioning in PUC 15, while ports 3 and 4 are the ones attached to defective PUC 16.

We observe in Figure 3.14 (c) how this issue is translated into a (pretty much) wrong estimation of the IL from PUCs 15, 16 and 20. But one question we can ask to ourselves is: does this matter? As PUC 16 is effectively down, there is no use about

using ports 3 or 4 since losses would be high in both cases anyway. The same applies when it comes to port 2 or PUC 20. There is no way to use any of them efficiently without traversing any of the damaged components. Even if the algorithm is not telling us with accuracy what elements are wrong in this scenario, it works at unveiling the problem at such section of the mesh, helping us discarding it for future use.

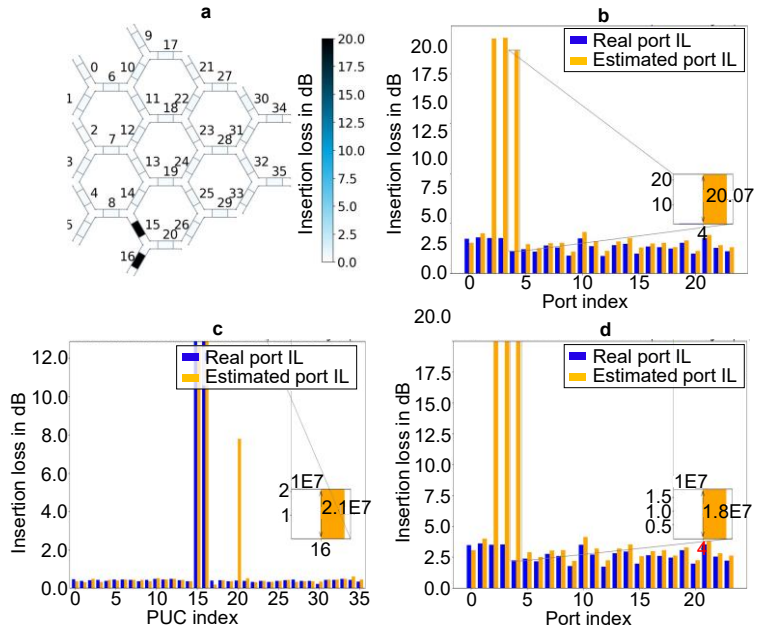


Figure 3.14 (a) Schematic of our 36-PUC simulated waveguide mesh, whose unit cells IL are modelled by a gaussian random variable centered at 0.4 dB and featuring a 0.05 dB variance. PUCs 15 and 16 insertion losses are set to 20 dB, (b) comparison between estimated and real optical ports IL after first self-characterization round, (c) comparison between estimated and real PUCs IL after first self-characterization round, (d) comparison between estimated and real optical ports IL after second self-characterization round.

3.3.3 Simulation of photonic circuits

Pathfinder algorithm can be also used to simulate photonic circuits (predicting their spectral responses) in a waveguide mesh with an arbitrary number of coupled cells. This can be helpful to reduce development costs associated to the design cycle and

validation of complex systems (usually costly, risky, and time-consuming), allowing the rapid development of new circuit designs.

To do so, this time we define the transmission distance of each graph edge as the spectral response of the PUCs at the central wavelength. Thus, we can estimate the overall accumulated cost of any path traversing the graph by directly multiplying their spectral responses at such wavelength. Adding up the accumulated transmission distances of all remaining paths will provide us a good approximation of the spectral response between both connections.

We can accelerate this process by defining a power threshold (in amplitude) rather than using bidirectional search as in previous examples. In such a way, any path featuring an optical power at central wavelength below this limit will be discarded, and therefore we prevent that branch from extending further through the mesh. This can be particularly useful for large arrangements, as only a few branches will remain after a few iterations, thus reducing the computation time of the algorithm significantly. However, we must choose this limit carefully. Figure 3.15 shows the dependency of both simulation error and elapsed computational time on it. To create it, we configured a set of programmable photonic processors with different sizes (134, 198, 397 and 599 PUCs) to emulate a Ring-Assisted MZI filter (RAMZI). Then, we calculated the frequency response of the system with the inductive method presented in [77] and our graph-based approach. For the latter, we employed four different threshold levels of -20, -30, -40 and -50 dB. Since the frequency response of the inductive method does not employ any hyperparameter, it is considered as the error-free trace.

Figure 3.15 (a) shows the comparison of the frequency responses of the synthesized RAMZI from our simulations and compare it with an alternative methodology based on a mathematical inductive method [77], [86] . We can observe that the fitting of the inductive method and the graph-based approach decreases for lower values of threshold. Similarly, Figure 3.15 (b) displays the mean elapsed times of the computation and the obtained minimum square error for the different thresholds. For example, we can observe how a medium-size mesh such as the 397-PUC one features a mean elapsed time of 5.93, 9.44, 13.57 and 16.84 seconds with associated mean square errors of 4×10^{-3}, 5×10^{-4}, 8×10^{-5} and 5×10^{-6} respectively for the above-mentioned threshold values to synthesize the RAMZI. This trend is similar for all other mesh sizes simulated in this experiment, so it can be inferred that the elapsed time increases linearly with the threshold value, whereas the MSE does the opposite exponentially. A MSE below 10^{-4} is sufficient for most applications, as non-ideal effects such as crosstalk, backscattering and receiver sensitivities can be more prominent. Additionally, the threshold selection will be application-dependent, and can greatly impact the synthesis of infinite-impulse response (IIR) filters specially as

Commented [DPL24]: Me
paper que compara graphs e ind

their spectral response provides an infinite number of non-zero components. For the remaining section, we will be employing a fixed threshold of -40 dB.

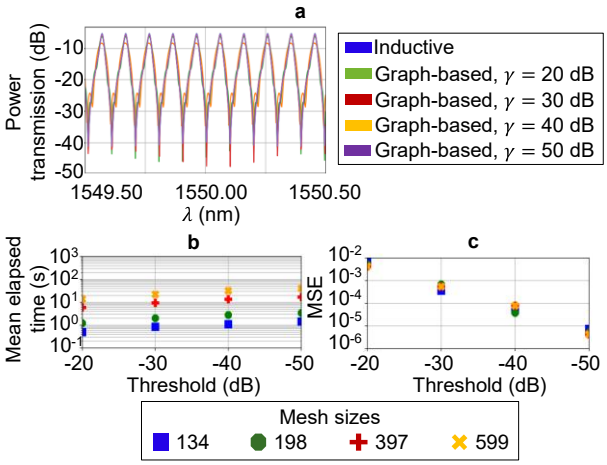


Figure 3.15. (a) Comparison of the simulated power spectra between our graph-based method under several power thresholds γ and inductive one for the synthesis of a balanced RAMZI filter in an 81-PUC waveguide mesh. (b) Variation of the elapsed time and (c) mean square error (MSE) between these methods with power threshold for the synthesis of the same circuit for four different mesh sizes.

In the following, we will compare the efficiency of this method to the inductive-based one presented in [77]. We will do so by benchmarking the elapsed time and demonstrating its applicability to a variety of applications. The configuration examples we will use are the balanced RAMZI filter presented in Figure 3.15, a Side-Coupled Integrated Spaced Sequence of Optical Resonators (SCISSOR), a 1x8 beamsplitter, an 8x8 unitary matrix, an arbitrary switch, and the all-passive state case. For each case, we sweep over two parameters: the wavelength resolution (going through 3, 11, 101 and 1001 wavelength points for a span of 1 nm centered at 1550 nm) and the number of PUCs (34, 36, 45, 72, 81, 87, 134, 198, 599, 799 and 1002), which will determine the mesh size. The insertion loss of each PUC will be fixed at 0.2 dB. Each circuit configuration involves the loading from a look-up table of the targeted circuit topologies and the configuration of the access waveguides to east and west ports. Once configured, we compute 10 times each simulation to enable the statistical analysis.

The first circuit is the balanced RAMZI filter previously introduced. Figure 3.16 (a) illustrates its synthesis in an 81-PUC waveguide mesh, using ports 15 and 33 as

input/output. The lower arm of the RAMZI would traverse PUCs 33, 40 and 47 after being split in PUC 25 while the upper one would be formed through PUCs 32, 39 and 46. PUCs 30, 31, 38, 44 and 45 are all set to bar state to configure the cavity of the structure. A simplified version of this circuit is illustrated in Figure 3.16 (b).

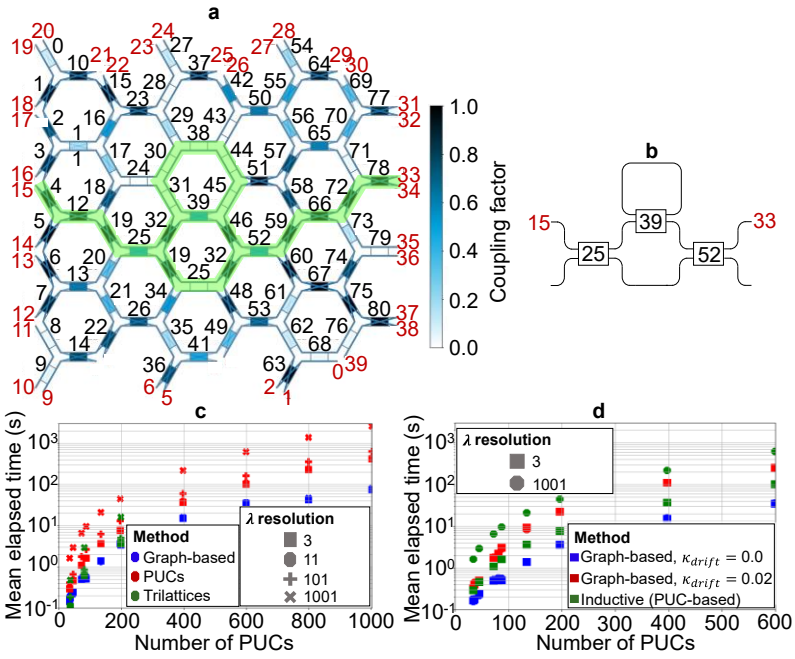


Figure 3.16. (a) Synthesis of a balanced RAMZI filter in a simulated 81-PUC waveguide mesh using ports 15 and 33 as input/output ports, (b) equivalent circuit synthesized, (c) evolution of the mean elapsed time (in seconds) with the number of PUCs for the synthesis of this circuit for four different wavelength resolutions with ideal coupling coefficients and (d) with coupling coefficients modelled as random variables featuring a drift with a standard deviation of 0.02.

The average elapsed times required for each simulation example can be found in Figure 3.16 (c). In it, we observe how the graph-based approach shows a consistent timing across different wavelength resolutions, showing its good scalability with the number of points. This approach is faster than the inductive method, with performance improvements of 5.6x, 5.8x, 8.3x and 32.8x for 1000-PUC meshes using a resolution of 3, 11, 101 and 1001 wavelength points, respectively. Additionally, it

also outperforms the trilattice-based approach when using large numbers of wavelength points. Only eight optical paths were found to arrive at destination port before power threshold was reached.

We also tested how the graph-based approach handles non-ideal components (i.e., PUCs presenting a phase response drift resulting in deviating configurations) during simulation. To do so, we modelled the coupling coefficient of each PUC forming the RAMZI as a truncated gaussian variable centered at its original (ideal) value and featuring 2% standard deviation drift. Values lying below 0 (bar state) or above 1 (cross state) are 'mirrored' by adding or subtracting respectively its absolute value to the original one. As illustrated in Figure 3.16 (d), we found that this resulted in an increase of the computational time for the graph-based approach. Specifically, for the 600-PUC example, it was around 2.5 times slower and 2.5 times faster than the inductive method for 3 and 1001 points, respectively. Additionally, it was 7 times slower than the graph approach when using ideal components.

Next, we present the simulation results for the optical SCISSOR defined in Figure 3.17 (a) between ports 12 and 38. This circuit is configured by setting PUCs 2, 7, 11, 13, 17, 20, 30, 33, 44, 47, 57, 60, 65, 67, 70, 75 and 80 to cross state, and PUCs 18, 19, 31, 32, 38, 40, 45, 46, 58 and 59 to bar state. Additionally, PUCs 24, 25, 51 and 52 are set in tunable coupler state to allow light to travel back and forth and be coupled throughout the structure. A simplified scheme is illustrated in Figure 3.17 (b). As with the previous example, Figure 3.17 (c) illustrates the elapsed time for the different simulation approaches. In this case, the graph-based approach, applied to a second order IIR, requires more computational time as the number of power contributions arriving at the output port increases up to 31. As a result, the elapsed computation time of graph-based approach is larger than that of a trilattice-based one for smaller size meshes and lies much closer to that of a single-cell inductive method for larger size meshes and small wavelength resolutions.

Repeating the test for imperfect components with the same methodology as in the previous example, we can observe in Figure 3.17 (d) that graph-based approach still outperforms inductive single-PUC approach for large wavelength points, offering better scalability. However, for a smaller number of points (3), the inductive method provides a faster computational time. For example, for 600-PUC meshes, the computation times are 275.85 and 103.8 seconds for the graph and inductive approach, respectively. Hence, growing to larger order IIR structures might prevent us from using this strategy and opt for inductive-based approaches.

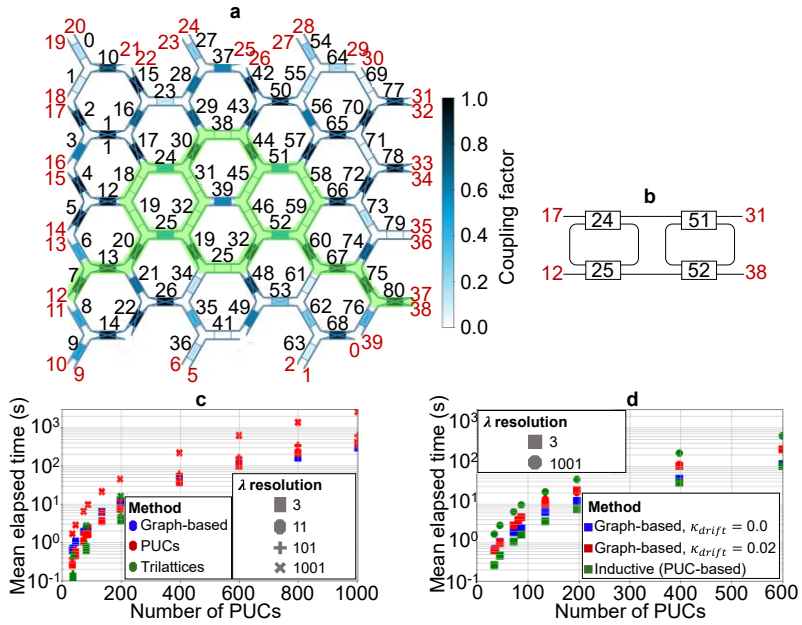


Figure 3.17. (a) Representation of a SCISSOR in a simulated 81-PUC waveguide mesh using ports 12 and 38 as input/output ports, (b) schematic of equivalent circuit synthesized, (c) evolution of the mean elapsed time (in seconds) with the number of PUCs for the synthesis of this circuit for four different wavelength resolutions with ideal coupling coefficients and (d) with coupling coefficients modelled as random variables featuring a drift with a standard deviation of 0.02.

Figure 3.18 (a) presents a 1x8 beamsplitter, outlined in Figure 3.18 (b). Such circuit takes port 15 as input and 0, 5, 7, 22, 29, 31, 32 and 33 as outputs. Here, light is split in PUCs 12, 18, 19, 25, 33, 40 and 77, all in Tunable Coupler (TC) state. To produce the spectral response of this circuit using our graph-based approach, we run the single pair algorithm recursively considering the eight output ports. As covered in Figure 3.18 (c), for the mesh sizes considered, the graph-based approach performs faster than the inductive one for lower number of spectral points, and much faster for larger spectral points. The efficiency of graph-based approach for this example relies on the rapid discarding process of optical paths during the execution of the algorithm. This no longer occurs, however, if we introduce non-ideal components such as in Figure 3.18 (d). In this case, graph-based approach's computation time increases dramatically because of the appearance and spreading of multiple paths during the execution of this method that may not be discarded up to final stages of its execution. As an example, for the 600-PUC mesh and larger spectral vectors (1001 wavelength

points), the graph-based approach requires around 1.6x more time than the inductive method, while the gap becomes of 10x for lower (3) number of points. This application demands higher resolution and number of points when validating the channel ripple conditions. Otherwise, the expected spectral behavior is inherently flat. Hence, the presence of non-ideal configurations and the increment of optical outputs may favor the selection of the inductive method approach.

