Accurate Infinite-order Crosstalk Calculation for Optical Networks-on-Chip

Alexandre Truppel, Tsun-Ming Tseng, *Member, IEEE*, and Ulf Schlichtmann, *Senior Member, IEEE*

***Abstract*—Optical Networks-on-Chip (ONoCs) are becoming more and more appealing due to rising network requirements in integrated circuits. One important aspect in ONoC design is the amount of crosstalk generated by the network because crosstalk and Signal-to-Noise Ratio (SNR) are strong limiting factors to network scale and performance. Calculation of crosstalk is thus essential for effective ONoC design. Motivated by this fact, we developed a new general linear algebra based method to calculate the steady-state of any system obeying a simple set of rules. We prove that ONoCs follow these rules and show how this general method can be applied in the particular case of ONoCs to calculate accurate first-order and infinite-order crosstalk and SNR results for any ONoC network.**

***Index Terms*—Silicon photonics, optical networks, linear algebra, crosstalk.**

I. Introduction

Current Integrated Circuits (ICs) are complex enough that full networking infrastructures are required to connect their system-on-chip components [1]. These Networks-on-Chip (NoCs) have so far been built using electrical connections, but NoCs are becoming more and more insufficient with the increase in processing power of ICs [2]. Optical Networks-on-Chip (ONoCs) have been proposed as a replacement to traditional electrical NoCs given their higher expected network performance [3], [4]. ONoCs use optical wires (waveguides) and light instead of electricity to transmit information.

One key design aspect of ONoCs is crosstalk. Many physical ONoC elements produce optical noise that disturbs the detection of the optical signals carrying information. This effect is worsened as the ONoC designs scale up and more light wavelengths are required. The Signal-to-Noise Ratio (SNR) at the optical demodulators of an ONoC must be high enough for communication to be effective. Thus, calculating noise levels and SNR for ONoC designs is an essential step to designing ONoCs with crosstalk in mind [5].

Current methods to calculate crosstalk only consider first-order noise, that is, noise caused directly by signals. However, this first-order noise goes on to create second, third and higher orders of noise as it travels through the ONoC, which worsens the SNR. They also commonly only use approximate values for the crosstalk factors [5]–[9]. In this work we present a linear algebra method based on flow graphs to calculate first-order and infinite-order incoherent noise levels and SNR values for any ONoC design using either approximate or accurate crosstalk factors. Our method can calculate the total amount of noise generated or discriminate the results on a signal by signal basis. Furthermore, the mathematical backbone of this method can be applied to any system fulfilling a simple set of rules, not just ONoCs.

This work is organised as follows. Section II introduces previous publications on ONoC crosstalk calculation. Section III establishes the mathematical foundation of this work and section IV follows with a matrix reduction technique that is useful to increase this method's performance. These two sections stand on their own and their utility extends beyond crosstalk calculation for ONoCs, so they are presented first. Section V then describes the working principles of ONoCs, insertion loss and crosstalk generation. Section VI explains how the principles from section III are applied to calculate SNR in ONoCs. Section VII touches on a few practical considerations of this method, namely how the reduction technique in section IV increases performance. Finally, section VIII presents results obtained by this method and section IX concludes this work.

II. Related work on ONoC crosstalk calculation

Multiple works have tackled the calculation of ONoC crosstalk [5]–[9]. However, these state of the art methods calculate only first-order noise (calculating higher orders of noise is considered to be computationally difficult [8]), operate only on specific ONoC designs and consider only approximate (average) versions of the crosstalk coefficients.

The method presented in this work is, to our knowledge, the first capable of lifting all of these restrictions: it can calculate both first-order and infinite-order noise for ONoCs, it can do so for any active or passive ONoC design and it can use both the approximate (average) and the accurate (wavelength-dependent) versions of the crosstalk coefficients.

III. Steady-state analysis of a flow graph

We start by describing a general version of our method. This general version can be applied to any system composed of multiple elements that obeys the following rules:

1) Each element holds some amount of a quantifiable property of the system. In this work we call this property energy.
2) Some elements may be sources or sinks of energy.
3) Fixed ratios of energy are transferred between elements over time.
4) If there are no active source elements inserting energy into the system, the total amount of energy in the system decreases to zero over time.

Alexandre Truppel, Tsun-Ming Tseng, and Ulf Schlichtmann are with the Institute for Electronic Design Automation, Technical University of Munich, Arcisstr. 21, Munich 80333, Germany (e-mail: alex.truppel@tum.de; tsun-ming.tseng@tum.de; ulf.schlichtmann@tum.de).

This work is supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) Project Number 439798838.

JLT DOI: 10.1109/JLT.2022.3210159
https://ieeexplore.ieee.org/document/9904824

Under the condition that all sources of energy input a constant amount of energy per unit of time into the system forever, the goal is to calculate the distribution of energy over the system's elements (the state of the system) after an infinite amount of time has passed, i.e. its steady-state.

For a system with n elements, consider a weighted directed graph $G = (V, E)$, abbreviated as flow graph in this work, with n vertices and edge weights $w(e)\ \forall e \in E$. The graph structure can be described by an adjacency matrix $A \in \mathbb{R}^{n\times n}$ where:

$$[A]_{ij} = \begin{cases} 1 & \text{if } (i,j) \in E \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

A transfer matrix $T \in \mathbb{R}^{n\times n}$ can be created based on the transpose of the adjacency matrix A and the edge weights $w(e)$

$$[T]_{ij} = w(j,i) \tag{2}$$

where we extend the definition of $w(e)\ \forall e \in E$ with $w(e) = 0\ \forall e \notin E$ for simplicity. This matrix fully defines the flow graph.

We associate each vertex $v \in V$ with a state variable x_v. Both the state variables and the edge weights are real and non-negative. The weights are constant but the state variables are time-varying so we index them on discrete time steps: $x_v[k]$. Each state variable $x_v[k]$ indicates how much energy is flowing through vertex v at time instant k. The propagation of energy through each vertex v in the graph can be described by the following relationship:

$$x_v[k+1] = \sum_{i\in V} w(i,v)\, x_i[k] \tag{3}$$

In other words, some fraction of the energy currently flowing at time k through vertices whose outgoing edges end at v will be propagated at time $k+1$ to vertex v, and some fraction of the energy flowing at time k through vertex v will be propagated at time $k+1$ to the vertices where the outgoing edges of v end at. An example is shown in fig. 1.

Note that the energy in a vertex v is not necessarily propagated in full to the rest of the system. Define $f_v = \sum_{i\in V} w(v,i)$. Vertex v is a lossless vertex if $f_v = 1$, a partial sink vertex if $f_v \in]0,1[$ and a full sink vertex if $f_v = 0$.

If all state variables $x_v[k]$ are combined into a state vector $x[k] \in \mathbb{R}^n$, the propagation relationship in eq. (3) can be written simply as:

$$x[k+1] = Tx[k] \tag{4}$$

That is, the transfer matrix T propagates energy one step forward in the flow graph.

Some vertices may be energy sources which insert energy into the system. Define δ_k as a one-hot column vector:

$$[\delta_k]_i = \begin{cases} 1 & \text{if } i = k \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

If one source vertex v produces a pulse (i.e. lasting only one time step) of energy with amount g, the addition of energy into the system can be expressed in a vector $b \in \mathbb{R}^n$ with $b = g\delta_v$. If the system starts with no energy, i.e. $x[k<0] = 0$, and the pulse is produced at $k = 0$, then $x[0] = b$. To calculate the propagation of energy through the graph we multiply by T, resulting in the relationship $x[k] = Tx[k-1] = T^k x[0]$.

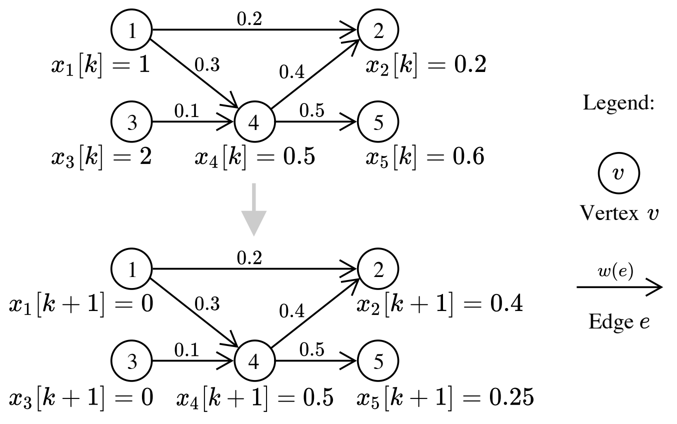

Figure 1. Example of how energy is propagated through the flow graph over one time step, as described by eq. (3).

Since multiple outgoing edges may exist for each vertex, energy that starts as a pulse in one vertex v spreads out through the graph over time. By multiplying by T^k we are not calculating one single propagation path of energy k steps through the graph, but instead a wavefront of energy propagating in parallel through all possible k-step paths starting at v in the graph.

In general, multiple source vertices may produce a pulse on the same time step. Thus, the complete definition of vector b is

$$b = \sum_{v\in V} g(v)\delta_v \tag{6}$$

where $g(v)$ is the amount of energy per pulse released by vertex v. We have $g(v) > 0$ for source vertices and $g(v) = 0$ otherwise.

We now formalise this method's goal as explained in the beginning of this section:

- Assume $x[k<0] = 0$.
- One or more sources are turned on at $k = 0$, each producing a constant amount of energy continuously (i.e. on each time step) forever. The release of energy into the system on each time step is defined in eq. (6).
- Calculate $x[\infty]$, that is, the steady-state value of the state vector x.

We can calculate $x[\infty]$ step-by-step. We start with $x[0] = b$. At $k = 1$, the energy from $k = 0$ has been propagated forward once and a new amount b of energy has been produced by the sources, resulting in:

$$x[1] = Tx[0] + b = Tb + b = (T + I)b \tag{7}$$

At $k = 2$, we repeat the process:

$$x[2] = Tx[1] + b = (T^2 + T + I)b \tag{8}$$

Clearly, we conclude that:

$$x[\infty] = \left(\sum_{i=0}^{\infty} T^i\right) b \tag{9}$$

This result is sensible: the steady-state amount of energy in each vertex is the sum of the contributions of all possible paths of energy of all lengths (between 0 and ∞ steps) starting at each active source (the non-zero entries in b).

Calculating $x[\infty]$ using an infinite sum of powers is not reasonable in practice. Fortunately, the well-known closed-form formula for the geometric series with a real number r

$$\sum_{i=0}^{\infty} r^i = \frac{1}{1-r} \quad \forall r \in \mathbb{R} : |r| < 1 \tag{10}$$

has a matrix analogue

$$\sum_{i=0}^{\infty} M^i = (I-M)^{-1} \quad \forall M \in \mathbb{R}^{n\times n} : \rho(M) < 1 \tag{11}$$

which is true for all square matrices M whose spectral radius $\rho(M) < 1$.

Define $\hat{M}$ as shorthand for $I - M$ for all matrices M. With the use of theorem 1, the calculation of $x[\infty]$ is reduced to solving a single linear system of equations:

$$x[\infty] = \left(\sum_{i=0}^{\infty} T^i\right) b \;\Leftrightarrow\; x[\infty] = (I-T)^{-1} b \;\Leftrightarrow\; \hat{T} x[\infty] = b \tag{12}$$

Note that the convergence of $x[\infty]$ does not depend on how much energy is input into the system, i.e. the amplitude of b. Convergence is guaranteed just based on b being finite and constant and $\rho(T) < 1$.

Section VI will explain how eq. (12) can be used to calculate insertion loss and crosstalk on ONoCs.

Theorem 1: The transfer matrix T satisfies the condition $\rho(T) < 1$.

Proof: Let $E\left(x[k]\right)$ be the total energy in the system given a state-vector $x[k]$ at time instant k. Then

$$E\left(x[k]\right) = \sum_{i=1}^{n} [x[k]]_i = \sum_{i=1}^{n} |[x[k]]_i| = \|x[k]\|_1 \tag{13}$$

since all the entries in x are non-negative.

If the system has the state $x[k]$ at time instant k, the total energy in the system after r time steps is:

$$E\left(x[k+r]\right) = E\left(T^r x[k]\right) = \|T^r x[k]\|_1 \tag{14}$$

Since the total energy in a system with no active source vertices must eventually decrease to zero (rule 4 of the system), there must be a value $r \in \mathbb{Z}^+$ of propagation steps where the following is true for all[1] state vectors $x[k]$:

$$E\left(x[k+r]\right) < E\left(x[k]\right) \;\Leftrightarrow\; \|T^r x[k]\|_1 < \|x[k]\|_1 \tag{15}$$

By the definition of induced matrix norm, eq. (15) implies $\|T^r\|_1 < 1$. The following is true for all induced matrix norms:

$$\rho(T) \leqslant \|T^r\|^{1/r} \quad \forall r \in \mathbb{Z}^+ \tag{16}$$

Therefore we conclude that $\rho(T) < 1$. ■

Corollary 1: The flow graph cannot have lossless cycles.

[1]Except the zero state vector where, by definition, the total energy is already zero.

Proof: A more intuitive and verifiable interpretation of rule 4 of the system is that the flow graph cannot have lossless cycles.

A lossless vertex is a vertex whose weights of its outgoing edges sum to 1. All energy held by a lossless vertex is kept in the system during propagation. A lossless tree is a connected series of lossless vertices where all energy held by the vertices in the tree is kept in the system during propagation. A lossless cycle is a lossless tree that closes back on itself with no energy losses. Once energy enters any vertex in the cycle, it will never escape and will stay in the system forever. Most, if not all, real physical systems have no perfect stores of energy, thus have no lossless cycles.

Clearly, for rule 4 to be satisfied, lossless cycles cannot exist. The reverse implication is also true, making rule 4 and the absence of lossless cycles equivalent statements.

Furthermore, if there are no lossless cycles, then by definition there must be a value $r \in \mathbb{Z}^+$ of propagation steps (larger than the length of the longest lossless tree) after which the total energy in the system has decreased, which also proves $\rho(T) < 1$. The reverse implication is also true.

In conclusion: rule 4, $\rho(T) < 1$ and the absence of lossless cycles are all equivalent statements. ■

IV. Equivalent reduction of a flow graph

In this section a process is presented that reduces the number of vertices in the flow graph, and thus reduces the size of the associated transfer matrix T, while keeping the relevant steady-state analysis results unchanged.

On any given flow graph, it is possible that some vertices are not source vertices. Likewise, when calculating the steady-state vertex values, it is possible that some values are not relevant as an analysis result. If a vertex is not a source vertex (so does not contribute to a non-zero value in the b vector) and is also not a vertex relevant for the result (its corresponding steady-state energy value in the x vector is not used after), then it can be eliminated before the $\hat{T}x = b$ system is solved.

We will now explain how a set of vertices can be removed from the flow graph without altering any of the relevant results, and in section VII-B this process will be used specifically in the context of ONoCs to speed up computation time.

Consider the graph $G = (V, E)$ with n vertices. Define R as the set of vertices to remove, Q as the set of intermediate vertices and $S = V \backslash (R \bigcup Q)$ as the rest of the vertices in V. Intermediate vertices are all vertices not in R that are connected directly to one or more vertices in R. Sets R, Q and S are disjoint.

To eliminate the vertices in R from the graph, all edges connected to vertices in R must be removed. These are edges internal to R and edges between R and Q. The problem can be represented in matrix form using the transfer matrix of the graph. We start with the original matrix T, which we partition into 9 sub-matrices:

$$T = \begin{bmatrix} A & B & 0 \\ C & D & F \\ 0 & H & J \end{bmatrix} \tag{17}$$

We assume, without loss of generality[2], that:

- the first set of rows/columns (sub-matrices A, B and A, C) are the in/out-edges respectively for the vertices in R,
- the second set of rows/columns (C, D, F and B, D, H) are the in/out-edges respectively for the vertices in Q, and
- the last set of rows/columns (H, J and F, J) are the in/out-edges respectively for the vertices in S.

This is shown graphically in fig. 2(a). Note that the dimensions of these sub-matrices are defined by the number of vertices in each set R, Q, S, and vertices in R are not connected to vertices in S so the two corresponding sub-matrices are filled with zeros.

By eliminating all edges connected to vertices in R, we get the new transfer matrix U:

$$U = \begin{bmatrix} 0 & 0 & 0 \\ 0 & K & F \\ 0 & H & J \end{bmatrix} \tag{18}$$

A graphical representation of U is shown in fig. 2(b). Sub-matrix K must be determined so that transfer matrices T and U are equivalent for vertices in $Q \bigcup S$.