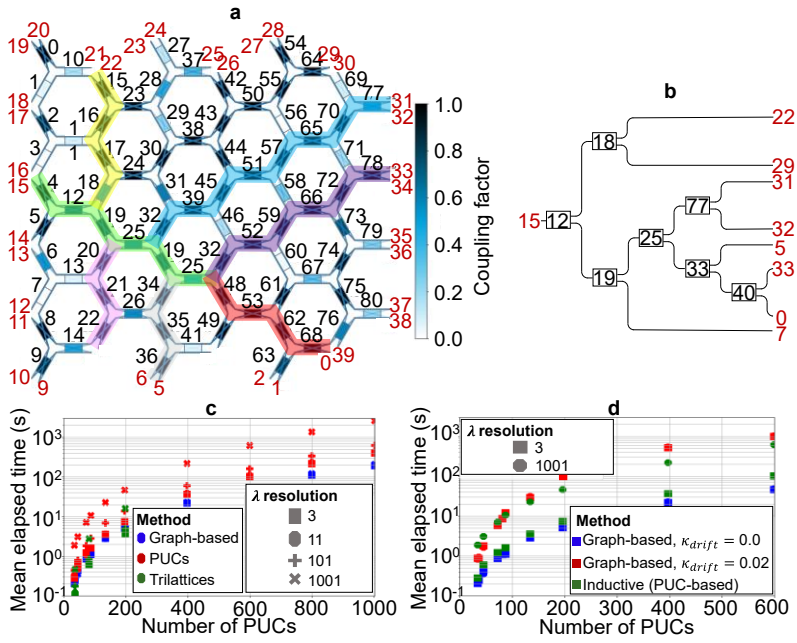


Figure 3.18 (a) Representation of a 1x8 beamsplitter in a simulated 81-PUC waveguide mesh using port 15 as input port and ports 0, 5, 7, 22, 29, 31, 32 and 33 as output ports, (b) schematic of the equivalent circuit synthesized, (c) evolution of the mean elapsed time (in seconds) with the number of PUCs for the synthesis of these circuits for four different wavelength resolutions with ideal coupling coefficients and (d) with coupling coefficients modelled as random variables featuring a drift with a standard deviation of 0.02.

Next two examples focus on multiple input-multiple output applications. First, we configure the 8x8 optical matrix appearing in Figure 3.19 (a) and represented in Figure 3.19 (b). Here, we performed simulations for three mesh sizes of 198, 397 and 599 PUCs since it was not possible to allocate this structure in smaller size meshes. In the figure, mesh ports 33, 34, 35, 36, 37, 38, 39 and 40 work as input ports and

mesh ports 1, 2, 3, 4, 5, 6, 7 and 8 operate as output ports. The configuration of an arbitrary linear matrix requires the configuration of the tunable couplers, driving PUCs 10, 11, 12, 13, 28, 29, 30, 43, 44, 45, 46, 61, 62, 63, 76, 77, 78, 79, 94, 95, 96, 109, 110, 111 and 112. In addition, we configure the access waveguides following a similar approach as in the beamsplitter case. The PUCs interconnecting the TC-configured cells are set into cross state. The configuration of the TCs is selected employing random distributions to maintain the arbitrariness of the configuration process.

In this case, the graph-based approach is run iteratively for each input/output pair to complete the triangular matrix of the resulting scattering matrix. The results, in Figure 3.19 (c), illustrate how the elapsed time of the graph approach is around 2 and 3 orders of magnitude larger than that of the inductive method for the larger and shorter wavelength vectors, respectively.

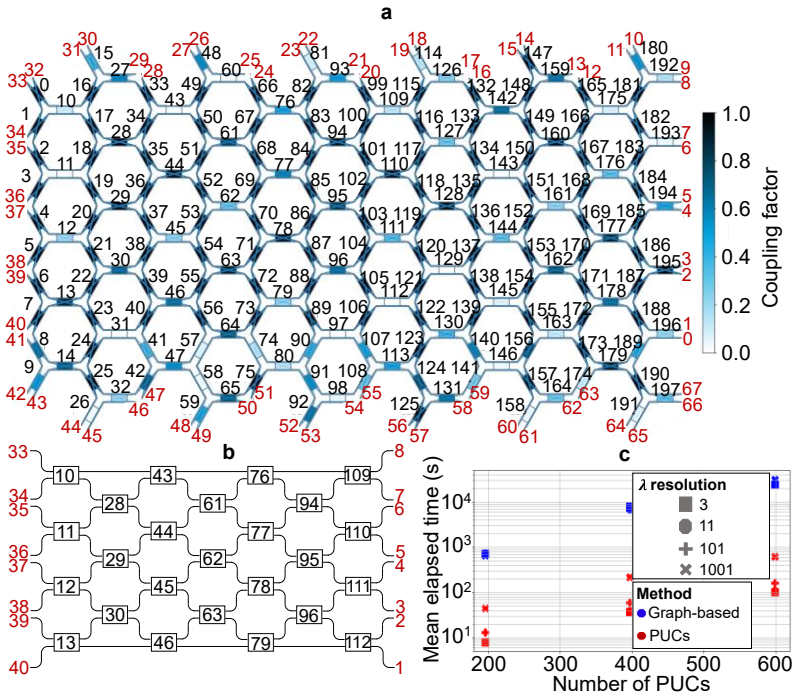


Figure 3.19 (a) Representation of an 8x8 identity matrix in a simulated 198-PUC waveguide mesh using ports 33, 34, 35, 36, 37, 38, 39 and 40 as input ports and ports 1, 2, 3, 4, 5, 6, 7 and 8 as output ports, (b) schematic of the equivalent circuit synthesized,

(c) evolution of the mean elapsed time (in seconds) with the number of PUCs for the synthesis of these circuits for four different wavelength resolutions with ideal coupling coefficients.

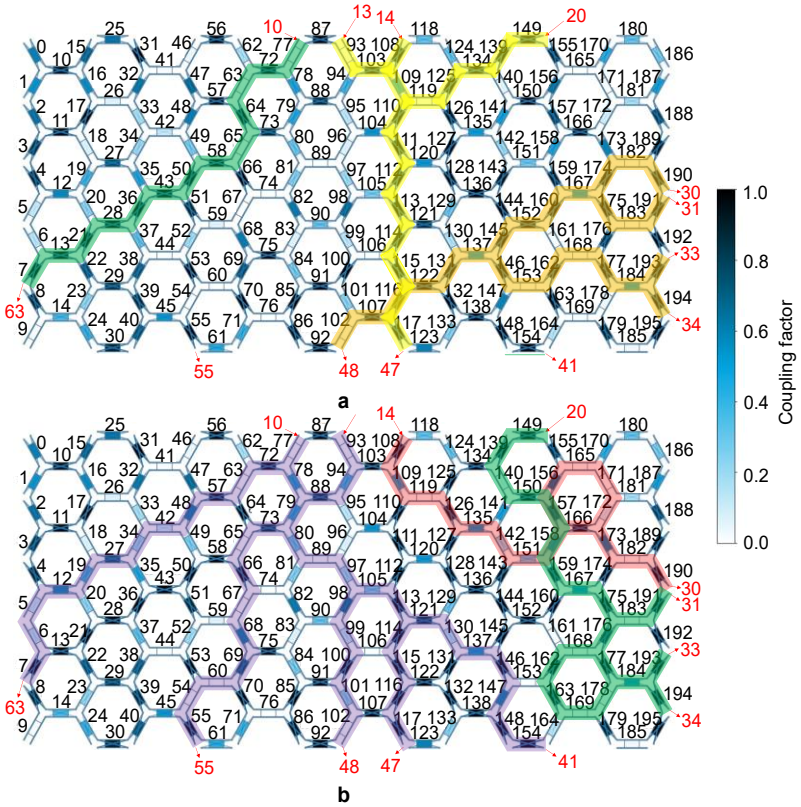


Figure 3.20. Waveguide mesh arrangement of 196 programmable unit cells configured as an arbitrary optical switch and processor including point-to-point (in green), point to multipoint (beamsplitters, in orange and pink), signal combiners (in yellow) and wavelength-sensitive filters (in red): (a) configuration A, (b) configuration B.

Our second multiple-input-multiple output circuit will be an arbitrary switch featuring two different operation modes, hereinafter referred to Config. A and Config. B. We can see in Figure 3.20 (a) how Config. A includes the synthesis of two optical paths connecting ports 10 and 63 and 41 and 55, a 1x4 beamsplitter using port 48 as input and ports 30, 31, 33 and 34 as outputs, and a 2x1 combiner with input ports 13 and 14, port 20 acting as output and using port 47 to drain optical leakage. At the same time, Config. B appears in Figure 3.20 (b) containing a 1x6 beamsplitter with

input port 10 and output ports 13, 41, 47, 48, 55 and 63, two optical paths connecting ports 20 and 33 and 31 and 34, and an optical ring resonator (ORR) between ports 14 and 30.

To exploit the analytical model developed, we determine three different use cases for both switch configurations. Case 1 represents a scenario using ideal components, in which the coupling factors of all PUCs constituting the synthesized circuits in both switches have no phase drift and therefore feature ideal bar/cross/tunable coupler states. For cases 2 and 3, we tested the handling of non-ideal components (i.e., presenting a phase response drift resulting in deviated configurations). The difference between both use cases resides in the state of the remaining passive PUCs of the waveguide mesh, with random and all-cross coupling values for test cases 2 and 3 respectively. Once configured, we have computed 60 times each simulation to enable a statistical analysis, employing two equally equipped desktop computers including 4-core and 3.60 GHz processors. For the graph-based approach, we employ a fixed power threshold of -40 dB.

As a benchmarking example, Figure 3.21 (a) illustrates the resulting spectral traces associated to Configuration A for the non-ideal component examples. We show the combination of ports when an input signal is injected through port 48. One can identify the optical targeted output ports of the 1x4 beamsplitter, the optical crosstalk at ports sharing common PUCs (< 20 dB), and the remaining ports (< 30 dB). Next, Figure 3.21 (b) shows the spectral traces associated to Configuration B for the non-ideal component examples. We show the combination of ports when an input signal is injected through port 14. We can identify the optical targeted output ports of the filtered channel, the optical crosstalk at ports sharing common PUCs (< 20 dB) and the remaining ports (< 30 dB). In addition, Table 3-III includes the elapsed time required for each application scenario.

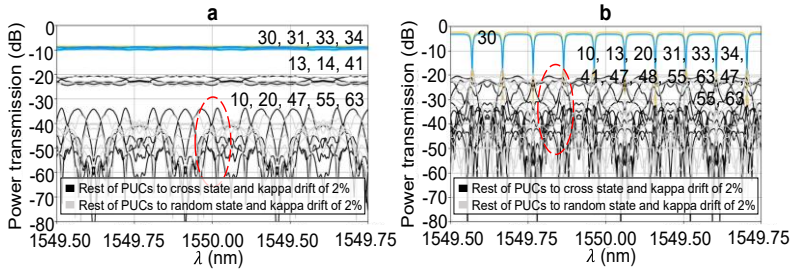


Figure 3.21. Spectral responses from every switch point highlighted in Figure 3.20 (a) under Configuration A, entering through port 48, (b) under Configuration B, entering through port 14. The simulation considers non-ideal-2%-drift at every programmable unit cell. Orange-colored spectra are provided using components with ideal coupling coefficients (case 1), while cyan- and navy-blue ones correspond to imperfect

components featuring phase drifts of 2% while setting the passive phases of the remaining PUCs to random and cross state, respectively.

Table 3-III. Summary of the mean elapsed times (in seconds) and standard deviations (between parentheses) after 50 repetitions of the synthesis of both switch configurations A and B using graph-based and inductive methods under the three scenarios described in the text.

		Number of λ points	Case 1	Case 2	Case 3
Graph-based	Config. A	101	9.28 (0.25)	413.90 (35.03)	205.81 (34.25)
		1001	9.71 (0.18)	437.60 (49.41)	269.02 (29.06)
	Config. B	101	29.30 (0.56)	381.17 (43.69)	209.04 (25.49)
		1001	29.58 (0.49)	397.90 (36.80)	187.07 (27.91)
Inductive	Config. A	101	12.04 (0.21)	12.16 (0.16)	12.05 (0.13)
		1001	42.25 (0.41)	42.45 (0.39)	42.87 (0.26)
	Config. B	101	13.51 (0.31)	13.69 (0.18)	13.99 (0.67)
		1001	45.21 (0.80)	45.48 (0.37)	43.27 (0.62)

Overall, for programmable photonic applications with many closed optical paths and lower number of feedback and feed-forward loops, the graph-based methodology again performs faster simulations. This is also true when only a small subset of elements is required from the scattering matrix. In contrast, the use of the inductive approach provides faster simulation times for arbitrary complex circuits requiring large number of optical ports.

Finally, we conclude this set of experiments by simulating the full response (i.e., using all mesh ports as inputs and outputs) of the waveguide mesh under a random configuration. This represents a scenario with arbitrary complex configurations present on certain applications or uncalibrated circuits. An interesting application of this feature deals with neuromorphic computing engines [67]. The results of the analysis are presented in Figure 3.22. For this application, motivated by the exigent

computational times of the graph-based approach, we selected mesh sizes up to the 196-PUC waveguide mesh only.

As for the 8x8 switch case, the graph-based approach runs iteratively for each input/output pair to complete the triangular matrix of the targeted scattering matrix. As for the beamsplitter and the 8x8 circuits, the randomness of PUC transmission states favors the existence of many candidate paths that take long by pathfinder to discard. This effect is exacerbated by the presence of internal feedback loops. Precisely, the performance of the graph-based approach is around 3 and 4 order of magnitude slower than the inductive method approach for the longest and shortest wavelength vector, respectively.

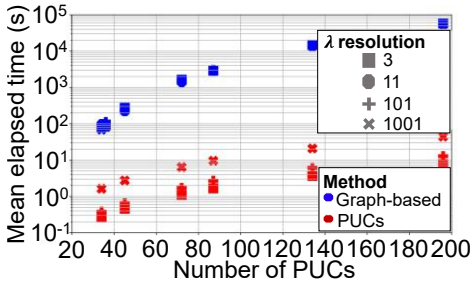


Figure 3.22. Evolution of the mean elapsed time (in seconds) with the number of PUCs for the obtention of the response of all ports in a passive waveguide mesh for four different wavelength resolutions.

To summarize the performance of the proposed methods, we can conclude that, for applications with many closed optical paths and lower number of feedback and feed-forward loops, the graph-based methodology performs faster simulations. This is also true when only a small subset of elements is required from the scattering matrix. In addition, the performance of this approach is significantly invariant with the number of wavelength points. Multicore electronic processors could be employed to improve the efficiency of this tool using parallel processing. In contrast, the use of the inductive approach provides faster simulation times for arbitrary complex circuits requiring many optical ports.

Chapter 4

Experimental Applications

4.1 Hardware and experimental set-ups

The previous two chapters covered the development of essential software utilities to enable the practical and automatic operation of programmable photonic processors. In this chapter, instead of using simulations, we will apply and demonstrate their performance using real devices. Specifically, we will utilize two different chip devices (and experimental set-ups) referred to as Design A and Design B.

4.1.1 Design A

Chip design A fabrication was carried at the Optoelectronic Research Centre (ORC) of the University of Southampton prior to the start of this thesis, in 2017 [49]. It consists of 30 programmable unit cells forming a 7-hexagonal cell waveguide mesh in the shape of a revolver. As commented back in chapter 1, each of these programmable unit cells is attached to two thermal units, each of which is accessible through two electrical pads. In summary, the device includes the following elements:

- 7 hexagonal cells.
- 30 programmable unit cells (PUCs).
- 24 input/output optical ports.
- 60 thermal tuners.
- 120 electrical DC pads.

The experimental set-up used to work with this device is presented in Figure 4.1. We connect a tunable laser (ANDO AQ4321D) featuring 1 pm wavelength resolution to one of the mesh optical ports, setting it as input. Another of its optical ports is connected to an optical spectrum analyser, working as output. To handle thermal stability, the PCB was held to a custom copper thermal chuck. In such way, heat can

be quickly transferred way from the chip substrate into the surrounding environment, maintaining optimal device performance.

By last, our set-up also included a group of fifteen independent PROMAX current sources to drive as many unit cells as possible. Many of design 1 phase shifter actuators were no longer functional due to the non-optimal robustness and performance of the metal layer and past material stress research, so we found out that this limited number of electrical sources was sufficient for our experiments.

Commented [DPL25]: No uso poco cuidadoso, una tecnol

Figure 4.1 summarizes the arrangement of the experimental set-up used for chip design A.

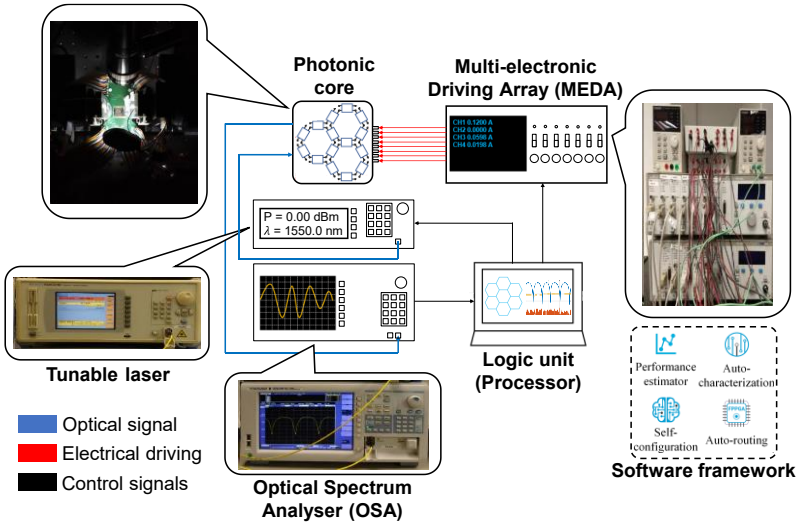


Figure 4.1. Schematic of chip design A. A 13-channel multi-electronic driving array feeds electrical power to several phase actuators of a 30-PUC waveguide mesh. A table-top tunable source laser is connected to the input, while the measurements are saved using an optical spectrum analyser.

4.1.2 Design B

Chip design B fabrication was delivered by a standard silicon photonic platform of 220 nm thick silicon back in 2021. The mesh design follows the flattened, hexagonal topology represented in Figure 1.5 (f) and proposed in [50]. This design consists of 72 programmable unit cells arranged in a 17-hexagon mesh such as the one from

Figure 4.2. Again, each of these cells counts with two independent thermal phase shifters.

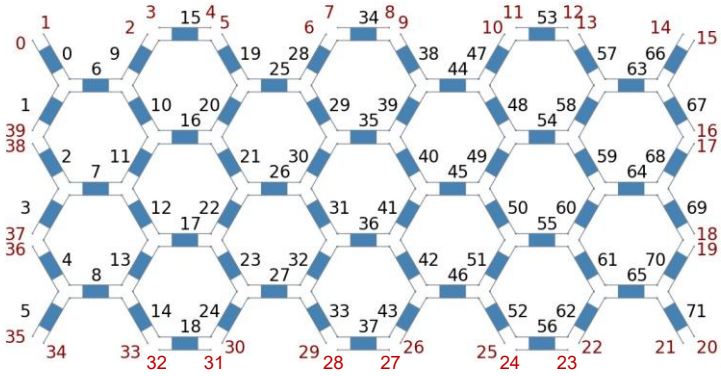


Figure 4.2. 72-PUC hexagonal mesh core used as part of our setup B.

All design B's optical ports are oriented to the same direction (left), so that the whole waveguide mesh can be accessible through a fixed/packaged fiber array. In contrast, Design A distributes the optical I/Os in two grating coupler arrays and the assembly do not incorporate optical packaging.

Commented [DPL26]: No de southampton no estaba empa 2 fiber arrays

Apart from these differences in the waveguide mesh topology, this design presents two more additional subsystems with respect to previous one. First, it includes an array of additional PUCs acting as a monitoring unit array (MUA) between the waveguide mesh and the input/output optical ports. These PUCs can be individually tuned to regulate the power coming inside the mesh as an optical variable attenuator and outside the chip, selecting the optical power ratio that feeds an internal photodetector.

Apart from the MUA, design B also counts with a selection of high-performance building-blocks acting as high-quality optical filters. These include A 4-order Ring-assisted Mach Zehnder Modulator (RAMZI), two 4-order lattice filters and a 4-order coupled-resonator optical waveguides (CROW) filter; and are connected to the optical core through its 12 southmost (out of 40) optical ports, whereas the remaining ones will be those leading to the MUA.

The set-up employed to operate this device also presents some relevant changes compared to the one from Figure 4.1. This time, the OSA was replaced by a multi-electronic monitoring array (MEMA), a custom array of photodetectors that allowed us to measure the system's optical output response through all ports simultaneously, hence allowing us to appreciate device's reconfigurability in real time. We also

replaced the PROMAX current sources by another custom multi-electronic driving array (MEDA) of 144 arrays, as many as needed to drive all the phase shifters from the optical core.

4.2 Experimental applications

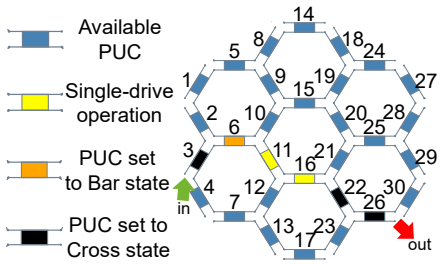
4.2.1 Self-configuration of photonic structures

4.2.1.1 Point-to-point interconnects

In order to gain intuition on the impact of the cost function selection, our first experiment involves the creation of an optical path and the mapping and extraction of a bi-dimensional (two variables) cost function space. To do so we set up the optical path connecting 'in' and 'out' ports shown in Figure 4.3 using a pre-defined current setting. This current setting was a single path from the many delivered through the pathfinder algorithm introduced in subsection 3.2. Next, we can use the results provided by the self-calibration routine explained in subsection 3.3.1 to set every PUC constituting such path to either cross or bar state as corresponded except for PUCs 11 and 16, which remained in passive state intentionally. Finally, we performed a current sweep on these two PUCs to create the plot contours appearing in Figure 4.4 and Figure 4.5, whose values are expressed by means of the cost functions specified in Figure 4.3 (b). These functions either attempt to maximize of the optical signal's output power at its central wavelength ($CF_{1\text{-}3}$) or average it over the whole spectrum ($CF_{4\text{-}6}$). Also, $CF_{1,4}$ are expressed in dB, while $CF_{2,5}$ and $CF_{3,6}$ appear in linear units.

During the analysis of results in both Figure 4.4 and Figure 4.5 we do not appreciate any significant differences between choosing the central element of the spectrum or averaging over all the samples captured by the OSA. Also, when we represent output power using linear units, the slope is steeper compared to the logarithmic scale. Intuitively, this should help to speed up convergence for first-order optimization algorithms, as the calculated gradients at each point will present much larger norms.

Commented [DPL27]: Al explicado en el capitulo 3 debe explicar en 2 lineas un resumen



(a)

CF	Description
1	$-10\log_{10}\left(\left\|H_{i,o}(\mathbf{v},\lambda)\right\|^2\right)$
2	$-\left\|H_{i,o}(\mathbf{v},\lambda)\right\|^2$
3	$-\left\|H_{i,o}(\mathbf{v},\lambda)\right\|$
4	$-10\log_{10}\left(\left\|\overline{H_{i,o}(\mathbf{v})}\right\|^2\right)$
5	$-\left\|\overline{H_{i,o}(\mathbf{v})}\right\|^2$
6	$-\left\|\overline{H_{i,o}(\mathbf{v})}\right\|$

(b)

Figure 4.3. (a) Labelled schematic of the waveguide mesh arrangement and driven PUCs by the MEDA, (b) List of cost functions used during the experiment.

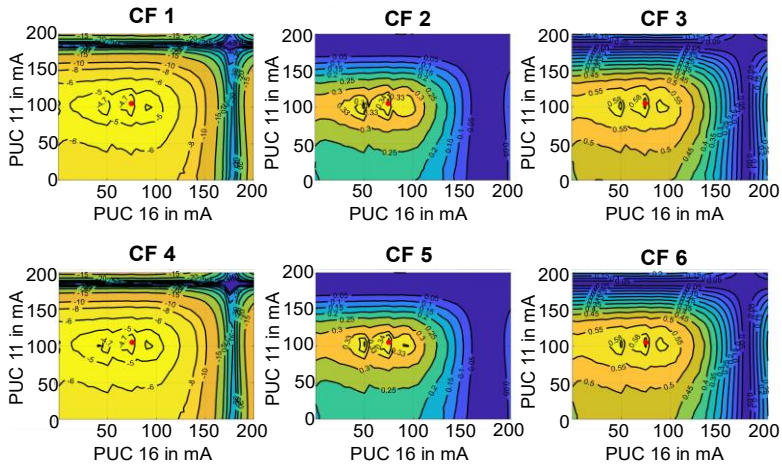


Figure 4.4. Contour plots of PUCs 11 and 16 using I (in mA) as variable in the synthesis of the optical path depicted in Figure 4.3 (a) for the cost functions defined in Figure 4.3 (b).The red point corresponds two an optimum configuration (the one obtained using pre-defined current settings), in which all optical power is maximized.

Commented [DPL28]: A y
definir aquí que indica el punto

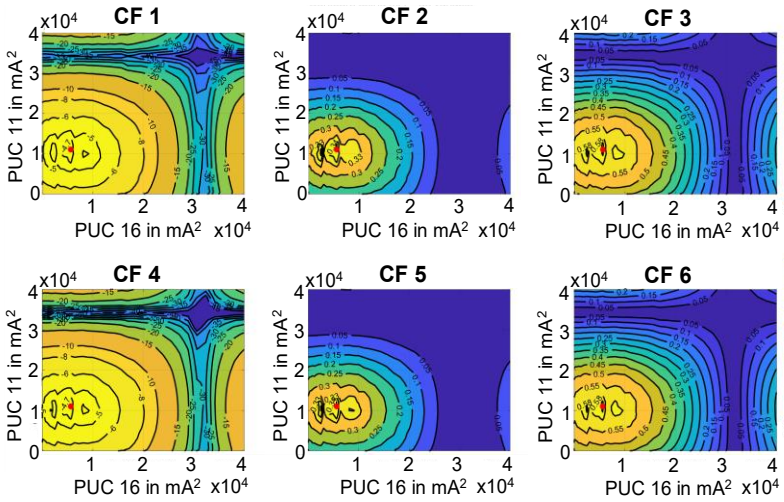


Figure 4.5. Contour plots of PUCs 11 and 16 using I^2 (in mA^2) as variable in the synthesis of the optical path depicted in Figure 4.3 (a) for the cost functions defined in Figure 4.3 (b). The red point corresponds two an optimum configuration (the one obtained using pre-defined current settings), in which all optical power is maximized.

4.2.1.2 Optical filters

In the next experiment, we synthesize a 2-PUC MZI by monitoring the set of PUCs highlighted in yellow in Figure 4.6 (a). The PUC highlighted in red (16) works under 'dual-drive' configuration, which means that we drive current into both of its upper and lower phase shift actuators to provide us independent control of its phase and amplitude response, allowing us to adjust the extinction ratio and the wavelength notch of the filter. After being provided with the spectral mask of our MZI filter by using its corresponding pre-set of currents, we ran a set of experiments using PSO algorithm over thirteen variables (corresponding to a single phase shifter from mesh PUCs 3, 6, 10, 11, 12, 13, 17, 21, 22, 23 and 26 and to both phase shifters from PUC 16). Results are shown in Figure 4.6 (b) and Figure 4.6 (c). There, we observe how the transmission spectra of the new filters closely match that of the one obtained using current presents -in fact, one of them even features a slightly higher ER of around 30 dB. In all these experiments (and in the ones that follow), we set an adaptive inertia parameter that gradually decreased from 1 to 0.35. We also set a limit of 40 mA (in absolute value) on the velocity of each particle's actuator to prevent strong variations in the particle's positions, especially at the beginning of the process, when the inertia coefficient is still large. We also set a limit on each particle's actuator position

Commented [DPL29]: Ind
estamos optimizando y a que P

Commented [DPL30]: Fig

between 0 and 200 mA to prevent damage to the chip. Each experiment took about two hours to complete because of the slow laser sweep (10 seconds per operation). Next, we repeated the same procedure to synthesize a 6-PUC ORR, whose results appear in Figure 4.7. Again, we obtained an ER of around 30 dB for this configuration.

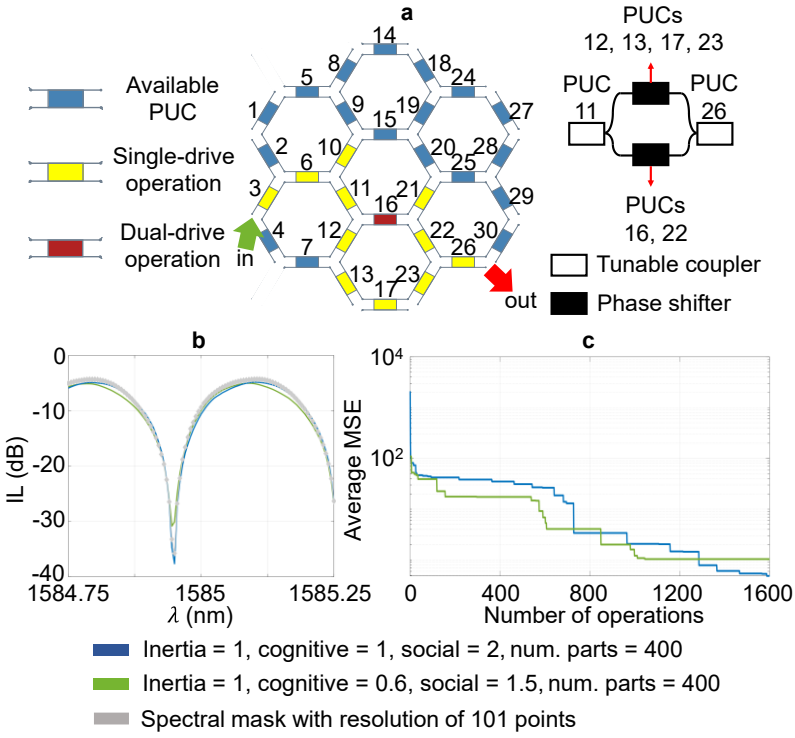


Figure 4.6. Experimental results of the synthesis of a 2-PUC unbalanced Mach-Zehnder Interferometer in our photonic processor using PSO algorithm in set-up A. (a) Schematic of the 30-PUC waveguide mesh. PUCs under use in our experiment appear highlighted in yellow and red (dual-drive configuration). (b) Final experimental traces obtained after executing the algorithm. The spectral mask of the filter was obtained through current presets. (c) Evolution of the average mean square error provided by the algorithm with the number of operations.

Next, we aimed to reproduce a 10-PUC ORR by using the highlighted arrangement from Figure 4.8 (a). To do so, we increased the number of samples of our trace to 301 from the 101 used in our previous experiments prior to the execution of the algorithm, as we are dealing with a filter with smaller FSR (≈ 0.1 nm for the 6-PUC ORR compared to 0.06 nm for the 10-PUC one) and therefore, with a more 'challenging' spectrum to be captured by our optical spectrum analyzer.

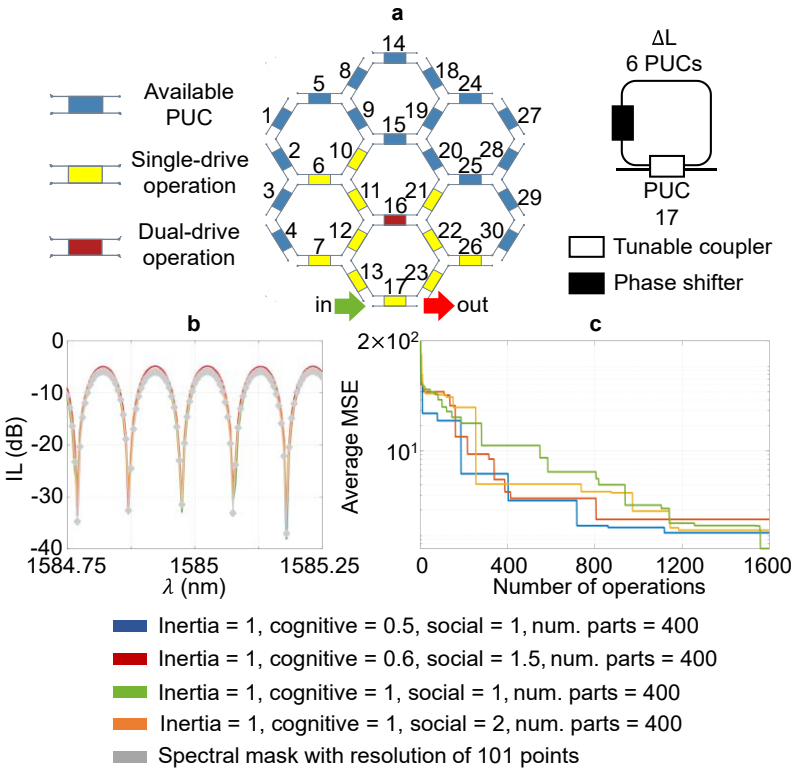


Figure 4.7. Experimental results of the synthesis of a 6-PUC optical ring resonator in our photonic processor using PSO algorithm in set-up A. (a) Schematic of the 30-PUC waveguide mesh. PUCs under use in our experiment appear highlighted in yellow and red (dual-drive configuration). (b) Final experimental traces obtained after executing the algorithm. The spectral mask of the filter was obtained through current pre-sets. (c) Evolution of the average mean square error provided by the algorithm with the number of operations.