The equivalence of two matrices for a set of vertices $V^* \subseteq V$ is defined based on eq. (12). Consider $x_i = \hat{T_i}^{-1} b$. Two transfer matrices T_1 and T_2 are equivalent for vertices V^* if any b vector which only contains non-zero values for vertices in V^* results in vectors x_1 and x_2 with the same values for all rows corresponding to vertices in V^*:

$$\begin{aligned}
& T_1 \overset{V*}{\longleftrightarrow} T_2 \\
\Leftrightarrow\ & \delta_i^T \hat{T_1}^{-1} \left(\sum_{j \in V^*} g(j) \delta_j \right) = \delta_i^T \hat{T_2}^{-1} \left(\sum_{j \in V^*} g(j) \delta_j \right) \\
& \qquad \forall g(j) \in \mathbb{R}, i \in V^* \\
\Leftrightarrow\ & \delta_i^T \hat{T_1}^{-1} g(j) \delta_j = \delta_i^T \hat{T_2}^{-1} g(j) \delta_j \\
& \qquad \forall g(j) \in \mathbb{R}, i, j \in V^* \\
\Leftrightarrow\ & \delta_i^T \hat{T_1}^{-1} \delta_j = \delta_i^T \hat{T_2}^{-1} \delta_j \qquad \forall i, j \in V^* \\
\Leftrightarrow\ & \left[\hat{T_1}^{-1} \right]_{ij} = \left[\hat{T_2}^{-1} \right]_{ij} \qquad \forall i, j \in V^*
\end{aligned} \tag{19}$$

To determine K such that matrix U is equivalent to T for vertices in $Q \bigcup S$, we must first calculate $\hat{T}^{-1}$ and $\hat{U}^{-1}$. The inverse of a 2×2 block matrix is

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix}^{-1} = \begin{bmatrix} (A - BD^{-1}C)^{-1} & -(A - BD^{-1}C)^{-1}BD^{-1} \\ -(D - CA^{-1}B)^{-1}CA^{-1} & (D - CA^{-1}B)^{-1} \end{bmatrix} \tag{20}$$

assuming A and D are invertible [10]. We partition matrices T and U into 4 sub-matrices each along the following boundaries

$$T = \left[\begin{array}{c|cc} A & B & 0 \\ \hline C & D & F \\ 0 & H & J \end{array}\right] \qquad U = \left[\begin{array}{c|cc} 0 & 0 & 0 \\ \hline 0 & K & F \\ 0 & H & J \end{array}\right] \tag{21}$$

and apply[3] eq. (20) to $\hat{T}$ and $\hat{U}$, resulting in

$$\hat{T}^{-1} = \begin{bmatrix} [\hat{T}^{-1}]_{11} & [\hat{T}^{-1}]_{12} \\ [\hat{T}^{-1}]_{21} & [\hat{T}^{-1}]_{22} \end{bmatrix} \tag{22}$$

$$\hat{U}^{-1} = \begin{bmatrix} I & 0 \\ 0 & [\hat{U}^{-1}]_{22} \end{bmatrix} \tag{23}$$

where:

$$\begin{aligned}
\left[\hat{T}^{-1} \right]_{22} &= \left(I - \begin{bmatrix} D & F \\ H & J \end{bmatrix} - \begin{bmatrix} C \\ 0 \end{bmatrix} (I - A)^{-1} \begin{bmatrix} B & 0 \end{bmatrix} \right)^{-1} \\
&= \left(I - \begin{bmatrix} D + C(I - A)^{-1}B & F \\ H & J \end{bmatrix} \right)^{-1}
\end{aligned} \tag{24}$$

$$\left[\hat{U}^{-1} \right]_{22} = \left(I - \begin{bmatrix} K & F \\ H & J \end{bmatrix} \right)^{-1} \tag{25}$$

According to eq. (19), if matrices T and U are equivalent for vertices in $Q \bigcup S$, then we must have $[\hat{U}^{-1}]_{22} = [\hat{T}^{-1}]_{22}$. By comparing eq. (24) with eq. (25) we have clearly:

$$K = D + C(I - A)^{-1}B \tag{26}$$

This result is sensible: the flow of energy from Q to R and back to Q is added to the already existing flow between vertices in Q.

In the case where R contains only one vertex, matrix B becomes vector b, matrix C becomes vector c, matrix A becomes scalar a and:

$$[K]_{ij} = [D]_{ij} + \frac{[c]_i [b]_j}{1 - a} \tag{27}$$

This simplifies further to $[K]_{ij} = [D]_{ij} + [c]_i [b]_j$ when the vertex in R does not have a self-looping edge ($a = 0$).

In summary, to eliminate a set R of vertices from the flow graph, start with the associated transfer matrix T, calculate $C(I - A)^{-1}B$ for the corresponding sub-matrices A, B, C in T, add the result to the corresponding sub-matrix D and zero the sub-matrices A, B and C. At this point, the size of the T matrix can be reduced by eliminating from it all the zero rows and columns.

V. ONoC DESIGN, ELEMENTS, INSERTION LOSS AND CROSSTALK

An ONoC uses light instead of electricity to transmit information between system-on-chip components. Each node of the network may send and/or receive information. The sender side of each node emits light signals in multiple wavelengths that must arrive at the receiving side of the correct nodes. Each signal has a specific wavelength and travels between a specific sender/receiver pair. The routing mechanism in the ONoC router is thus fundamental [11].

[2]The order of the vertices in T can be changed at will by re-ordering rows and columns with $T' = PTP^T$, where P is a permutation matrix, and does not influence any result.

[3]The sub-matrices on the diagonal of the block decompositions of $\hat{T}$ and $\hat{U}$ are invertible.

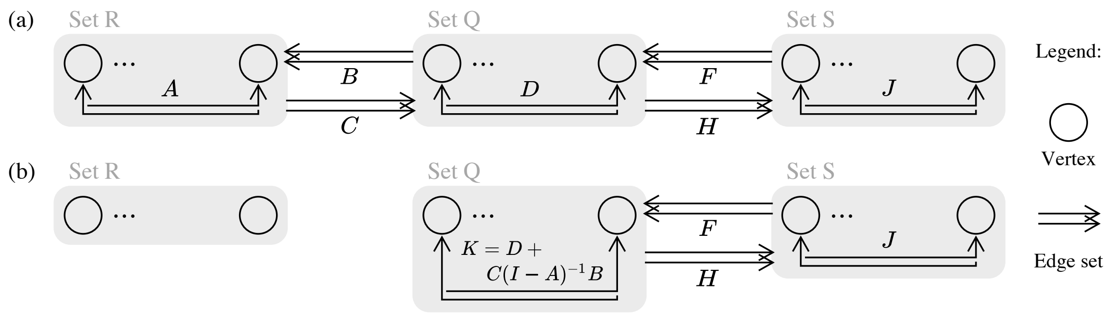

Figure 2. Simplifying a flow graph by removing all edges connected to vertices in the set R. Each set of edges shown is associated with the corresponding sub-matrix of the transfer matrix T of the flow graph. (a) Before simplification. (b) After simplification.

A. ONoC elements

An ONoC is composed of the following relevant optical elements: laser sources, modulator arrays, demodulator arrays, waveguides, Micro-Ring Resonators (MRRs) and waveguide terminators [12].

Laser sources emit light in multiple wavelengths. This light is guided by waveguides to the modulator array of each ONoC node. Each modulator array contains multiple modulators, one for each wavelength, and each modulates light by an electrical data signal. This light then travels through the ONoC router taking the designed path according to its wavelength, eventually reaching the demodulator, where the electrical data signal is retrieved.

MRRs are used to achieve wavelength-based routing [13]. These are silicon ring structures that can be tuned to specific wavelengths. When a signal travels on a waveguide and gets close to an MRR also adjacent to a second waveguide, some energy goes *through* the MRR and continues on the same waveguide, and some energy is *dropped* onto the second waveguide [14]. This effect, shown in fig. 3, works the same way with both parallel and perpendicular waveguides so long as both waveguides are close enough to the MRR. The fractions of energy transmitted to the through and drop ports depend strongly on both the resonance wavelengths of the MRR and the wavelength of the signal. This is the key to wavelength-based routing using MRRs.

An MRR has a set of resonance wavelengths Λ^{res} approximately defined by $\Lambda^{res} \subset \left\{\mu\frac{2\pi R}{i} \mid i \in \mathbb{Z}^+\right\}$ where R is the MRR's radius and μ is a physical parameter. Commonly, the considered band of wavelengths is between $1500\,\mathrm{nm}$ and $1600\,\mathrm{nm}$. The fractions of the input energy transmitted to the through (f_t) and drop (f_d) ports are functions dependent on the wavelength λ of the signal. Their simplified shape is shown in fig. 4 and they are described approximately by

$$f_d(\lambda) = \frac{k_1\left(\frac{\lambda^{res}}{2Q}\right)^2}{(\lambda-\lambda^{res})^2+\left(\frac{\lambda^{res}}{2Q}\right)^2} \tag{28}$$

$$f_t(\lambda) = \frac{(\lambda-\lambda^{res})^2+k_2\left(\frac{\lambda^{res}}{2Q}\right)^2}{(\lambda-\lambda^{res})^2+\left(\frac{\lambda^{res}}{2Q}\right)^2} \tag{29}$$

where λ^{res} is the resonant wavelength in Λ^{res} closest to λ, Q is the MRR's quality factor and k_i are other physical parameters [15]. We have $k_i > 0$ but $k_1 + k_2 < 1$.