The average MSE results of our synthesized filter with respect to the corresponding spectral mask were again obtained through an already known pre-set of currents. As observed in Figure 4.8 (b, c), results suggest that further increasing the number of operations would lead to a better matching between the obtained spectrum and the filter mask, which still features around 20 dB and a similar passband insertion loss for both synthesized structures. We can speed this process by using broadband sources or faster tunable lasers.

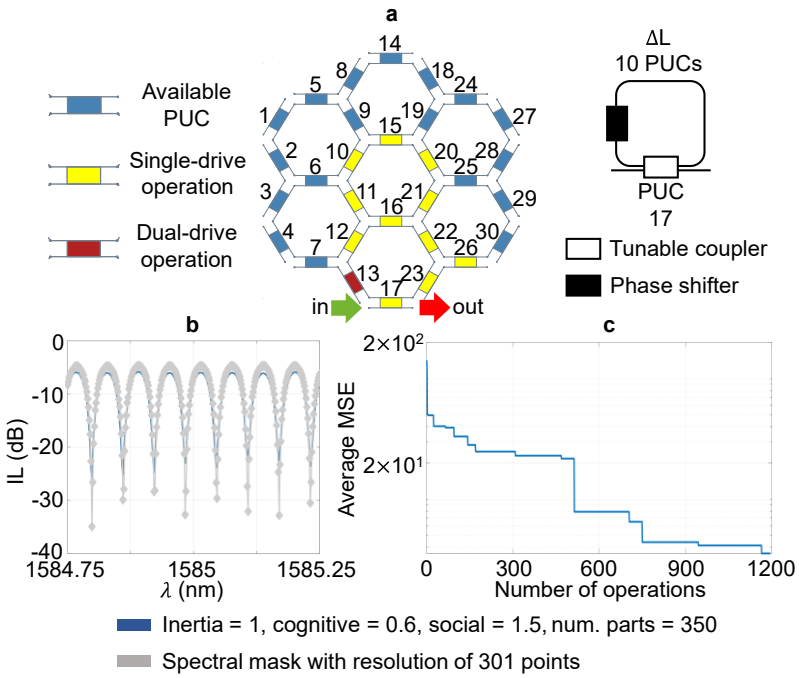


Figure 4.8. Experimental results of the synthesis of a 10-PUC optical ring resonator in our photonic processor using PSO algorithm in set-up A. (a) Schematic of the 30-PUC waveguide mesh. PUCs under use in our experiment appear highlighted in yellow and red (dual-drive configuration). (b) Final experimental traces obtained after executing the algorithm. The spectral mask of the filter was obtained through current pre-sets. (c) Evolution of the average mean square error provided by the algorithm with the number of operations.

We also synthesized the 4-PUC MZI using the set of highlighted PUCs illustrated in Figure 4.9 (a). Unlike previous experiments, this time we employed a self-made

spectral mask in the same manner than our simulations using the following expression:

$$H(\lambda) = 20\log_{10}\left(\left|\cos\left(\pi\frac{\lambda}{FSR} - \delta\right)\right|\right) \quad (8)$$

in which both the FSR and the wavelength λ are expressed in nm. Phase variation δ represents any arbitrary shift of the spectral response for our filter -which can be achieved through dual-drive operation-, and it is set in our case to 0 rad. In addition, we flattened both passband and eliminated band regions (-1.5 dB in the passband and -28 dB in the stopband) of our mask to provide a slightly modified spectrum from the one that can be achieved using the set of PUCs at our disposal. Such flat spectral response would resemble more to the one provided by a higher-order filter rather than to the one supplied by our first order MZI. In any case, we can observe from the figure how the algorithm 'does its best' to match the filter response to the spectral mask provided by the user as much as it can. In accordance to simulated results in this work, the degree of similarity between both spectra is expected to increase if more electrical channels to drive a larger number of unit cells are at our disposal. We performed this measurement on a different chip than the one used in previous figures, whose grating design was centered at 1570 nm rather than at 1585 nm. Looking at the variation of the measured MSE (Figure 4.9 (c)), it can be observed that it does not vary quite significantly compared to those from previous figures. This happens due to the use of a larger number of sampling points during the experiment (501, in contrast to the 101 and 301 used in Figure 4.6 and Figure 4.7, and Figure 4.8, respectively). As a result, there is a much larger number of points close to the spectrum passband that contribute to reduce the average error, especially at early stages of the algorithm when the filter notches have not been still formed.

To illustrate the dynamic operation of the photonic processor enabled by the algorithms presented, we continue with another experiment that employs a precalibrated mesh and a set of pre-defined circuits illustrated in Figure 4.10. To conduct the experiment, as shown in Figure 4.10 (a), we first employed the self-calibration routine to calibrate the response of every phase actuator and PUC from chip design A, along with their power consumptions. Then, we used this information as input for the auto-routine algorithm to set up seven different circuit implementations, referred in Figure 4.10 (b) as configs. 1-7: a 10-PUC optical ring resonator (ORR), a 4-PUC imbalanced MZI, a simultaneous combination of a 6-PUC ORR working in parallel with a 2-PUC imbalanced MZI, a 6-PUC ORR, a second-order coupled resonator optical waveguide (CROW) with cavity length of 6 PUC, a simultaneous combination of two delay line channels of 6 and 5 PUCs, and a 12-PUC ORR. Translating each circuit configuration to an array $\boldsymbol{v}$, including the required electrical current value for each phase actuator in the arrangement and loading each at a time to the processor allows us to demonstrate the first dynamic reconfiguration

Commented [DPL32]: Me
los experimentos anteriores a e
Commented [ALH33R32]
Commented [DPL34]: En
en este chip.
Commented [ALH35R34]
poner que las sacaste a mano? I
mostrando la utilidad de la auto

of a multipurpose waveguide mesh arrangement. Figure 4.10 (c) illustrates how the waveguide mesh arrangement functionality evolves over time. The precise spectral performance can be found in the lower insets. The tuning steps are limited to jumps of 5 mA to prevent undesired overshoots, to protect the circuit and to better illustrate the dynamic response between configurations. Each step can be done in <1 s, limited by the hardware of the control system employed and its USB connections, as well as the serial peripheral interface.

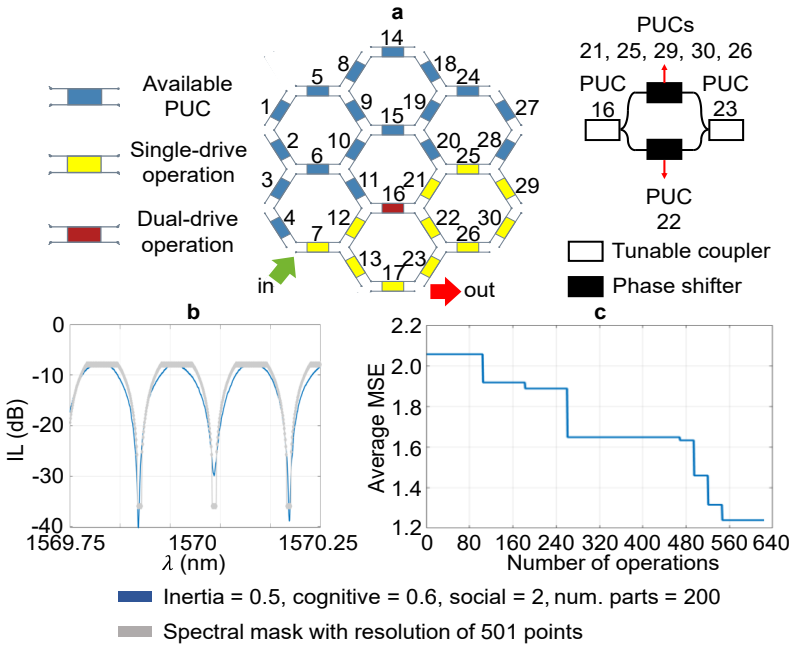


Figure 4.9. Experimental results of the synthesis of a 4-PUC unbalanced Mach-Zehnder Interferometer in our photonic processor using PSO algorithm in set-up A. (a) Schematic of the 30-PUC waveguide mesh. PUCs under use in our experiment appear highlighted in yellow and red (dual-drive configuration). (b) Final experimental traces obtained after executing the algorithm. The spectral mask of the filter was obtained through current presets. (c) Evolution of the average mean square error provided by the algorithm with the number of operations.

Commented [DPL36]: La de salir por 23.

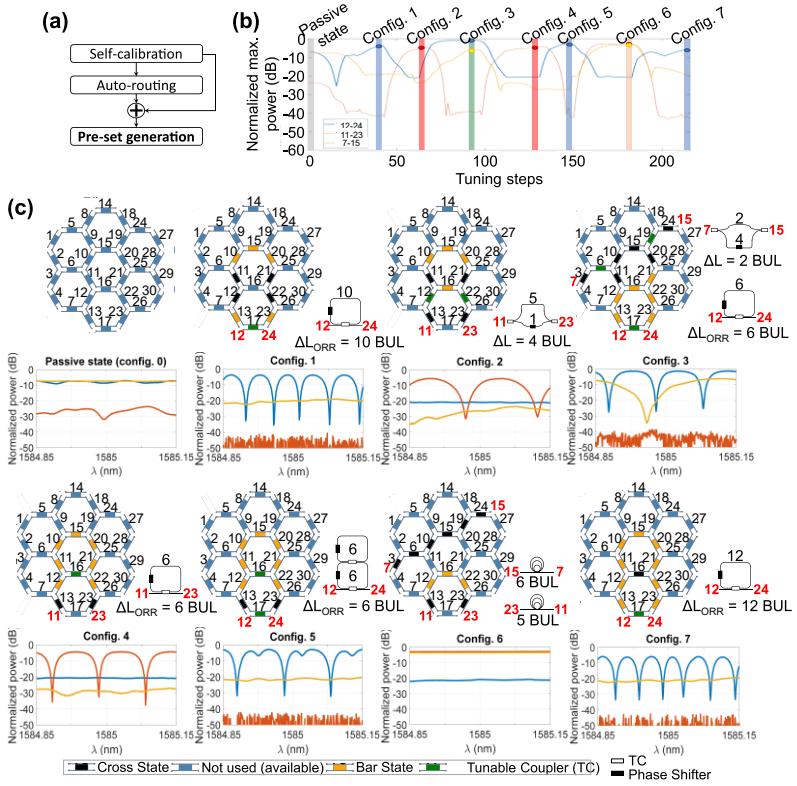


Figure 4.10. Experimental results for sequential circuit programming using auto-routing and prior-knowledge-based algorithms. The algorithms are applied to a 30-PUC hexagonal waveguide mesh with measured normalized maximum optical powers at channels 12-24, 11-13 and 7-17. (a) Workflow of the experiment following a self-calibration routine, the auto-routing algorithm, and the generation of pre-sets. (b) Dynamic configuration illustrated by the evolution of the normalized maximum optical power versus tuning steps with maximum current step change of 5 mA allowed per phase actuators for the three optical channels. (c) Waveguide mesh arrangement with relevant unit cells configured in passive, cross, bar, or tunable coupling states (up), and normalized spectral response measured for each circuit configuration (down) for the following eight configurations: Config. 0: passive state, Config. 1: 10-BUL ORR, Config. 2: 4-BUL MZI, Config. 3: 2-BUL MZI and 6-BUL ORR, Config. 4: 6-BUL ORR, Config. 5: 6-BUL CROW, Config. 6: 5-BUL and 6-BUL ORR, Config. 7: 12-BUL ORR. Traces are normalized to a straight waveguide with coupling and propagation loss of 22 dB.

4.2.2 Self-calibration of photonic waveguide meshes

In this subsection, we proceed to the calibration of a real photonic processor's waveguide mesh (design B) following the same steps presented in chapter 3.

Table 4-I shows the results for five different physical assemblies. The first four ones show around 10^5 operations, whereas the latest one features around a 40% more. The latest column reveals the reason: C7_S assembly could not calibrate all its phase shifters (PS) effectively due to electrical interconnection issues in a couple of phase actuators. The algorithm will attempt many times to re-calibrate defective PS in vain, with the consequent increase of this number of operations.

Regarding the elapsed time to carry out this operation, we can observe that almost in all cases it exceeds twelve hours. While this time may look significant, we must consider that we only need to perform this operation once on our device. Additionally, as we will comment in next chapter, any improvements in the involved subsystems speeds (Laser, MEMA or MEDA) lead to a substantial improvement.

Table 4-I. Calibration results of several physical 72-PUC assemblies.

Assembly ID	Operations	Time (h)	Successful calibration of all PS?
E7_S	104239	13.8	Yes
E9_S	109737	13.5	Yes
C9_S	93387	11.7	Yes
B5_S	101565	12.4	Yes
C7_S	145107	18.0	No

4.2.3 Self-characterization of photonic processors

In this subsection, we will show the performance of the self-characterization routine presented in chapter 3 on our design B. The steps to accomplish this task are pretty much the same than those executed during the simulations; however, there are some small differences, mostly regarding to the existence of the HPBBs in this design.

Commented [DPL37]: En y reportado esos HPB?

Commented [ALH38R37]: con una malla S, he dejado los en el lab). ¿Pudiste echar un vi

As introduced at the beginning of this chapter, these HPBBs are connected to the mesh core from Figure 4.2 by means of optical ports 22 to 33. As these mesh ports are therefore not connected to any of the MEMA outputs, it will be pointless to use them to accumulate paths going to them, since we would not be able to measure their optical powers. In other words, we must carry out this task using ports 0 (input) to 21, and 34 to 39 only.

A direct consequence of this can be deduced by having a look to the figure. By doing so, we can observe how the losses of the three PUC triplets at the bottom side of the arrangement (PUCs 14, 18 and 24; PUCs 33, 37 and 43; PUCs 52, 56 and 62) cannot be decoupled in any way, as -opposite to the simulated waveguide mesh used during chapter 3, which did not count with any HPB attached to it. In other words: with the actual availability of output optical ports in this design, there is no single path going through PUC 18 (to cite an example) that does not also traverse PUCs 18 and 24; and the same goes for the remaining two PUC triplets beside. This limits the accuracy with which we can estimate the insertion loss of each of those PUCs in the laboratory.

Paths searching and path power retrieving processes at the beginning of this process are the same in both scenarios. The first difference comes during linear fitting. Figure 4.11 represents the fitting of all paths retrieved between ports 0 and 1 in 40 seconds using a real device. Note how, opposite to the ones previously showed in Figure 3.9, we no longer dispose of a visual reference of the actual coupling losses, since this is the real magnitude we are trying to deduce from this experiment.

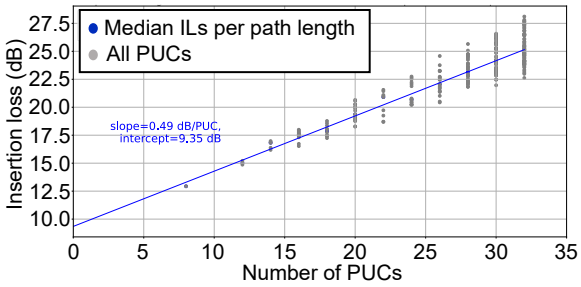


Figure 4.11. Linear fitting of all the accumulated paths using our pathfinder algorithm between mesh ports 0 and 1 during 40 seconds on a real chip.

In Chapter 3, we concluded that the error of the estimation of the optical access loss (intercept point in previous figure) is critical for the PUC mesh insertion loss correct estimation. In real hardware experiments, the accumulated loss can suffer from extra deviations coming from optical, thermal and electrical crosstalk, introducing deviations and outliers in the loss vs PUCs representations. This signal deviation can be both positive (optical crosstalk in large paths) and negative (non-ideal coupling factor state introducing additional loss).

A first approach to improve the access loss estimation is then performing the fitting using the medians of all accumulated same-length-paths' insertion loss rather than using the paths directly to mitigate the impact of the measurement outliers. In addition, we can set a power threshold to the minimum optical power received to

discard malfunctioning paths or spurious measurements. Note that, however, this threshold should be low enough to detect defective PUCs.

Figure 4.12 illustrates the self-characterization results of a real photonic processor. First, Figure 4.12 (a) illustrates the slopes of all fitted regression lines. We can observe how they lay close 0.50 dB, quite in accordance to specifications obtained during the design stage. Then, Figure 4.12 (b) showcases port characterization results. There, we observe how optical ports 4 and 6 feature large insertion losses (more than twice larger than median). This extra loss can be within the access port or the fiber-to-chip loss. Figure 4.12 (c) portrays the estimated PUC insertion losses across the structure. This plot reveals several malfunctioning PUCs (specially PUC 6); however, none of them exhibit such faulty performance as those simulated in previous chapter. In contrast, there are other several PUCs (1, 3, 7, 9, 34, 48, 57-60, 68 amongst others) showcasing surprisingly low insertion losses ($\leq$ 0.4 dB).
The probability of measuring loss below 0.3 dB is unlikely due to the reproducibility of the fabrication process. For that reason, we can conclude two points. First, the current algorithm accuracy is impacted by system non-idealities and further research is required to get better approximations. Secondly, the current algorithm state is good enough to find both considerably bad-performer ports and/or PUCs to avoid their use during system operation.

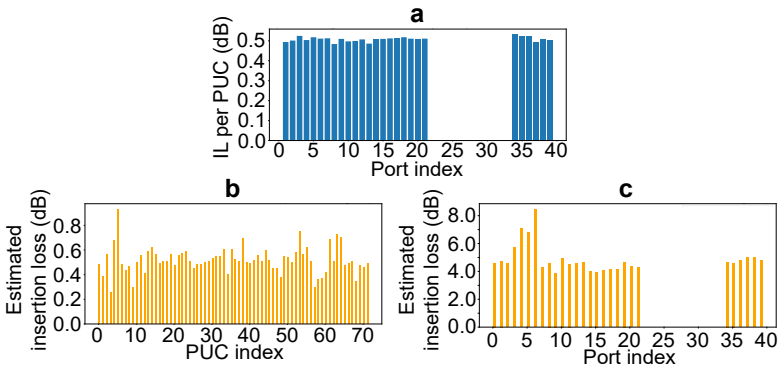


Figure 4.12. Self-characterization results in a real, 72-PUC device (Design B); (a) retrieved slopes of all linear regression fittings, (b) estimated insertion losses for the 72 PUCs of the waveguide mesh, (c) estimated insertion losses for the 28 available optical ports of the waveguide mesh.

4.2.4 Optical switch

Finally, we carried out the synthesis of an optical switch such as the one synthesized at the end of chapter 3 in our Design B. Figure 4.13 and Figure 4.15 represents the coupling factors of each unit cell during the experiment, previously obtained through

Commented [DPL39]: Un
es si al descartar los caminos q
a mapear que una cierta puc est
que pasan por ella?

Commented [ALH40R39]
parece?

Commented [DPL41]: Par
exfo? O el mema interno?

Commented [ALH42R41]

Commented [ALH43]: @

our self-calibration routine. As we do not count with as many PUCs as in our simulated mesh (72 versus 196), we will only configure four and five different circuits for configurations A and B, respectively.

We first start with Config. A. Figure 4.13 displays the mesh state during the synthesis of its four constituent circuits: two point-to-point interconnections between ports 37 and 6 and between ports 2 and 36, a 1x2 beamsplitter using port 37 as input and ports 9 and 10 as outputs, and a 1x4 beamspliter with input at port 35 and optical outputs in channels 16, 17, 19 and 20. Note that, in this configuration, some unit cells (8, 11, 25) are being reused to both power consumption and yield. At the same time, unit cells 36, 39, 64 and 65 are set to tunable coupler state to split the power effectively in the beamsplitters.

Figure 4.14 displays the optical spectra of the generated circuits. The top-left plot, entitled as D1, displays the optical powers measured at output port 2. We observe how the interconnection matching this port with input 36 has been done successfully, featuring an overall insertion loss of around -10 dBm corresponding to the straight, red line above. A second blue, straight line can be distinguished around 36 dB below this spectrum. This line corresponds to the power spectra of D2, the second interconnect between ports 37 and 6. This spurious contribution might correspond to any (small, in any case) inaccuracy during the setting of the coupling factor of PUC 11, which is used by both configurations. This can also be observed while looking at D2 plot (displaying measured spectra at port 6), showing again both spectra in exchanged positions featuring an optical crosstalk of around 34 dB.

A similar performance can be also observed in D3 plot, representing measured spectra at ports 9 and 10. We can observe how both arms of the 1x2 beamsplitter feature a similar output power of around 20 dBm, with less than 1dB of difference between them. Additionally, we can also distinguish the spectrum of previous D2 interconnect slightly above the spectra of the remaining circuits. The reason might lie, again, in the reuse of PUC 25 by both configurations, which must not be featuring an ideal ‘cross’ state. The optical crosstalk is rather good, though -of more than 25 dB.

By last, we can observe the optical spectra of the four outputs of our 1x4 beamsplitter (D4). All of them lie again in the -20 dBm range. The optical crosstalk is slightly larger than in previous scenarios, but note how this time we are no longer able to distinguish any other optical structure even though PUC 8 is shared between this configuration and D1 one. This must be because this PUC lies so far from the target output ports that any spurious contribution from D1 appears to be faded out.

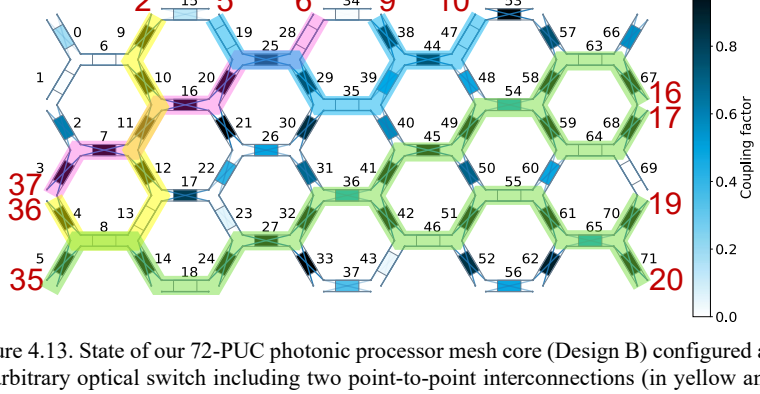


Figure 4.13. State of our 72-PUC photonic processor mesh core (Design B) configured as an arbitrary optical switch including two point-to-point interconnections (in yellow and pink), a 1x2 beamsplitter (in cyan) and a 1x4 beamsplitter (in green).

Now we move to Config. B, represented in Figure 4.15. This configuration includes: a 3-PUC optical interconnect between ports 5 and 6 (D1), a 6-PUC interconnection between ports 10 and 16 (D2), a 1x2 beamsplitter using port 2 as input and ports 36 and 37 as outputs (D3), an optical add-drop multiplexer (OADM) with input at port 20 and outputs in 9 and 35 (D4) and an unbalanced 4-PUC Mach-Zehnder interferometer between ports 17 and 19 (D5).

The power spectra of each synthesized circuit appear in Figure 4.15. Starting from D1 plot, we observe a -10 dBm flat response corresponding to this configuration. No other recognizable spectra are visible in this representation, lying more than 35 dB above the optical crosstalk. Moving on to D2 one, we observe how its power spectra lies again around -10 dBm, again more than 35 dB above the optical crosstalk. Note how the output power of both D1 and D2 spectra are very similar. This could indicate that one or more of the three PUCs forming the D1 interconnect may be malfunctioning, as we would typically expect the D1 spectrum to feature slightly higher optical power than D2 since it traverses fewer PUCs.

D3 spectrum is similar than its counterpart from Config. 1, but showing again a larger optical power. Observe how both beamsplitters traverse the same number of optical PUCs, however, the one from Config. 1 goes through PUCs 19 and 25; two of the three ones used by D1 from Config. 2 and increasing our suspicious about their performance. The optical crosstalk featured for D3 (dominated by the contribution of the through channel of the OADM, since they are sharing PUC 8) is of around 22 dB.

Next goes the configuration of a OADM device (D4), featuring an extinction ratio of 30 dB for the add port (9) and of around 8 dB in its through one (35). Balancing both

Commented [DPL44]: Cre
también para mostrar el caso si
chapter 3? No es extrictamente
demasiadas trazas en el dibujo.

Commented [DPL45R44]
ajustaría el y-lim entre -10 y -

Commented [ALH46R44]
Yo lo dejaría tal y como está. M
procesó Ana y yo no tengo los

outputs extinction ratios was not a concern during this experiment, as we only intended to showcase the splitter performance as a proof-of-concept. Therefore, we directly set PUCs 35 and 36 coupling factors to 0.1 and 0.75, respectively. We also observe how D3 and D5 are the circuits contributing most to enhance optical crosstalk, since they are sharing PUCs 8 (as before mentioned) and 65 with D4.

Finally comes D5 spectrum, showing again an extinction ratio larger than 30 dB and a free spectral range larger than that of the OADM, since it is formed by a cavity of 4 PUCs (59, 54, 49, 50 and 55 in its upper branch and 60 as lower branch) rather than 6.

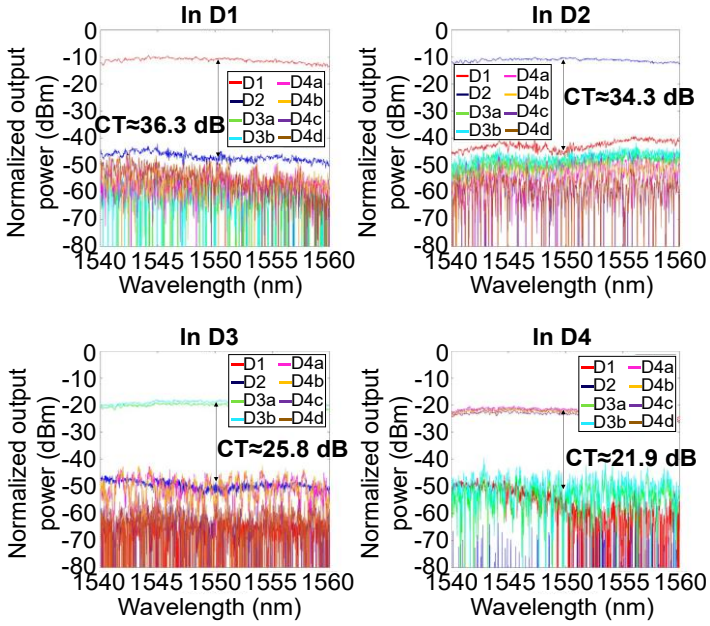


Figure 4.14. Experimental results of our optical switch featuring configuration A. From left to right and from top to bottom: a seven-PUC interconnect between ports 2 and 36 (D1), a seven-PUC interconnect between ports 6 and 37 (D2), a 1x2 beamsplitter using port 5 as input and ports 9 and 10 as outputs (D3) and a 1x4 beamsplitter using port 35 as input and ports 16, 17, 19 and 20 as outputs (D4).

Now we move to Config. B, represented in Figure 4.15. This configuration includes: a 3-PUC optical interconnect between ports 5 and 6 (D1), a 6-PUC interconnection between ports 10 and 16 (D2), a 1x2 beamsplitter using port 2 as input and ports 36

and 37 as outputs (D3), an optical add-drop multiplexer (OADM) with input at port 20 and outputs in 9 and 35 (D4) and an unbalanced 4-PUC Mach-Zehnder —
Figure 4.15. State of our 72-PUC photonic processor mesh core (Design B) configured as an arbitrary optical switch including two point-to-point interconnections (in orange and cyan), a 1x2 beamsplitter (in pink), an optical add-drop multiplexer (in green) and an unbalanced Mach-Zehnder interferometer (in yellow).

The power spectra of each synthesized circuit appear in Figure 4.16. Starting from D1 plot, we observe a -10 dBm flat response corresponding to this configuration. No other recognizable spectra are visible in this representation, lying more than 35 dB above the optical crosstalk. Moving on to D2 one, we observe how its power spectra lies again around -10 dBm, again more than 35 dB above the optical crosstalk. Note how the output power of both D1 and D2 spectra are very similar. This could indicate that one or more of the three PUCs forming the D1 interconnect may be malfunctioning, as we would typically expect the D1 spectrum to feature slightly higher optical power than D2 since it traverses fewer PUCs.

D3 spectrum is similar than its counterpart from Config. 1, but showing again a larger optical power. Observe how both beamsplitters traverse the same number of optical PUCs, however, the one from Config. 1 goes through PUCs 19 and 25; two of the three ones used by D1 from Config. 2 and increasing our suspicious about their performance. The optical crosstalk featured for D3 (dominated by the contribution of the through channel of the OADM, since they are sharing PUC 8) is of around 22 dB.

Next goes precisely our OADM device (D4), featuring an extinction ratio of 30 dB for the add port (9) and of around 8 dB in its through one (35). Balancing both outputs extinction ratios was not a concern during this experiment, as we only intended to showcase the splitter performance as a proof-of-concept. Therefore, we directly set PUCs 35 and 36 coupling factors to 0.1 and 0.75, respectively. We also observe how D3 and D5 are the circuits contributing most to enhance optical crosstalk, since they are sharing PUCs 8 (as before mentioned) and 65 with D4.

Finally comes D5 spectrum, showing again an extinction ratio larger than 30 dB and a free spectral range larger than that of the OADM, since it is formed by a cavity of 4 PUCs (59, 54, 49, 50 and 55 in its upper branch and 60 as lower branch) rather than 6.

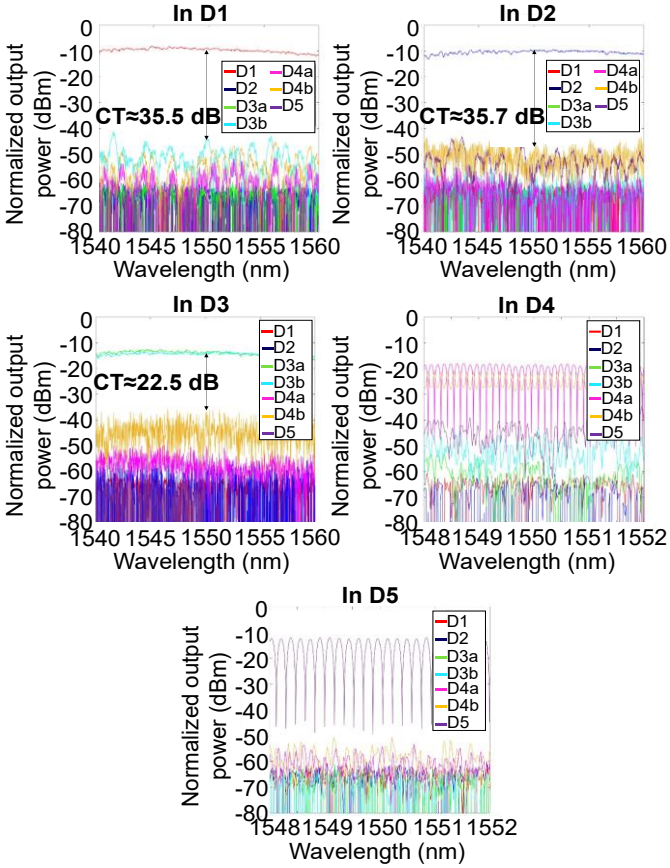


Figure 4.16. Experimental results of our optical switch featuring configuration B. From left to right and from top to bottom: a three-PUC interconnect between ports 5 and 6 (D1), a six-PUC interconnect between ports 10 and 16 (D2), a 1x2 beamsplitter using port 2 as input and ports 36 and 37 as outputs (D3), an optical add-drop multiplexer using port 20 as input and ports 9 and 35 as outputs (D4) and a 4-PUC, unbalanced Mach-Zehnder interferometer using port 17 as input and port 19 as output (D5).

Chapter 5

Summary, Conclusion and Future Work

5.1 Summary and Conclusions

This thesis builds upon the emerging technology of programmable photonic integrated circuits to propose several software-driven strategies to auto-configure, calibrate and characterize these devices, providing a myriad of telecom, sensing and life science applications, amongst others.