Optical waveguides, unlike electrical wires, can cross on the same plane. These crossings are sources of loss and noise but do not alter signal paths: a signal traveling on a waveguide that crosses other waveguides will continue traveling in the same direction on the original waveguide. Lastly, waveguide terminators are placed at the ends of waveguides and their purpose is to absorb as much light as possible to avoid back reflection.

B. Insertion loss

Waveguides and MRRs are real physical elements, therefore signals interacting with them always lose a fraction of their energy. This is called insertion loss. The total insertion loss of the router determines directly the minimum amount of power the laser sources must produce so that signals are detected correctly at the demodulators. Therefore, minimising insertion loss at design time is key to reducing power consumption.

Insertion loss is commonly reported in dB (logarithmic scale), but this work requires that it be used as a multiplicative factor. The relationships in both formats between input energy, output energy and insertion loss are

$$\text{output}_{dB} = \text{input}_{dB} - il_{dB} \tag{30}$$

$$\text{output}_f = \text{input}_f \times il_f \tag{31}$$

and the corresponding conversion formula is $il_{dB} = -10\log_{10}(il_f)$. Naturally, here we have $il_{dB} > 0$ and $il_f < 1$.

There are five types of insertion loss [5]. Waveguides cause *propagation loss* (L_p) and *bend loss* (L_b). Propagation loss in dB is proportional to the length of the waveguide and bend loss in dB depends on the radius of the bend. Waveguide crossings cause a fixed amount of *crossing loss* in dB (L_c) per crossing.

Lastly, MRRs cause two types of loss: *drop loss* (L_d) and *through loss* (L_t). These are directly related to eq. (28) and eq. (29). If a signal with wavelength λ arrives at an MRR where λ is a resonance wavelength, the signal is routed to the drop port but loses some energy in the process. Thus, we have drop loss $il_f = f_d(\lambda)$. Conversely, if a signal with wavelength λ arrives at an MRR where λ is **not** a resonance wavelength, the signal is routed to the through port but loses some energy in the process. Thus, we have through loss

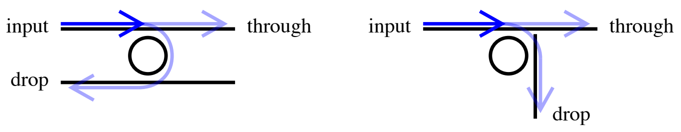

Figure 3. Routing mechanism of an MRR. A fraction of the energy that arrives at the input port continues on the same waveguide and is routed to the through port; another fraction is moved to a second waveguide and routed to the drop port. This mechanism works with both parallel and perpendicular waveguides as long as both waveguides are close enough to the MRR.

$il_f = f_t(\lambda)$. Some of the signal energy which is not routed to the correct port becomes noise, as explained in section V-C. These through/drop loss factors are very accurate but depend on both the wavelength of the signal and the radius of the MRR. Therefore, many works simplify and approximate them by averaging the through/drop loss factors over all signal wavelengths and MRR radii used [5]–[8]. Our method can use both the average and the accurate versions.

C. Crosstalk

Crosstalk broadly refers to the influence of each signal on other signals [16]. Consider a signal traveling between two ONoC nodes. A fraction of its energy arrives at the assigned demodulator, a fraction arrives at other demodulators, causing crosstalk, and the rest is lost due to insertion loss. Any signal energy that strays from its intended path becomes noise because when it inevitably arrives at incorrect demodulators it interferes with the detection of the signals assigned to them.

There are two types of crosstalk: inter-channel and intra-channel crosstalk [5]. Inter-channel crosstalk happens when the wavelength of the noise is distant enough from the wavelength of the signal. Intra-channel crosstalk happens when the wavelengths of the signal and noise are the same or close enough. The distinction is whether the noise can be effectively filtered out from the signal before detection. Naturally, this makes intra-channel crosstalk worse than inter-channel crosstalk.

There are five sources of noise, shown in fig. 5 [5]. A signal that arrives at a waveguide terminator is never fully absorbed. *Terminator back reflection* (C_{tbr}) is the fraction of noise energy reflected back onto the waveguide in the reverse direction. A signal that arrives at a crossing mostly continues in the same direction on the same waveguide, but a fraction of the energy escapes through the sides of the crossing as noise and another fraction is reflected back as noise. These are, respectively, *crossing side spill* (C_{css}) and *crossing back reflection* (C_{cbr}).

Lastly, MRRs are the sources of two types of noise. If a signal with wavelength λ arrives at an MRR where λ is a resonance wavelength, the signal is routed to the drop port but some energy still escapes to the through port as noise. This is *MRR on-resonance through* (C_{r1t}) noise with $C_{r1t} = f_t(\lambda)$. Conversely, if a signal with wavelength λ arrives at an MRR where λ is **not** a resonance wavelength, the signal is routed to the through port but some energy still escapes to the drop port as noise. This is *MRR off-resonance drop* (C_{r0d}) noise with $C_{r0d} = f_d(\lambda)$. Like with the insertion loss factors above, both

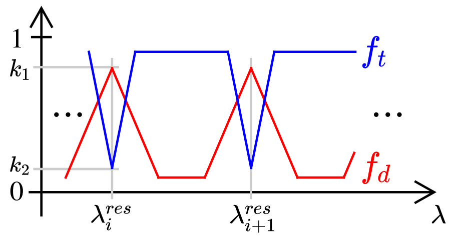

Figure 4. Simplified shape of the fractions of the input energy transmitted to the through (f_t) and drop (f_d) ports of an MRR depending on the signal wavelength λ. The resonant wavelengths of the MRR are $\lambda_i^{res} \in \Lambda^{res}$ and k_i are physical parameters. When the wavelength of light λ is far away from all λ_i^{res}, most of the input energy is transmitted to the through port. When λ is close to a λ_i^{res}, most of the input energy is transmitted to the drop port. When λ is equal to a λ_i^{res} (i.e. $\lambda \in \Lambda^{res}$), the fractions of input energy transmitted to the drop and through ports are k_1 and k_2 respectively.

average and accurate versions of these factors are available and both can be used in our method.

For each signal received at a demodulator, the noise also arriving at the demodulator and interfering with the detection of the signal can be separated into three types: intra-channel noise caused by the signal itself taking a different path to arrive at its demodulator, intra-channel noise caused by other signals of the same wavelength and inter-channel noise caused by other signals of different wavelengths. SNR is defined as the ratio between signal and noise power at the demodulator. Our method can calculate the SNR for each type of noise separately.

VI. Steady-state crosstalk calculation for ONoCs

We now explain how to calculate SNR for any ONoC design using the method in section III. This is possible because ONoCs satisfy all four rules: waveguides hold specific amounts of energy in transit, modulators/demodulators are sources/sinks of energy, insertion loss and crosstalk factors transfer fixed fractions of energy between waveguide locations, and all ONoC elements cause insertion loss so ONoCs cannot have lossless cycles.

Assume a modulator sends a signal into the router. As it travels through the router, two types of interactions are possible: the signal may lose a fraction of its energy due to insertion loss or the signal may bifurcate due to noise-inducing elements. Therefore, we identify two types of light streams traveling through the router and two types of transformations that can be applied to them. A light stream can either be signal or noise, and both stream types can suffer insertion loss and bifurcation. Signal/noise streams contribute to signal/noise energy levels respectively at the demodulators they arrive at.

When a signal goes through a noise-inducing element, it bifurcates and a fraction of its energy becomes first-order noise. This noise stream travels through the router suffering insertion loss like any other light stream. If it encounters a noise-inducing element, it bifurcates again and a fraction of the first-order noise energy becomes second-order noise (no transformation can be applied to a noise stream that generates a signal stream back). This process may happen up to an infinite number of times depending on the design of the

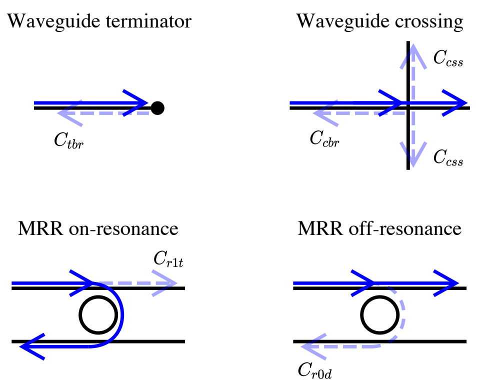

Figure 5. Noise sources in ONoCs: terminator back reflection (C_{tbr}), crossing side spill (C_{css}), crossing back reflection (C_{cbr}), MRR on-resonance through (C_{r1t}) and MRR off-resonance drop (C_{r0d}). Full arrows are signal paths, dashed arrows are noise paths.

router. The energy of all $\{1, 2, ...\infty\}$-order noise streams is summed together to contribute to the noise energy levels for the demodulators the streams arrive at.

To model the propagation of light through an ONoC and calculate all SNR results, first we translate the ONoC into a corresponding set of flow graphs, then we convert the graphs into transfer matrices and then we apply the mathematical method in section III. Since streams go through different paths and suffer different transformations depending on their wavelength, a flow graph G_λ must be created for each wavelength λ used in the ONoC. This results in a different transfer matrix T_λ per wavelength.

A. Building a flow graph for an ONoC

A three-step process is used to convert an ONoC design into a flow graph G_λ for a specific wavelength λ: first the locations of compound vertices on the router are identified, then each compound vertex is converted into multiple single vertices and then directed edges between single vertices are added with the appropriate edge weights.

Compound vertices are placed on waveguides such that, for all locations where a light stream can suffer insertion loss and/or bifurcation, there are two adjacent compound vertices, one just before the transformation and the other just after. Figure 6(a) shows the locations of compound vertices for ONoC elements. A compound vertex is placed on each end of a waveguide, just before a waveguide terminator, on the four sides of a waveguide crossing and on the four locations around an MRR. A compound vertex is also placed just after each modulator and just before each demodulator.

Each compound vertex tracks the amount of signal and noise energy going through the waveguide on its location. Since there are two types of streams and streams can travel in two directions on a waveguide, each compound vertex is decomposed into four single vertices: signal direction top/left, signal direction bottom/right, noise direction top/left, noise direction bottom/right. Figure 6(b) shows examples of this decomposition. Flow graphs are composed of these single vertices.

To build a flow graph of an ONoC comprised of multiple individual ONoC elements, the compound vertices (and, by extension, the single vertices) of each element are merged with the ones from its adjacent elements. An example is shown in fig. 7.

Having listed all the vertices of a flow graph, the final step is to add edges with the correct weights. Edges describe what fractions of energy are transferred between vertices, and thus how light is propagated through the router. Since a light stream can suffer insertion loss and/or bifurcation between vertices, the edge weights are of type loss and crosstalk respectively. Figure 6(b) shows the edges required for each ONoC element. Edge direction follows intuitively from the light direction associated with each vertex.

For each MRR, the choice to use the on- or off-resonance case is based on the path the signal should take according to the design of the network. The L_d, L_t, C_{r0d} and C_{r1t} weights can also be the average or the accurate versions. The average weights are the same for all G_λ graphs but the accurate weights also depend on the wavelength λ of the flow graph G_λ and the resonance wavelengths of the MRR.

B. Transfer matrix types

We identify four edges types depending on the types of the vertices they are connected between and the type of their weight:

- *Type SL* from a signal (S) vertex to a signal vertex with a loss (L) weight.
- *Type SX* from a signal (S) vertex to a noise vertex with a crosstalk (X) weight.
- *Type NL* from a noise (N) vertex to a noise vertex with a loss (L) weight.
- *Type NX* from a noise (N) vertex to a noise vertex with a crosstalk (X) weight.

The distinction between these four edge types is crucial to the method and can be explained in three stages:

1) If we only consider SL edges, then only signals are propagated through the graph. Signal streams suffer insertion loss before arriving at their respective demodulators, but no noise streams are generated and thus demodulators will receive no noise energy.
2) If we consider SL, SX and NL edges then only signal streams and first-order noise streams are propagated through the graph. SL edges are responsible for the propagation of signal streams, SX edges are responsible for the generation of first-order noise streams from signal streams and NL edges are responsible for the propagation of first-order noise streams. Demodulators receive both signal and first-order noise energy. Therefore, first-order SNR can be calculated.
3) If NX edges are also considered along with SL, SX and NL edges, then n-order noise streams generate $n+1$-order noise streams. Noise received by the demodulators will be the sum of all $\{1, 2, ...\infty\}$-order noise and the SNR values calculated will be all-order SNR.

These three stages result in three possible T_λ transfer matrix types for each G_λ flow graph: T_λ^S containing only SL edges,

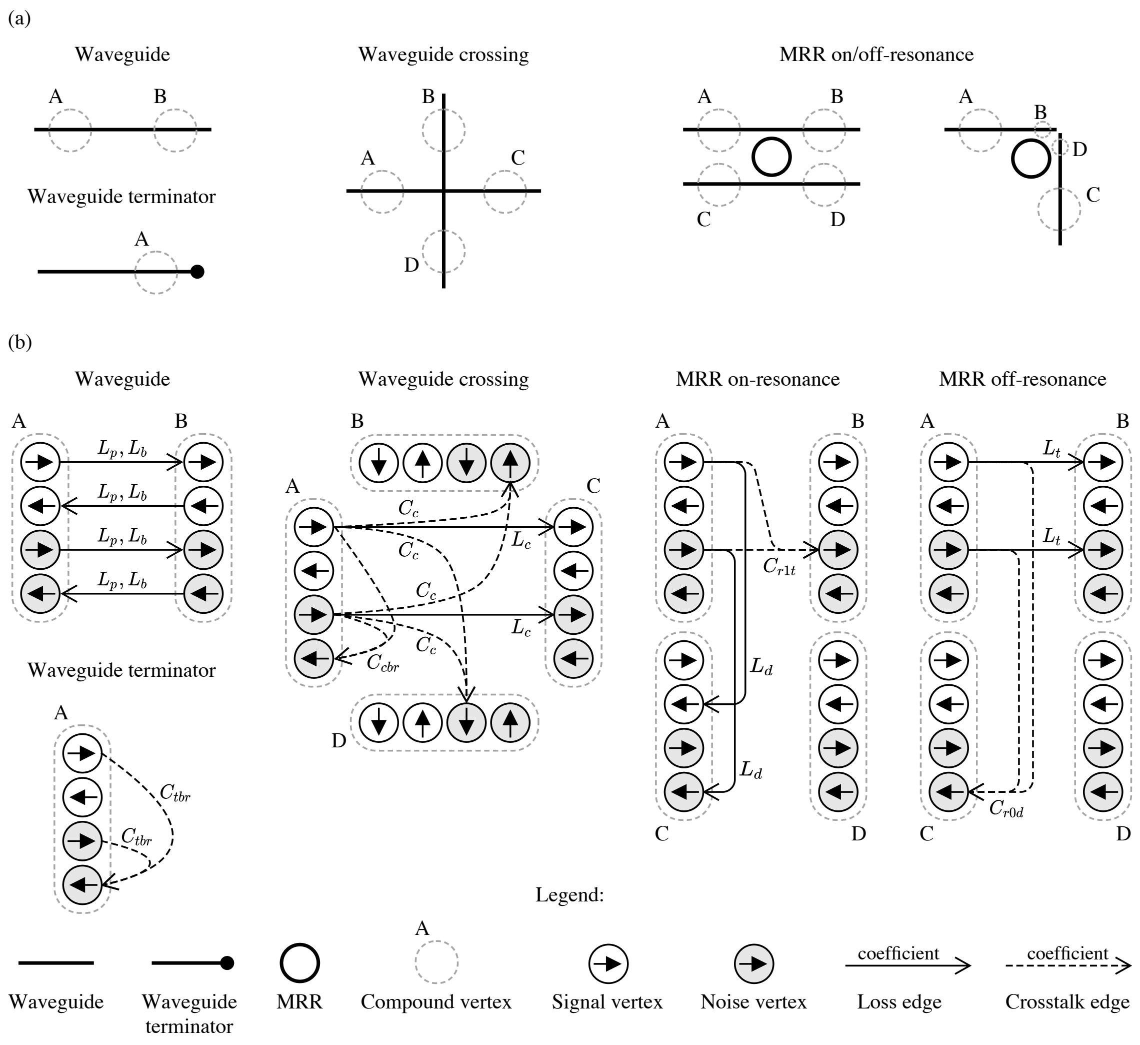

Figure 6. (a) Location of compound vertices for ONoC elements. For the MRR on/off-resonance case, the two possible waveguide placement configurations around an MRR are shown. Both configurations have 4 compound vertices A,B,C,D. Although they are placed in different locations relative to the MRR, they work exactly in the same way when converting to a flow graph. (b) The decomposition of the compound vertices into single (signal and noise) vertices and the edges connecting them. The loss/crosstalk weight used in each edge is also shown. In the waveguide case, the weights are propagation loss (L_p) and/or bending loss (L_b) depending on the path of the waveguide. For simplicity and clarity, in the waveguide crossing and MRR cases only the edges leaving vertices in the compound vertex A are shown. The edges leaving compound vertices B, C and D are analogous and thus not drawn.