In Chapter 2, we introduced self-configuration routine, which allows users to create any optical structure on a programmable photonic processor using two approaches based on pathfinder algorithms and computational algorithms, respectively, For the computation algorithms, the processor adjusts the electrical driving to the core's phase shifter actuators to reach the target mask in an iterative process. Here, we demonstrate its application for the synthesis of optical filters, a 1x8 beamsplitter, and optical interconnections.

Commented [DPL47]: Ha
No ponemos nada del simulado

Commented [ALH48R47]

Chapter 3 presents two other procedures, known as self-calibration and self-characterization, that provide a full-system evaluation based on graph theory. Both approaches rely on a graph-based procedure defined as pathfinder algorithm that allows to retrieve all feasible optical interconnections between any given set of mesh ports within a specified timeframe. Specifically, self-calibration algorithm extracts the injected current-versus-optical power mapping curves for each phase shifter actuator of our photonic processor. With this information, users can directly set any arbitrary coupling factor to the processor's unit cells, which provides thus the possibility to configure optical circuits without recurring to computational optimization. Additionally, knowing the passive state of the photonic cells can lead to significant power savings. Self-characterization routine deals with the estimation of the insertion loss of every constituent component (unit cells and access ports) of our photonic processor. This evaluation offers users an understanding of which circuit areas can be rendered as non-functional and hence discarded for future use. Note that,

despite this, the rest of the circuit is still operational -a big advantage of generic-purpose designs in comparison to application-specific ones. At the end of this chapter, we also present a simulation tool based on pathfinder algorithm that allows the simulation of photonic circuits on any arbitrary mesh structure, a very useful feature for further development of programmable photonic libraries and to speed up the growth of new circuit designs.

While previous chapters only showcase these utilities in a simulated domain, Chapter 4 presents their application to real-world chip designs. Here, we will work with two different designs with their respective set-ups. Chip design A features 30 programmable unit cells with 24 surrounding optical ports, while chip design B consists of 72 programmable unit cells, 28 access ports and 12 additional connections to high-performance optical filters outside the waveguide mesh. Using these tools, we demonstrate the synthesis of a wide variety of optical structures in our photonic processor and the applicability of self-calibration and characterization routines presented in previous chapter.

All in all, we believe that the proposed tools in this thesis constitute a strong foundation towards the design of more complex and sophisticated software functionalities, paving the way towards the use unprecedented applications in this technology such as neuromorphic computing or quantum information processing, between many others.

5.2 Future work

While programmable integrated photonics has relentlessly evolved during recent years from a promising proof of concept to a solid (and even commercial [87]) reality, there are still many open questions and issues pending to be addressed. Here we outline several ones:

5.2.1 Reconfiguration speed

All software routines presented along this work require a compilation time to achieve their required functionality. For most final applications, it would be advantageous to reduce the elapsed time during reconfiguration by increasing the speed and/or minimizing the number of processing, driving, and monitoring operations. Specifically, these operations can be broken down as specified in Figure 5.1.

During each operation, the algorithm provides the next configuration settings based on the current ones plus the readout monitoring data. The process involves the manipulation of said signals and the execution of the optimization algorithm, as described in Chapter 2. Then, these signals are translated and transmitted to the driving electronic circuitry. The response of heater-based, thermo-optics is limited to

2.2-30 μs for Ti-based heaters, 5.6-35 μs for silicon-doped heaters and 65.5-300 μs for Ti-based heaters with under-etched waveguides [88]. Finally, after the readout operation takes place, data follows an analog-to-digital conversion before being transmitted to logic unit for next operation.

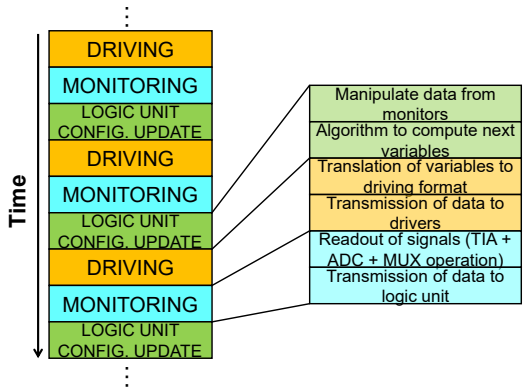


Figure 5.1. Division of the different stages during a single operation of the self-configuration method proposed in Chapter 2.

The processing times of the algorithm vary depending on the number of driving signals (side of the array of variables) and hardware used. With current electronic processor performance, the overall processing time lies in the μs regime for vectors with 10 to 1000 variables. However, data transmission between logic unit, driving and monitoring circuitry can cause significant delays. To avoid bottlenecks in the internal transmission of data required for each operation, different protocols can be employed. For instance, USB 2.0 (with theoretical rates of up to 480 Mbit/s), and USB 3.0 (theoretical rates of up to 4.8 Gbit/s) would enable the data transfer (driving for phase actuators) for 1000 channels in 100 μs and 10 μs, respectively. Alternatively, protocols such as PCIExpress 3.x, 4.0, 5.0 or 6.0 can provide better transfer rates of up to a few GB/s.

Additionally, there are some utilities, such as the synthesis of optical filters, that require the readout from multiple wavelength points. To accomplish this, the system can use a tunable laser, a set of fixed lasers at different wavelengths, or a broadband source in conjunction with a tunable passband filter. In most cases, the tuning speed is comparable to that of the phase shifter in the programmable photonic circuit, but control electronics and laser stabilization is typically limited to the milliseconds regime.

Overall, with considerable engineering efforts, we could assume that the total "operation delay" (involving setting computation, driving and monitoring) can be potentially last in the range of 20-200 μs (5-50 KHz). Moving to the MHz or GHz regime (which might be a requisite for several applications such as optical packet switching) would require the use of alternative tuning mechanisms and the development of specialized integrated electronic circuitry.

5.2.2 Improved convergence

Further programming strategies and optimization methods would benefit the algorithms presented in this work to reduce their required number of operations, accelerating convergence. Some approaches may be:

- Using and combining different optimization methods. Our research suggests that using both global and local search algorithms can significantly improve future scalability and convergence rates of the optimization process.
- Combining the auto-routing algorithm presented in chapter 3 and the self-configuration method from chapter 2 for the synthesis of optical circuits. This approach, already in use during self-calibration routine, reduces the number of variables and thus the optimization search space while maintaining circuit flexibility.
- Employing Principal Component Analysis (PCA) algorithms to remove phase shifters that have minimal impact during optimization. This helps to gradually reduce the number of variables in the optimization process.

5.2.3 Power consumption

The trend in silicon-on-insulator technology for heater evolution shows a reduction in the thermal tuning crosstalk and overall electrical power consumption of tuning elements. This not only benefits the overall circuit power consumption, but also reduces the complexity of the control electronics required for driving purposes. While current technology is advancing towards more robust driving and phase tuning actuators, further improvements to quantify and qualify the robustness of phase tuning technology are still needed to fully consolidate it.

Appendix A

Pseudocodes of graph-based algorithms

A.1Basic path searching through the waveguide mesh

The first procedure to be introduced, `graph_creation`, creates the reciprocal graph representation of any given interferometric structure. To do so, it receives the following parameters:

- `wma_core`: object of class 'WMA' containing all information relative to the waveguide mesh (and, in turn, about each of its constituent PUCs: insertion loss, group index, basic unit delay…)
- `pucs_per_col`: array object containing the number of PUCs in each column of the waveguide mesh. To cite an example, chip design A topology from chapter 4 would be represented as [4, 3, 6, 4, 6, 3, 4] while chip design B would correspond to [6, 3, 6, 4, 6, 3, 6, 4, 6, 3, 6, 4, 6, 3, 6] (note that the sums of the elements of both arrays add up to 30 and 72, the overall number of mesh PUCs in both designs respectively).
- `mesh_architecture`: describes the mesh architecture by a string object ("hexagonal", "rectangular", "triangular", "feed-forward" …). This value will therefore determine how edges connect graph nodes in the same way access waveguides connect PUC optical ports.

To complete its task and as appearing in the pseudocode below, the routine uses several auxiliar methods, here detailed:

- `cells_from_pucs_per_col` takes `pucs_per_col` and `mesh_architecture` as arguments and provides `cells_per_col`, the number of closed polygons per column (in array format) described by the

PUCs, an essential part to define graph's vertices. As an example, for chip designs A and B from chapter 4 the values returned would be [2, 3, 2] and [2, 3, 2, 3, 2, 3, 2], respectively.

- `i_cells_from_cells_per_col` takes `cells_per_col` and `mesh_architecture` as arguments and returns the surrounding 'imaginary' cells in array format, necessary to define perimetric graph vertices. For example, for chip designs A and B from chapter 4 the values returned would be [3, 2, 2, 2, 3] and [3, 2, 2, 2, 2, 2, 2, 2, 3], respectively.
- `get_up_vertices` and `get_low_vertices` methods supply two separate sets of vertices each —all in array format— corresponding to every PUC upper (`up_start_vertices`, `up_end_vertices`) and lower (`low_start_vertices`, `low_end_vertices`) connections respectively.
- `edges_from_vertices` routine returns the `graph_mesh` object per se, a 2D-array with as many columns as graph edges (in both directions) and separate rows to define the input nodes, the destination nodes, the corresponding mesh PUC, its spectral response, its insertion loss, and its basic unit delay for each of them. This matrix is illustrated in Figure A.1.

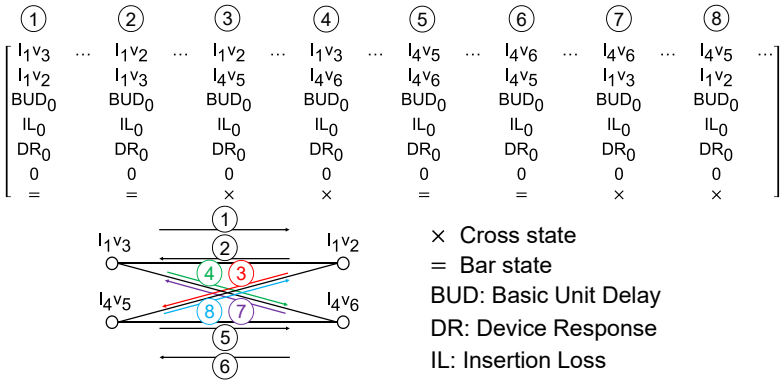


Figure A.1. Matrix form of the graph representation of a 36-PUC waveguide mesh. Each column represents a separate graph edge (in both ways). Columns are filled in the specific order illustrated in the picture below, showcasing the graph representation of PUC 0. Such order has been chosen to facilitate the addition of our extra constraint to avoid using a PUC in two different transmission states (bar, cross) at the same time.

- `make_io_list` returns a list of peripheric mesh nodes (acting as access ports), oriented clockwise from top left corner.

- `create_ports_dictionary` associates one port index to each element of the list of peripheric nodes given by `make_io_list` method.

In turn, this method returns the following variables:

- `graph_mesh`: output of `edges_from_vertices` routine, above explained and illustrated.
- `ports_dictionary`: equivalence in dictionary format between physical optical ports and nodes in graph format. Optical ports are always numbered clockwise from top left side of the mesh.

```
1: PROCEDURE graph_creation, (wma_core, pucs_per_col, mesh_architecture)
2:       CALL cells_from_pucs_per_col(pucs_per_col, mesh_architecture)
         RETURNING cells_per_col
3:       CALL i_cells_from_cells_per_col(cells_per_col, mesh_architecture)
         RETURNING i_cells_per_col
4:       CALL get_up_vertices(cells_per_col, i_cells_per_col, mesh_architecture)
         RETURNING (up_start_vertices, up_end_vertices) # Gets upper vertices for
         each PUC graph representation
5:       CALL get_low_vertices(cells_per_col, i_cells_per_col, mesh_architecture)
         RETURNING (low_start_vertices, low_end_vertices) # Gets lower vertices
         for each PUC graph representation
6:       SET (coupling_factor) FOR EACH puc IN wma_core
7:       pucs_responses = COMPUTE puc_response FOR EACH puc IN wma_core
8:       CALL edges_from_vertices(up_start_verts, up_end_verts, low_start_verts,
         low_end_verts, pucs_responses) RETURNING graph_mesh
9:       CALL make_io_list(graph_mesh, cells_per_col, i_cells_per_col,
         mesh_architecture) RETURNING io_list
10:      CALL create_ports_dictionary(io_list, i_cells_per_column)
         RETURNING ports_dictionary
11:      RETURN (graph_mesh, ports_dictionary)
```

The next method on our list would be `update_paths`. This method provides a list of the updated, remaining paths along with their corresponding accumulated distances (`updated_paths` and `updated_paths_costs` variables) after an iteration of pathfinder algorithm. It requires the following parameters:

- `graph_mesh`: 2D-array returned by `graph_creation` method including the graph representation of the waveguide mesh (illustrated in Figure 3.1).
- `figure_of_merit`: performance metric (accumulated physical distance, insertion loss, power consumption…) with respect to which the path search process takes place.
- `current_paths`: list containing all 'surviving' paths. Each path is also represented as a list, and includes all nodes traversed during path search up to that point in time.

- `current_paths_costs`: array containing the accumulated costs of each path in `current_paths` up to that point in time.
- `threshold` (optional): minimum power threshold allowed for an optical path if the `figure_of_merit` under use is the accumulated insertion loss.

The auxiliary routines employed during this task are below described:

- `get_TD_opposite_tx_state` returns `TD_opposite_tx_state` variable, obtained by looking at the column in `graph_mesh` matrix belonging to `current_node` and `previous_node` variables. The edge arrangement proposed in Figure A.1 facilitates this search. This variable corresponds to the respective four `graph_mesh` matrix columns from the same PUC in opposite transmission state (in every direction).
- `get_neighbors` provides all adjacent edges to a given one, along with their respective transmission distances (including the not-physically-allowed ones, whose transmission distances would be set to infinity).
- `operate_weights` updates the accumulated distance for a given path (`path_dist`) after traversing an adjacent edge with a given cost. The way to operate these values (adding them or multiplying them) would depend on the figure of merit under consideration (basic unit length, PUC spectral response, etc.)

```
1: PROCEDURE update_paths, (graph_mesh, figure_of_merit, current_paths,
                    current_paths_costs, threshold)
2:      SET updated_paths, updated_paths_costs EQUAL to EMPTY LIST
3:      FOR EACH path, path_dist IN current_paths, current_paths_costs
4:              current_node ← path[-1] # We take last element from 'path'
5:              IF LENGTH(path) > 1
6:                      previous_node ← path[-2] # Element prior to
                        current_node in 'path'
7:                      CALL get_TD_opposite_tx_state(graph_mesh, current_node,
                        previous_node) RETURNING deleted_cost
8:                      SET TD_opposite_tx_state EQUAL TO ∞
9:              END IF
10:             CALL get_neighbors(current_node) RETURNING
                neighbors_current_node
11:             FOR EACH neighbor, cost IN neighbors_current_node
12:                     candidate_path ← INCLUDE neighbor IN path
13:                     CALL operate_weights(path_dist, cost, figure_of_merit)
                        RETURNING updated_cost
14:                     IF updated_cost == ∞ OR upated_cost < threshold
15:                             CONTINUE
16:                     ELSE
17:                             INCLUDE candidate_path IN updated_paths
18:                             INCLUDE updated_cost IN updated_paths_costs
19:                     END IF
```

```
20:             END FOR
21:             IF LENGTH(path) > 1
22:                     SET TD_opposite_tx_state EQUAL TO deleted_cost
23:             END IF
24:     END FOR
25:     RETURN (updated_paths, updated_paths_costs)
```

Next comes `pathfinder_bud` ('bud' here referring to 'basic unit delay'). This was the original and simplest version of pathfinder algorithm proposed in [89], focusing on retrieving as many optical connections (represented in `path_list` variable) as possible between a given start and destination nodes (here referred to as `source_node` and `dest_node`) before a given `timeout` expires. Observe that this function uses the `update_paths` one defined previously regularly. Here we assume that the required parameters to make this routine work were previously passed as arguments to avoid obscuring the pseudocode definition more than necessary.