$T_\lambda^{SN_1}$ containing SL, SX and NL edges, and $T_\lambda^{SN_\infty}$ containing all edges. All these matrices have the same size because the number of vertices in the graph doesn't change[4] but the fewer edge types considered, the sparser the matrix is.

Depending on the choice of T_λ matrix type and on how the b vector is constructed, different kinds of information can be obtained from solving $\hat{T}_\lambda x = b$. Matrix T_λ^S can only be used to calculate insertion loss for signals whereas matrices $T_\lambda^{SN_*}$ (where $*$ is 1 or ∞) can also be used to calculate noise energy levels and SNR values at the demodulators.

C. Calculating signal insertion loss

Assume a signal is sent on wavelength λ. Let i be the vertex where the signal's modulator sends signal energy into the router and j be the vertex where the signal's demodulator receives signal energy from the router.

[4]There is an exception to this statement that will be explained in section VII-B.

To calculate this signal's total insertion loss, first solve $\hat{T}_\lambda^* x = b$ for vector x where $*$ is any of the three matrix types and $b = 1\delta_i$. Then, the insertion loss on this signal's path through the router from vertex i to vertex j ($il_{i\to j}$) is the logarithm of the ratio of signal energy sent into the router ($[x]_i$) to signal energy received from the router ($[x]_j$):

$$\begin{aligned} il_{i\to j} &= 10\log_{10}\left(\frac{[x]_i}{[x]_j}\right) = 10\log_{10}\left(\frac{1}{[x]_j}\right) \\ &= -10\log_{10}\left([x]_j\right) \end{aligned} \tag{32}$$

This reasoning can be extended to multiple signals at once as long as their wavelengths are the same and their paths do not interfere, i.e. do not contain the same waveguides. Just use eq. (6) with $g(v) = 1\ \forall v \in V^*$ where V^* is the set of vertices of the corresponding modulators sending signal energy into the router. Then solve $\hat{T}_\lambda^* x = b$ once and apply eq. (32) for each signal. This reduces the computational burden of calculating the insertion loss of all signals.

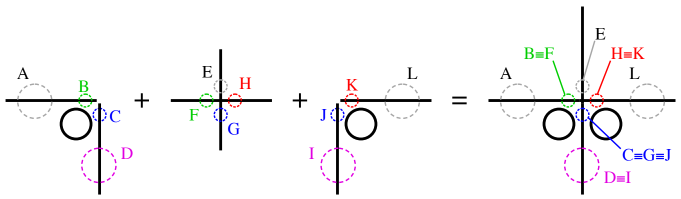

Figure 7. Example of how the compound vertices of isolated elements shown in fig. 6(a) are merged to form a flow graph of an ONoC. In this case, the four compound vertices of two MRRs each and the four compound vertices of a waveguide crossing are combined into a total of seven compound vertices.

D. *Calculating noise at demodulators*

Assume a signal is sent on wavelength λ. Let i be the vertex where the signal's modulator sends signal energy into the router, j be the vertex where the signal's demodulator receives signal energy from the router and v_k be all the vertices where the demodulators receive noise energy from the router.

To calculate the noise caused by this signal on all demodulators, solve $\hat{T}^{SN_*}_{\lambda} x = b$ where $*$ is the chosen noise type and $b = g\delta_i$ where g is the signal's energy at the modulator. The signal's energy at the demodulator is given by $[x]_j$ and the noise energies at the demodulators are given by $[x]_{v_k}$.

By solving one system per signal, the noise caused by each signal on each demodulator can be calculated separately. Intra-channel noise caused by the signal itself, intra-channel noise caused by other signals and inter-channel noise can all be calculated separately in this way. However, like in section VI-C, the computational burden can be reduced by combining signals with the same wavelength and non-interfering paths into the same b vector. In this case, values $[x]_{v_k}$ contain the total amount of noise generated by all the selected signals and received by each demodulator.

E. *Calculating SNR at demodulators*

To obtain all SNR values, first calculate the insertion loss of all signals using section VI-C and from the insertion loss determine the required energy of each signal at the modulator[5]. Then, calculate the signal and noise energies at the demodulators using section VI-D and from there derive the SNR values. To reduce the computational burden, multiple signals with the same wavelength and non-interfering paths can be combined into the same b vector in which case only the sum of the generated noise is obtained instead of individual noise energy levels.

VII. Practical considerations

There are multiple practical aspects to take into consideration when solving $\hat{T}x = b$ that can make this method faster and more memory efficient. These will be discussed now.

A. *Sparse matrix methods*

The average transfer matrix density (ratio of non-zero entries to total entries in the matrix) is very low and approaches

[5]In case the energy levels of the signals are already available, this step can be skipped.

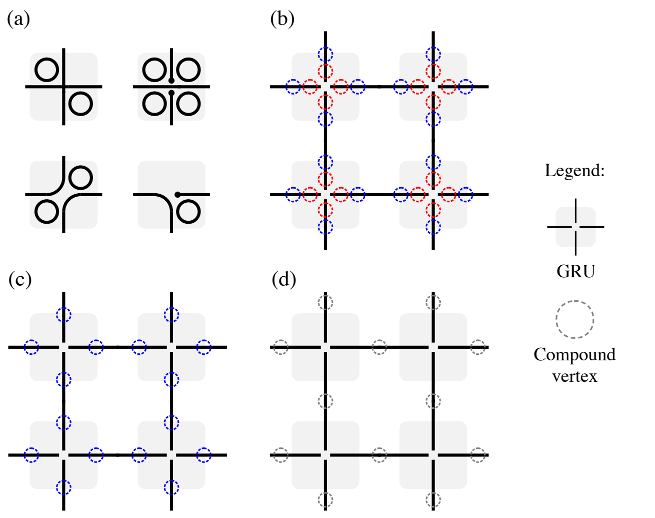

Figure 8. (a) Example GRU configurations. (b) Location of the eight compound vertices on a GRU: four internal vertices (in red) and four port vertices (in blue). (c) Removing internal vertices. (d) Merging port vertices. Final result has up to 4× lower vertex count for GRU-based designs.

zero as the number of vertices in the flow graph increases. The use of sparse matrices and sparse matrix methods is thus essential to storing T efficiently and solving $\hat{T}x = b$ quickly.

Methods for solving linear systems can be divided into two types: direct and iterative. Direct methods guarantee a full precision solution in a finite number of steps. Conversely, iterative methods start from an initial guess and converge (some methods, asymptotically) to the solution with each iteration. Because of their iterative nature, iterative methods can trade accuracy for speed by stopping earlier when a good enough solution is achieved.

Direct methods for sparse linear systems include sparse versions of Cholesky, LU and QR factorisations. To avoid fill-in during factorisation, multiple strategies exist. One such strategy is ordering vertices in the matrix to reduce its bandwidth, which can be applied here very easily. We have found that LU factorisation with Cuthill-McKee vertex ordering works well.

Iterative methods for sparse linear systems include Conjugated Gradient (CG) methods, the Generalized Minimal Residual method (GMRES) and the Biconjugate Gradient Stabilized method (BiCGSTAB). Iterative methods may also benefit greatly from using a preconditioning matrix. We have found that BiCGSTAB with an incomplete LU preconditioner works very well.

B. *Transfer matrix reduction techniques*

Any process by which a transfer matrix's size is reduced or its sparsity is increased without affecting the result can have the potential to save memory and speed up the matrix method.

A vertex whose corresponding row and column of the transfer matrix are empty, i.e. filled with zeros, is a vertex without any ingoing or outgoing edges. These vertices can be eliminated from the flow graph and transfer matrix without consequence. Because of this, since noise vertices in T^S are not used, T^S can actually be half the width and height of the

T^{SN_*} matrices. Likewise, when the process from section IV is used, the resulting empty rows/columns in sub-matrices A, B, C can be removed. However, good sparse matrix method implementations should be able to handle empty rows/columns with little to no overhead, dismissing the requirement to remove them from the matrix once they're emptied.