This function also uses the following auxiliary routines:

- As its name states, `get_elapsed_time` method calculates the time that has elapsed between `start_timer()` marker and the moment of its invocation, to check whether it is lower than timeout or not.
- `get_united_paths` takes common_nodes variable, given by the intersection of `path_i` and `path_j` variables coming from both source and destination nodes and, if such variable is not empty (i.e., both paths have intersected anywhere), runs through it and provides all united paths formed from the beginning of `path_i` to each intersection and from there to the beginning of `path_j`.

```
1: PROCEDURE pathfinder_bud, (source_node, dest_node, timeout,)
2:      start_timer()
3:      SET path_list EQUAL to EMPTY LIST
4:      CALL update_paths(figure_of_merit="BUD", current_paths=[source_node])
        RETURNING (paths_source, paths_costs_source)
5:      CALL update_paths(current_paths=[dest_node]) RETURNING (paths_dest,
        paths_costs_dest)
6:      WHILE get_elapsed_time() < timeout
7:              FOR EACH path_i IN paths_source
8:                      FOR EACH path_j IN paths_dest
9:                              common_nodes ← path_i ⋂ path_j
10:                             IF LENGTH(common_nodes) > 0
11:                                     get_united_paths(path_i, path_j,
                                        common_nodes) RETURNING found_path
12:                             END IF
13:                             IF found_path NOT IN path_list
14:                                     INCLUDE found_path IN path_list
15:                             END IF
16:                     END FOR
17:             END FOR
```

```
18:             CALL update_paths(figure_of_merit="BUD", paths_source,
                paths_costs_source) RETURNING paths_source, paths_costs_source)
19:             CALL update_paths(figure_of_merit="BUD", paths_dest,
                paths_costs_dest) RETURNING paths_dest, paths_costs_dest)
20:     END WHILE
21:     RETURN path_list
```

The implementation of pathfinder's additional feature for the synthesis of arbitrary optical spectra presented in [86] between any given pair of optical nodes is also summarized below for the method `pathfinder_spectra`. Note how all the variables and methods taking part of this approach have already been presented previously:

```
1: PROCEDURE pathfinder_spectra, (source_node, dest_node, timeout, threshold)
2:      start_timer()
3:      SET path_list, path_cost_list EQUAL to EMPTY LIST
4:      CALL update_paths(figure_of_merit="IL", current_paths=[source_node],
        threshold)RETURNING(paths, paths_costs)
5:      WHILE get_elapsed_time() < timeout && paths NOT EMPTY
6:              FOR EACH path, path_cost IN paths, paths_costs
7:                      IF path[-1] != dest_node
                                CONTINUE
                        END IF
8:                      IF path NOT IN path_list
9:                              INCLUDE path IN path_list
10:                             INCLUDE path_cost IN path_cost_list
11:                     END IF
12:             END FOR
13:             CALL update_paths(figure_of_merit="IL", paths_source,
                paths_costs_source, threshold) RETURNING paths_source,
                paths_costs_source
14:             END WHILE
15:     RETURN path_list, path_cost_list
```

To represent the corresponding power spectrum, we would only need to sum all the power spectra provided by `path_cost_list` variable, as detailed in subsection 3.3.3.

A.2Waveguide mesh self-calibration

We now move to self-calibration routine pseudocodes. The whole process is summarized in Figure A.2.

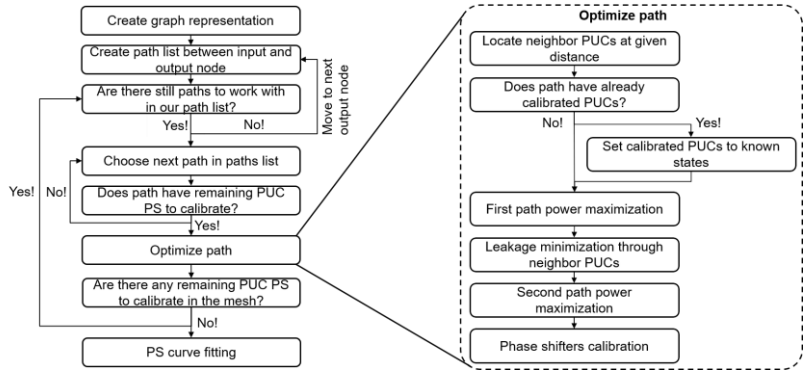


Figure A.2. Block diagram summarizing the whole self-calibration process, from top to bottom.

These operations can be also described through the pseudocode below, presenting `mesh_self-calibration` function. This routine is composed by several sub-routines that will be introduced next after.

```
1: PROCEDURE mesh_self-calibration, (wma_core, in_port)
2:       pending_pucs ← wma_core.pucs # List of pending mesh pucs to calibrate,
         initially including all of them since process has not yet started.
3:       SET calibration_dict EQUAL to EMPTY DICT
4:       CALL graph_creation(wma_core, pucs_per_column,
         mesh_architecture)RETURNING graph_mesh, ports_dictionary
5:       out_ports ← [port FOR port IN ports_dictionary IF port != in_port]
6:       FOR EACH out_port IN out_ports:
7:                CALL pathfinder_bud(in_port, out_port, timeout) RETURNING
                 path_list
8:                FOR EACH path IN path_list:
9:                        pucs_to_calibrate ← pending_pucs ⋂path
10:                       IF LENGTH(pucs_to_calibrate) != 0
11:                               CALL optimize_path(wma_core, path,
                                  calibration_dict)
12:                               CALL ps_calibration(wma_core, path,
                                  calibration_dict) RETURNING calibration_dict
13:                       END IF
14:               END FOR
15:               CALL update_pending_pucs(calibration_dict, pending_pucs)
                  RETURNING pending_pucs # If both phase shifters of any specific
                  PUC have been calibrated, we remove such puc from pending_pucs
                  list.
16:               IF pending_pucs == 0
17:                       BREAK
18:               END IF
19:      END FOR
```

First two steps (creating the graph representation of our mesh and accumulating all paths between an arbitrary port and its adjacent one) were already covered by functions `graph_creation` and `pathfinder_bud` from previous subsection, so let us focus our attention on `optimize_path` method, which aims to optimize the power of a given path (`path_with_states`) connecting input and output nodes.

This method uses several subroutines, listed below:

- `get_path_neighbors` provides a list (`neighbor_list`) of the adjacent PUCs to every PUC forming the path at the specified `neighbor_distance` (in units of PUCs).
- `use_calibration_results` allows to set those path PUCs to fixed transmission states (bar/cross) provided both of their phase shifters have previously successfully calibrated by the algorithm.
- `set_neighbors_to_cross` works similarly than `use_calibration_results`, fixing to cross state every PUC from `list_neighbors` if they were already successfully calibrated to minimize power leakage.
- `path_power_drop` travels through all PUCs constituting the optical path and modifies each of their coupling factors iteratively to reduce the output power by a factor (`puc_power_drop`) specified by user.
- `optimize_power` method deals with power maximization/minimization of path/neighbor PUCs through computational optimization (for more detailes, please refer to Chapter 2). PUCs under use will be defined by `PUC_list`, and the operation to take place will be determined by a `max_flag` boolean that will indicate to carry out power maximization when set to 1 and power minimization otherwise.

```
1: PROCEDURE optimize_path, (wma_core, path_with_states, calibration_dict)
2:       CALL get_path_neighbors(path_with_states, neighbor_distance, graph_mesh)
         RETURNING list_neighbors
3:       CALL use_calibration_results(wma_core, path_with_states,
         calibration_dict)
4:       CALL set_neighbors_to_cross(wma_core, list_neighbors, calibration_dict)
         # Lines 3 and 4 will do nothing the first time optimize_path is used.
5:       CALL optimize_power(wma_core, PUC_list=path_with_states, max_flag=1)
6:       CALL path_power_drop(wma_core, path_with_states, puc_power_drop)
7:       CALL optimize_power(wma_core, PUC_list=list_neighbors, max_flag=0)
8:       CALL optimize_power(wna_core, PUC_list=path_with_states, max_flag=1)
9:       RETURN
```

After the path has been successfully optimized, next comes the individual calibration of each remaining phase shifter, summarized in the `ps_calibration` method pseudocode available below.

```
1: PROCEDURE ps_calibration, (wma_core, path_with_states, calibration_dict)
2:       FOR EACH puc IN path_with_states
3:               FOR EACH phase_shifter IN puc
4:                       IF phase_shifter IN calibration_dict
5:                               CONTINUE
6:                       END IF
7:                       SET opposite_phase_shifter EQUAL to 0
8:                       CALL phase_shifter_sweep(wma_core, phase_shifter)
                         RETURNING ps_calibration_results
9:               INCLUDE ps_calibration_results IN calibration_dict
10:              END FOR
11:      END FOR
12:      RETURN calibration_dict
```

where the method `phase_shifter_sweep` performs a current sweep on the phase shifter actuator and reads the corresponding output power to draw an injected current versus optical power curve such as the ones presented in Figure 3.6, returning a `ps_calibration_results` object indicating the currents that need to be injected on that phase shifter actuator to operate the PUC in bar/cross state.

A.3Waveguide mesh self-characterization

The following functions allow us to estimate the insertion loss of every PUC and access port of the waveguide mesh. The process is illustrated in Figure A.3 and in below pseudocode. We recommend delving into subsections 3.3.1 and 3.3.2 to get a clearer understanding of the working principle of each of the presented sub-routines.

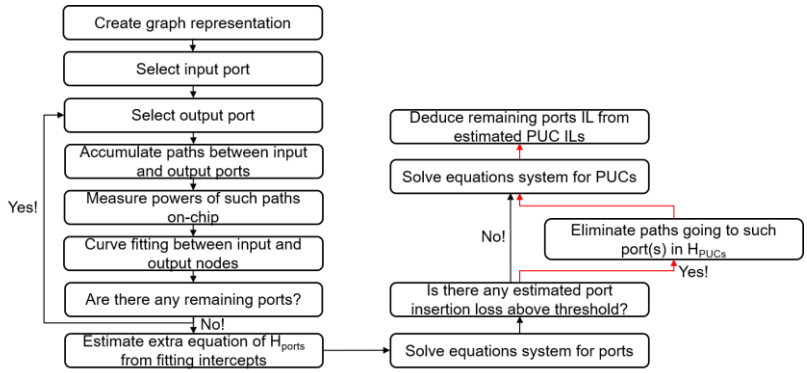


Figure A.3. Block diagram summarizing the whole mesh self-characterization (optical ports + mesh PUCs), from top to bottom.

```
1: PROCEDURE mesh_self-characterization, (wma_core, input_port)
2:       CALL graph_creation(wma_core, pucs_per_column, mesh_architecture)
         RETURNING graph_mesh, ports_dictionary
3:       CALL get_paths_to_all_ports(input_port, graph_mesh, timeout) RETURNING
         paths_to_all_ports
4:       CALL measure_paths_powers(wma_core, paths_to_all_ports) RETURNING
         paths_powers
5:       CALL fitting_to_all_ports(paths_to_all_ports, paths_powers) RETURNING
         intercept_losses, fitting_lines_slopes
6:       CALL create_matrix_ports(intercept_losses, input_port) RETURNING
         matrix_interconnections
7:       CALL get_ports_losses(matrix_interconnections, intercept_losses)
         RETURNING ports_losses
8:       CALL identify_suspiciously_characterized_ports(ports_losses) RETURNING
         suspicious_ports
8:       CALL create_matrix_pucs(paths_to_all_ports, suspicious_ports) RETURNING
         matrix_pucs
9:       CALL remove_coupling_losses_from_path_losses(paths_powers, ports_losses)
         RETURNING paths_losses_wo_coupling_loss
10:      CALL get_pucs_losses(matrix_pucs, paths_losses_wo_coupling_loss)
         RETURNING pucs_losses
11:
12:      RETURN pucs_losses, ports_losses
```

Author's Publication List

Peer-reviewed international Journal Publications

1. A. López, D. Pérez, P. DasMahapatra and J. Capmany, "Auto-routing algorithm for field-programmable photonic gate arrays," Optics Express vol. 28, no. 1, pp. 737-752, 2020.

2. D. Pérez, A. López, P. DasMahapatra and J. Capmany, "Multipurpose self-configuration of programmable photonic circuits," Nature Communications vol. 11, no. 1, pp. 1-11, 2020.

3. E. Sánchez*, A. López* and D. Pérez, "Simulation of highly-coupled programmable photonic circuits," Journal of Lightwave Technology, vol. 40, no. 19 pp. 6423-6434, 2022.

International Conference Publications

1. A. López, D. Pérez, P. DasMahapatra and J. Capmany, "Dynamic reconfiguration in field-programmable photonic arrays," presented at the 45th European Conference on Optical Communication (ECOC), Dublin, Ireland, 23-26 Sept. 2019.

2. D. Pérez, A. López, P. DasMahapatra and J. Capmany, "Field-Programmable Photonic Array for multipurpose microwave photonic applications," presented at the 2019 International Topical Meeting on Microwave Photonics (MWP), Ottawa, Canada, 2019.

3. D. Pérez, A. López, A. Macho, P. DasMahapatra and J. Capmany, "Towards field-programmable photonic gate arrays," presented at Smart Photonic and Optoelectronic Integrated Circuits (SPIE OPTO) XXII, San Francisco, California, USA, 1-6 Feb. 2020.

4. A. López, D. Pérez, P. DasMahapatra and J. Capmany, "Self-reconfigurable Field Programmable Photonic Gate Arrays Using First-order Optimization Techniques," presented at Conference of Lasers and Electro-Optics (CLEO), Washington DC, USA, 10-15 May 2020.

5. A. López*, E. Sánchez* and D. Pérez, "Simulation of an Arbitrary Optical Switch on a Dense Programmable Photonic Processor," presented at the 48th European Conference on Optical Communication (ECOC), Basel, Switzerland, 18-22 Sept. 2022.

6. A. López, M. Gutiérrez and D. Pérez, “Automatic self-calibration of Programmable Photonic Processors,” presented at the IEEE Photonics Conference (IPC), Vancouver, Canada, 13-17 Nov. 2022.

7. D. Pérez, A. López et. al., “Dynamically-Reconfigurable Photonic Integrated Circuits and Its Control Algorithms,” to be presented at Conferencce of Lasers and Electro-Optics (CLEO), San Jose, USA, 7-12 May 2023.

Bibliography

[1] A. N. Saxena, *Invention of integrated circuits: Untold important facts*. Singapore, Singapore: World Scientific Publishing, 2009.

[2] R. N. Noyce, "Semiconductor device-and-lead structure, reprint of U.s. patent 2,981,877 (issued April 25, 1961. Filed July 30, 1959)," *IEEE Solid-State Circuits Soc. Newsl.*, vol. 12, no. 2, pp. 34–40, 2007.

[3] K. Grifantini, "Moore's Law," *MIT Technology Review*, 22-Dec-2008. [Online]. Available: https://www.technologyreview.com/2008/12/22/216817/moores-law/. [Accessed: 08-Mar-2023].

[4] M. M. Atalla, E. Tannenbaum, and E. J. Scheibner, "Stabilization of silicon surfaces by thermally grown oxides," *Bell Syst. tech. j.*, vol. 38, no. 3, pp. 749–783, May 1959.

[5] F. Wanlass and C. Sah, "Nanowatt logic using field-effect metal-oxide semiconductor triodes," in *1963 IEEE International Solid-State Circuits Conference. Digest of Technical Papers*, Philadelphia, PA, USA, 1963.

[6] B. Lojek, *History of semiconductor engineering*. Berlin, Germany: Springer, 2010.

[7] S. Voinigescu, *High-frequency integrated circuits*. Cambridge, England: Cambridge University Press (Virtual Publishing), 2014.

[8] R. K. Ratnesh *et al.*, "Advancement and challenges in MOSFET scaling," *Mater. Sci. Semicond. Process.*, vol. 134, no. 106002, p. 106002, Nov. 2021.

[9] R. H. Dennard, F. H. Gaensslen, H.-N. Yu, V. L. Rideout, E. Bassous, and A. R. LeBlanc, "Design of ion-implanted MOSFET's with very small physical dimensions," *IEEE J. Solid-State Circuits*, vol. 9, no. 5, pp. 256–268, Oct. 1974.

[10] G. E. Moore, "Cramming more components onto integrated circuits, Reprinted from Electronics, volume 38, number 8, April 19, 1965, pp.114 ff," *IEEE Solid-State Circuits Soc. Newsl.*, vol. 11, no. 3, pp. 33–35, Sep. 2006.

[11] "Transistor count," *Wikipedia*, 04-Mar-2023. [Online]. Available: https://en.wikipedia.org/wiki/Transistor_count. [Accessed: 08-Mar-2023].

[12] B. Wang, "Samsung Targets Mass Production of 2 nm by 2025 and 1.4 nm by 2027," *nextBigFuture*, 10-Oct-2022. .