Many ONoC routers are built from multiple copies of one basic, configurable, routing structure. These structures are self-contained collections of ONoC elements that perform simple routing tasks between a fixed number of ports (commonly four or five). They are copied multiple times and interconnected through a specific topology to form an ONoC router for as many network nodes as required. Each structure is configured internally so that all structures work together to accomplish the routing requirements of the full ONoC router. For example, active ONoCs can be built from the Crux [17], Cygnus [18] or OTAR [19] structures interconnected through mesh, fat-tree or folded torus topologies [5]. Passive ONoCs (wavelength-routed ONoCs) can be built using Photonic Switching Elements (PSEs) [20] or a grid of Generic Routing Units (GRUs) [21]. Crucially, for each of these routing structures, there is a relatively small finite amount of possible internal configurations. Because of this, it is possible to list these configurations and use the process from section IV to reduce the transfer matrices of each configuration substantially. This reduction process only has to be done once because all the simplified transfer matrices can be saved in a library. Then, when calculating results for an ONoC design (which uses a specific combination of routing structure configurations) its transfer matrix can be built directly by combining the simplified transfer matrices from the library. When solving multiple $\hat{T}x = b$ systems for different ONoC designs based on the same structure library, the repeated effort caused by indirectly solving multiple copies of the same structure configurations is eliminated.

Here we use a passive ONoC built using GRUs as an example. A GRU is a configurable waveguide crossing with four external ports. It can have a waveguide crossing, up to four MRRs and/or up to two waveguide bends. GRUs are interconnected in a grid topology.

GRUs have a total of 73 possible internal configurations[6] of crossing, MRR and bend placements. Some examples are shown in fig. 8(a). According to section VI-A, a GRU requires eight compound vertices to fully describe its routing behaviour: four internal vertices and four port vertices. Thus, a naïve flow graph for a grid of GRUs requires 8 compound vertices per GRU. This is shown in fig. 8(b).

Since the routing structures are self-contained, reusable and finite in number, a library of their simplified versions can be built *a priori*. The process from section IV is used to remove all internal vertices of each structure. For a GRU-based design, this results in a flow graph with only 4 compound vertices per GRU, i.e. a $2\times$ reduction in vertex count. This is shown in fig. 8(c).

A further optimisation is possible when connecting the structures together: instead of having a compound vertex per port for each structure, have a compound vertex per pair of connected ports. The insertion loss of the waveguide section connecting the two ports is shared between both structures. In the 2×2 GRU grid example of fig. 8, merging port vertices brings the total number of compound vertices down to 12, as shown in fig. 8(d). This is a reduction of $\frac{16}{12} = 1.33\times$ compared to fig. 8(c), for a total reduction of $2 * 1.33 = 2.66\times$ compared to fig. 8(b). Note that compound vertices on the boundaries of the GRU grid cannot be merged. As the GRU grid size n increases, the number of vertices that cannot be merged increases with $O(n)$, but the number of vertices that can be merged increases with $O(n^2)$. Thus, this provides an asymptotical vertex count reduction of $2\times$, for a total reduction that approaches $2 * 2 = 4\times$.

More complex structures lead to larger matrix reductions. For example, a Crux router [5] used in an active ONoC has only 780 possible configurations regardless of wavelength count. By analysis of the Crux router's design together with the vertex placement rules described in section VI-A, we can determine that each structure requires $24n + 34$ internal compound vertices and 10 port compound vertices, where n is the number of wavelengths. Since we can remove all internal vertices and merge most port vertices, this process leads to a reduction in vertex count of up to $\frac{24n+34+10}{10} \times 2$, which is at least $13.6\times$ for $n = 1$ and already $56.8\times$ for $n = 10$.

TABLE I
INSERTION LOSS AND CROSSTALK TECHNOLOGY PARAMETER VALUES FROM [5].

Insertion loss	
Propagation loss	$0.274\,\mathrm{dB/cm}$
Bend loss	$0.005\,\mathrm{dB}/90°$
Crossing loss	$0.05\,\mathrm{dB}$
Drop loss	$1\,\mathrm{dB}$
Through loss	$0.005\,\mathrm{dB}$
Crosstalk	
Terminator back reflection	$50\,\mathrm{dB}$
Crossing side spill	$40\,\mathrm{dB}$
Crossing back reflection	—
MRR on-resonance through	$25\,\mathrm{dB}$
MRR off-resonance drop	$20\,\mathrm{dB}$

VIII. EXPERIMENTAL RESULTS

We implemented this method in C++ and made use of Eigen [22], a C++ linear algebra library with sparse matrix solvers. To exemplify the use of this method, we calculated the SNR results for the 16 node passive ONoC design in [12]. This design uses 17 wavelengths to provide connectivity between 16 network nodes with a total of $16 \times 15 = 240$ signals[7]. The design of this ONoC allows all signals to travel without contention, meaning that all 240 sender/receiver connections can be in use at the same time. Naturally, this causes the highest amount of crosstalk in the router and is the case considered in the following results analysis. For this analysis we used the technology parameters from [5] shown in table I.

[6]Including rotations and reflections.

[7]The network does not have communication paths from a node to itself.

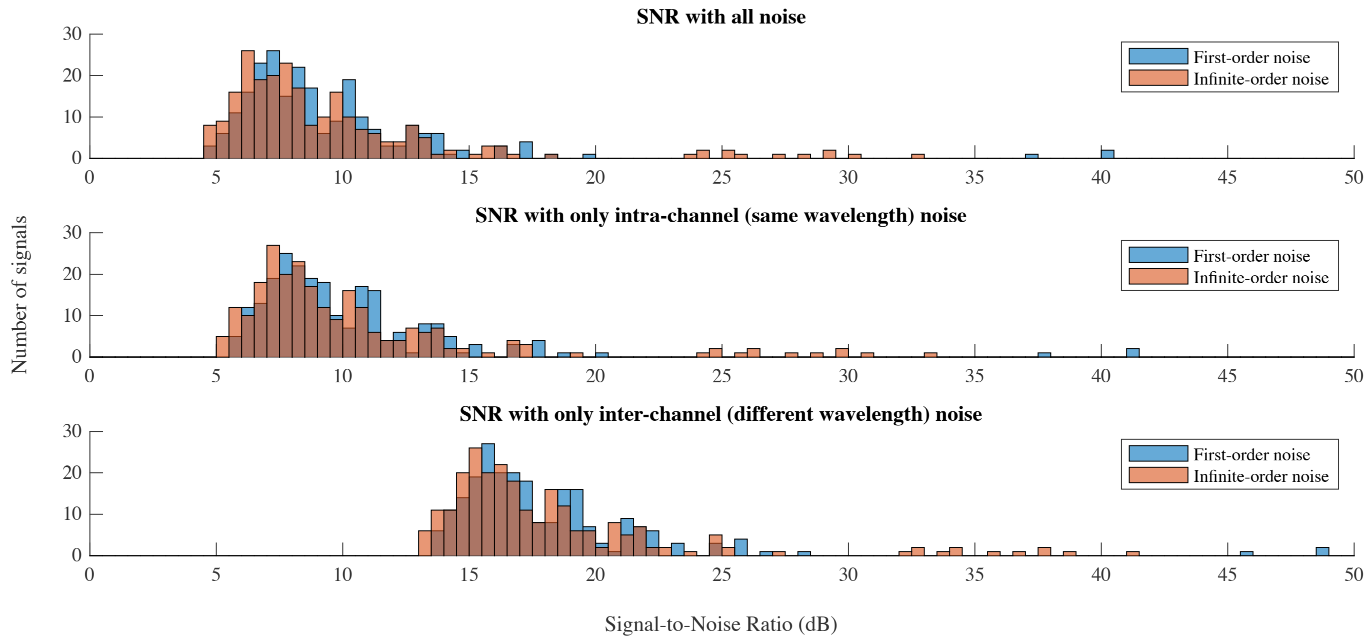

Figure 9. Histogram of SNR results for all 240 signals of a 17 wavelength passive ONoC router.

We followed the process in section VI-E. First we solved 17 linear systems (one per wavelength) to calculate the insertion loss of all 240 signals. Each signal is sent into the network with the minimum amount of energy required to overcome its insertion loss considering a demodulator sensitivity of -20 dBm [23]. We then solved 17 linear systems again to calculate the signal and noise energy levels in the network. To compare between first-order and infinite-order SNR we solved the second batch of 17 linear systems twice: once with $T_\lambda^{SN_1}$ matrices and once with $T_\lambda^{SN_\infty}$ matrices. We used BiCGSTAB with an incomplete LU preconditioner to solve all linear systems.

From the resulting steady-state vectors we can obtain the noise energy level that each signal's demodulator receives on each wavelength. The noise received on the signal's wavelength is intra-channel noise and the sum of the noise received on other wavelengths is inter-channel noise. A histogram of the SNR results for all signals is shown in fig. 9.