[13] R. Courtland, "Transistors Will Stop Shrinking in 2021, Moore's Law Roadmap Predicts," *IEEE Spectrum*, 22-Jul-2016. [Online]. Available: https://spectrum.ieee.org/transistors-will-stop-shrinking-in-2021-moores-law-roadmap-predicts#:~:text=The%20trajectory%20of%20transistor%20feature,a%20sharp

%20turn%20in%202021.&text=After%20more%20than%2050%20years,shrinking%20in%20just%20five%20years. [Accessed: 04-Jun-2019].

[14] S. W. Keckler, K. Olukotun, and H. P. Hofstee, Eds., *Multicore Processors and Systems*, 2009th ed. New York, NY: Springer, 2012.

[15] K. Dhananjay, P. Shukla, V. F. Pavlidis, A. Coskun, and E. Salman, "Monolithic 3D integrated circuits: Recent trends and future prospects," *IEEE Trans. Circuits Syst. II Express Briefs*, vol. 68, no. 3, pp. 837–843, Mar. 2021.

[16] S. K. Selvaraja and P. Sethi, "Review on Optical Waveguides," in *Emerging Waveguide Technology*, InTech, 2018.

[17] G. P. Agrawal, *Fiber-optic communication systems*, 3rd ed. Nashville, TN: John Wiley & Sons, 2002.

[18] K. C. Kao and G. A. Hockham, "Dielectric-fibre surface waveguides for optical frequencies," *Proc. Inst. Electr. Eng.*, vol. 113, no. 7, pp. 1151–1158, Jul. 1966.

[19] R. A. Soref and J. P. Lorenzo, "Single-crystal silicon: a new material for 1.3 and 1.6 μm integrated-optical components," *Electron. Lett.*, vol. 21, no. 21, p. 953, 1985.

[20] A. Rahim *et al.*, "Taking silicon photonics modulators to a higher performance level: state-of-the-art and a review of new technologies," *Adv. Photonics*, vol. 3, no. 02, Apr. 2021.

[21] X. Mu, S. Wu, L. Cheng, and H. Y. Fu, "Edge couplers in silicon photonic integrated circuits: A review," *Appl. Sci. (Basel)*, vol. 10, no. 4, p. 1538, Feb. 2020.

[22] L. Cheng, S. Mao, Z. Li, Y. Han, and H. Y. Fu, "Grating couplers on silicon photonics: Design principles, emerging trends and practical issues," *Micromachines (Basel)*, vol. 11, no. 7, p. 666, Jul. 2020.

[23] Z. Wang *et al.*, "Novel light source integration approaches for silicon photonics," *Laser Photon. Rev.*, vol. 11, no. 4, p. 1700063, Jul. 2017.

[24] R. Jones *et al.*, "Heterogeneously integrated InP\/silicon photonics: Fabricating fully functional transceivers," *IEEE Nanotechnol. Mag.*, vol. 13, no. 2, pp. 17–26, Apr. 2019.

[25] L. Vivien *et al.*, "42 GHz p.i.n Germanium photodetector integrated in a silicon-on-insulator waveguide," *Opt. Express*, vol. 17, no. 8, pp. 6252–6257, Apr. 2009.

[26] D. Benedikovic *et al.*, "Silicon–germanium receivers for short-wave-infrared optoelectronics and communications," *Nanophotonics*, vol. 10, no. 3, pp. 1059–1079, Jan. 2021.

[27] O. Guirdham, "Global Silicon Photonics Market Growth Trajectory," *EIN Presswire*, 20-Dec-2022. [Online]. Available:

https://www.einnews.com/pr_news/607263931/global-silicon-photonics-market-growth-trajectory. [Accessed: 26-Dec-2022].

[28] A. Rahim, T. Spuesens, R. Baets, and W. Bogaerts, "Open-access silicon photonics: Current status and emerging initiatives," *Proc. IEEE Inst. Electr. Electron. Eng.*, vol. 106, no. 12, pp. 2313–2330, Dec. 2018.

[29] J. Klamkin *et al.*, "Indium Phosphide Photonic Integrated Circuits: Technology and Applications," in *2018 IEEE BiCMOS and Compound Semiconductor Integrated Circuits and Technology Symposium (BCICTS)*, San Diego, CA, 2018.

[30] D. J. Blumenthal, R. Heideman, D. Geuzebroek, A. Leinse, and C. Roeloffzen, "Silicon Nitride in Silicon Photonics," *Proc. IEEE Inst. Electr. Electron. Eng.*, vol. 106, no. 12, pp. 2209–2231, Dec. 2018.

[31] A. Rahim *et al.*, "Expanding the silicon photonics portfolio with silicon nitride photonic integrated circuits," *J. Lightwave Technol.*, vol. 35, no. 4, pp. 639–649, Feb. 2017.

[32] W. Bogaerts and L. Chrostowski, "Silicon photonics circuit design: Methods, tools and challenges," *Laser Photon. Rev.*, vol. 12, no. 4, p. 1700237, Apr. 2018.

[33] W. Bogaerts and A. Rahim, "Programmable Photonics: An Opportunity for an Accessible Large-Volume PIC Ecosystem," *IEEE J. Sel. Top. Quantum Electron.*, vol. 26, no. 5, pp. 1–17, Sep. 2020.

[34] L. Carroll *et al.*, "Photonic packaging: Transforming silicon photonic integrated circuits into photonic devices," *Appl. Sci. (Basel)*, vol. 6, no. 12, p. 426, Dec. 2016.

[35] S. M. Trimberger, "Three ages of FPGAs: A retrospective on the first thirty years of FPGA technology," *Proc. IEEE Inst. Electr. Electron. Eng.*, vol. 103, no. 3, pp. 318–331, Mar. 2015.

[36] C. Minkenberg, G. Rodriguez, and N. Kucharewski, "Redefining the economics of reach with integrated optics," in *2018 IEEE 19th International Conference on High Performance Switching and Routing (HPSR)*, Bucharest, Romania, 2018.

[37] D. Inniss and R. Rubenstein, *Silicon photonics*. Oxford, England: Morgan Kaufmann, 2016.

[38] J. Capmany and D. Perez, *Programmable Integrated Photonics*. London, England: Oxford University Press, 2020.

[39] D. Pérez, I. Gasulla, and J. Capmany, "Field-programmable photonic arrays," *Opt. Express*, vol. 26, no. 21, pp. 27265–27278, Oct. 2018.

[40] R. Soref and B. Bennett, "Electrooptical effects in silicon," *IEEE J. Quantum Electron.*, vol. 23, no. 1, pp. 123–129, Jan. 1987.

[41] M. W. Pruessner, D. Park, D. A. Kozak, T. H. Stievater, and W. S. Rabinovich, "Effective index tuning in micro-opto-mechanical structures using gradient

electric forces," in *2016 IEEE Photonics Society Summer Topical Meeting Series (SUM)*, 2016, pp. 164–165.

[42] M. Stegmaier, C. Ríos, H. Bhaskaran, and W. H. P. Pernice, "Thermo-optical effect in phase-change nanophotonics," *ACS Photonics*, vol. 3, no. 5, pp. 828–835, May 2016.

[43] D. Pérez-López, A. M. Gutierrez, E. Sánchez, P. DasMahapatra, and J. Capmany, "Integrated photonic tunable basic units using dual-drive directional couplers," *Opt. Express*, vol. 27, no. 26, pp. 38071–38086, Dec. 2019.

[44] L. Zhuang, C. G. H. Roeloffzen, M. Hoekman, and K. J. Boller, "Programmable photonic signal processor chip for radiofrequency applications," *Optica*, 2015.

[45] M. Reck, A. Zeilinger, H. J. Bernstein, and P. Bertani, "Experimental realization of any discrete unitary operator," *Phys. Rev. Lett.*, vol. 73, no. 1, pp. 58–61, Jul. 1994.

[46] D. A. B. Miller, "Self-configuring universal linear optical component [Invited]," *Photonics Res.*, vol. 1, no. 1, p. 1, Jun. 2013.

[47] W. R. Clements, P. C. Humphreys, B. J. Metcalf, W. S. Kolthammer, and I. A. Walsmley, "Optimal design for universal multiport interferometers," *Optica*, vol. 3, no. 12, p. 1460, Dec. 2016.

[48] D. Pérez, I. Gasulla, J. Capmany, and R. A. Soref, "Reconfigurable lattice mesh designs for programmable photonic processors," *Opt. Express*, vol. 24, no. 11, pp. 12093–12106, May 2016.

[49] D. Pérez *et al.*, "Multipurpose silicon photonics signal processor core," *Nat. Commun.*, vol. 8, no. 1, p. 636, Sep. 2017.

[50] D. P. Lopez, "Programmable integrated silicon photonics waveguide meshes: Optimized designs and control algorithms," *IEEE J. Sel. Top. Quantum Electron.*, vol. 26, no. 2, pp. 1–12, Mar. 2020.

[51] W. Bogaerts *et al.*, "Programmable photonic circuits," *Nature*, vol. 586, no. 7828, pp. 207–216, Oct. 2020.

[52] D. Pérez, I. Gasulla, P. Das Mahapatra, and J. Capmany, "Principles, fundamentals, and applications of programmable integrated photonics," *Adv. Opt. Photonics*, vol. 12, no. 3, p. 709, Sep. 2020.

[53] J. Capmany and D. Novak, "Microwave photonics combines two worlds," *Nat. Photonics*, vol. 1, no. 6, pp. 319–330, Jun. 2007.

[54] D. Marpaung, J. Yao, and J. Capmany, "Integrated microwave photonics," *Nat. Photonics*, vol. 13, no. 2, pp. 80–90, Feb. 2019.

[55] W. Zhang and J. Yao, "Photonic integrated field-programmable disk array signal processor," *Nat. Commun.*, vol. 11, no. 1, p. 406, Jan. 2020.

[56] B. G. Lee and N. Dupuis, “Silicon photonic switch fabrics: Technology and architecture,” *J. Lightwave Technol.*, vol. 37, no. 1, pp. 6–20, Jan. 2019.

[57] Q. Cheng, S. Rumley, M. Bahadori, and K. Bergman, “Photonic switching in high performance datacenters [Invited],” *Opt. Express*, vol. 26, no. 12, pp. 16022–16043, Jun. 2018.

[58] A. Wonfor, H. Wang, R. V. Penty, and I. H. White, “Large port count high-speed optical switch fabric for use within datacenters [invited],” *J. Opt. Commun. Netw.*, vol. 3, no. 8, p. A32, Aug. 2011.

[59] N. Thomas-Peter *et al.*, “Integrated photonic sensing,” *New J. Phys.*, vol. 13, no. 5, p. 055024, May 2011.

[60] M. C. Estevez, M. Alvarez, and L. M. Lechuga, “Integrated optical devices for lab-on-a-chip biosensing applications,” *Laser Photon. Rev.*, vol. 6, no. 4, pp. 463–487, Jul. 2012.

[61] R. Heideman, M. Hoekman, and E. Schreuder, “TriPleX-based integrated optical ring resonators for lab-on-a-chip and environmental detection,” *IEEE J. Sel. Top. Quantum Electron.*, vol. 18, no. 5, pp. 1583–1596, Sep. 2012.

[62] J. Carolan *et al.*, “QUANTUM OPTICS. Universal linear optics,” *Science*, vol. 349, no. 6249, pp. 711–716, Aug. 2015.

[63] N. C. Harris *et al.*, “Large-scale quantum photonic circuits in silicon,” *Nanophotonics*, vol. 5, no. 3, pp. 456–468, Aug. 2016.

[64] A. Ben Abdallah and K. N. Dang, *Neuromorphic computing principles and organization*, 1st ed. Cham, Switzerland: Springer Nature, 2022.

[65] Y. Shen *et al.*, “Deep learning with coherent nanophotonic circuits,” *Nat. Photonics*, vol. 11, no. 7, pp. 441–446, Jul. 2017.

[66] G. Sarantoglou, A. Bogris, C. Mesaritakis, and S. Theodoridis, “Bayesian photonic accelerators for energy efficient and noise robust neural processing,” *IEEE J. Sel. Top. Quantum Electron.*, vol. 28, no. 6, pp. 1–10, Nov. 2022.

[67] H.-T. Peng, M. A. Nahmias, T. F. de Lima, A. N. Tait, and B. J. Shastri, “Neuromorphic Photonic Integrated Circuits,” *IEEE J. Sel. Top. Quantum Electron.*, vol. 24, no. 6, pp. 1–15, Nov. 2018.

[68] H. Zhou, Y. Zhao, X. Wang, D. Gao, J. Dong, and X. Zhang, “Self-learning photonic signal processor with an optical neural network chip,” *arXiv [eess.SP]*, 18-Feb-2019.

[69] G. Giamougiannis *et al.*, “Universal linear optics revisited: New perspectives for neuromorphic computing with silicon photonics,” *IEEE J. Sel. Top. Quantum Electron.*, vol. 29, no. 2: Optical Computing, pp. 1–16, Mar. 2023.

[70] D. P. López, “Programmable integrated silicon photonics waveguide meshes: optimized designs and control algorithms,” *IEEE J. Sel. Top. Quantum Electron.*, 2019.

[71] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, Nagoya, Japan, 2002.

[72] E. T. Oldewage, A. P. Engelbrecht, and C. W. Cleghorn, "The merits of velocity clamping particle swarm optimisation in high dimensional spaces," in *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, Honolulu, HI, 2017.

[73] D. Pérez-López, A. López, P. DasMahapatra, and J. Capmany, "Multipurpose self-configuration of programmable photonic circuits," *Nat. Commun.*, vol. 11, no. 1, p. 6359, Dec. 2020.

[74] C. K. Madsen and J. H. Zhao, *Optical Filter Design and Analysis*. Wiley, 1999.

[75] Y. Xie, L. Zhuang, and A. J. Lowery, "Picosecond optical pulse processing using a terahertz-bandwidth reconfigurable photonic integrated circuit," *Nanophotonics*, vol. 7, no. 5, pp. 837–852, May 2018.

[76] M. Burla, "Advanced integrated optical beam forming networks for broadband phased array antenna systems," PhD, University of Twente, 2013.

[77] D. Pérez and J. Capmany, "Scalable analysis for arbitrary photonic integrated waveguide meshes," *Optica*, 2019.

[78] R. J. Trudeau, *Introduction to graph theory*, 2nd ed. Mineola, NY: Dover Publications, 1994.

[79] F. Zhou, Q. Yang, T. Zhong, D. Chen, and N. Zhang, "Variational graph neural networks for road traffic prediction in intelligent transportation systems," *IEEE Trans. Industr. Inform.*, vol. 17, no. 4, pp. 2802–2812, Apr. 2021.

[80] M. Grandjean, "A social network analysis of Twitter: Mapping the digital humanities community," *Cogent Arts Humanit.*, vol. 3, no. 1, p. 1171458, Dec. 2016.

[81] D. G. Bonchev and D. H. Rouvray, *Chemical graph theory*. London, England: Taylor & Francis, 1991.

[82] N. Deo, *Graph theory with applications to engineering and computer science*, 1st ed. Mineola, NY: Dover Publications, 2016.

[83] X. Chen, P. Stroobant, M. Pickavet, and W. Bogaerts, "Graph Representations for Programmable Photonic Circuits," *J. Lightwave Technol.*, vol. 38, no. 15, pp. 4009–4018, Aug. 2020.

[84] S. S. Skiena, *The Algorithm Design Manual*, 2nd ed. London, England: Springer, 2011.

[85] X. Xu *et al.*, "Self-calibrating programmable photonic integrated circuits," *Nat. Photonics*, Jul. 2022.

[86] E. Sanchez, A. Lopez, and D. Perez-Lopez, “Simulation of highly coupled programmable photonic circuits,” *J. Lightwave Technol.*, pp. 1–12, 2022.

[87] “iPronics delivers first reconfigurable photonic microchips,” *Electrooptics*, 02-Aug-2023. [Online]. Available: https://www.electrooptics.com/news/ipronics-delivers-first-reconfigurable-photonic-microchips. [Accessed: 02-Sep-2023].

[88] M. Jacques, A. Samani, E. El-Fiky, D. Patel, Z. Xing, and D. V. Plant, “Optimization of thermo-optic phase-shifter design and mitigation of thermal crosstalk on the SOI platform,” *Opt. Express*, vol. 27, no. 8, pp. 10456–10471, Apr. 2019.

[89] A. López, D. Pérez, P. DasMahapatra, and J. Capmany, “Auto-routing algorithm for field-programmable photonic gate arrays,” *Opt. Express*, vol. 28, no. 1, pp. 737–752, Jan. 2020.