It is clear that in this design SNR is limited mostly by the intra-channel noise. While inter-channel noise energy levels are higher (there are more signals on more wavelengths compared to intra-channel noise), this noise is filtered just before arriving at the demodulator. The average (over all signals) infinite-order SNR with just inter-channel noise is $18.08\,\mathrm{dB}$, with just intra-channel noise is $10.12\,\mathrm{dB}$ and with all noise is $9.47\,\mathrm{dB}$. The effect of inter-channel noise becomes more pronounced as the number of used wavelengths increases.

On average, infinite-order SNR with all noise is only $0.65\,\mathrm{dB}$ lower than first-order SNR with all noise. However, nine signals do not receive any first-order noise, making their first-order SNR infinite, while in fact the average infinite-order SNR with all noise for these signals is $27.06\,\mathrm{dB}$. Furthermore, even for signals whose first-order SNR is not infinite, the maximum difference between first-order SNR and infinite-order SNR (with all noise) is as high as $40.38 - 24.23 = 16.15\,\mathrm{dB}$. We conclude that, while for most signals the infinite order and first order results are close, some outliers can only be correctly characterised using infinite-order noise calculations. Infinite-order calculations are thus necessary for a full and accurate analysis of an ONoC.

Noise-enabled ($T_\lambda^{SN_*}$) transfer matrices obtained directly from the process in section VI-E have a size of 5408×5408. Each first-order noise and infinite-order noise linear system is solved on average in $658\,\mu\mathrm{s}$ and $2232\,\mu\mathrm{s}$ respectively[8]. Using the matrix reduction techniques from section VII-B, the matrix size becomes 1376×1376 and each system is solved on average in $123\,\mu\mathrm{s}$ and $332\,\mu\mathrm{s}$ respectively. This is a $3.93\times$ reduction in matrix size (approximately $4\times$ as expected from GRU based designs) and a $5.35\times$ and $6.72\times$ increase in speed respectively.

A. *Steady-state flow map*

The steady-state result vector x can be used for more than just calculate SNR results. In general, it contains the steady-state energy levels for all vertices in the flow graph. All this information can be superimposed on the flow graph itself to create useful visualisations. For example, in the case of ONoCs, it can convey graphically the steady-state optical energy distribution on a per signal basis, per wavelength basis, or in total. An example is shown in fig. 10. Furthermore, by using the flow propagation relationships in eqs. (7) and (8), flow animations can also be produced frame by frame. This provides ONoC designers with powerful analysis capabilities in a very hands-on approach, such as discerning whether the ONoC contains any optical energy hotspots, understanding

[8]Using a 2.3 GHz 8-Core Intel Core i9 on a single thread.

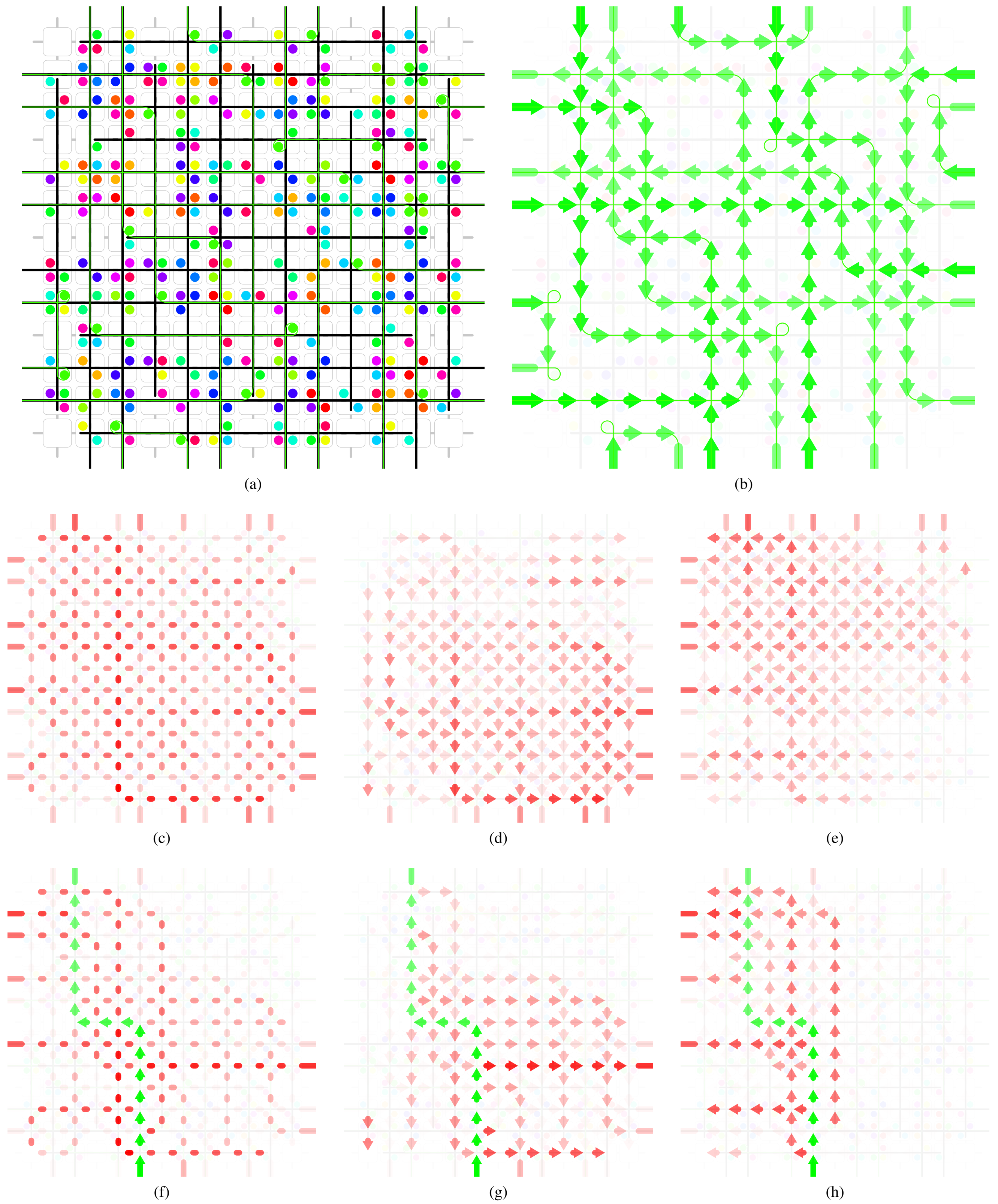

Figure 10. Breakdown of noise generation and propagation in a router using a steady-state flow map. (a) Complete router design. The modulators and demodulators of the 16 nodes are connected to the sides of the router. Circles are MRRs and color indicates their resonance wavelength. (b) Signal paths of one specific wavelength. (c) Distribution of infinite-order noise on the router waveguides caused by the signals in (b). Stronger red indicates a higher noise level. (d-e) Noise levels in (c) discriminated by direction. (f-h) Same as (c-e) but for one signal only.

which ONoC elements are most responsible for noise generation, exploring how noise flows through the ONoC and from there deriving the best locations for design modifications (for example, adding extra MRRs and waveguide terminators to filter noise on certain wavelengths).

IX. Conclusion

In this work we described a new general method to analyse the steady-state of a system. This method can be applied to any system obeying a few simple rules, namely not having any perfect stores of energy. The steady-state is obtained by solving $\hat{T}x = b$ with a transfer matrix obtained from the system's flow graph. We also derived a process through which the flow graph and transfer matrix can be reduced in complexity without affecting the steady-state results.

We applied this method to the calculation of first-order and infinite-order noise and SNR of any ONoC design. Our method is quick and efficient through the use of sparse matrices and methods for sparse linear systems and accurate by supporting the wavelength-dependent versions of the insertion loss and crosstalk factors. It also allows for powerful analysis and visualisation capabilities.

In the future, we intend to research potential design optimisation capabilities that a method based on linear algebra affords. Most importantly, using iterative methods for linear systems to trade off accuracy for speed inside an optimisation loop, providing iterative methods with starting solutions from previous optimisation loops to further speed up computation, and performing sensitivity analysis on the linear systems to better direct the iterative improvements made by the optimisation processes.

References

[1] E. Fusella *et al.*, "Path setup for hybrid noc architectures exploiting flooding and standby," *IEEE Transactions on Parallel and Distributed Systems*, pp. 1403–1416, 2017.

[2] X. Wang *et al.*, "A highly scalable optical network-on-chip with small network diameter and deadlock freedom," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, pp. 3424–3436, 2016.

[3] M. Ortín-Obón *et al.*, "A complete electronic network interface architecture for global contention-free communication over emerging optical networks-on-chip," in *Proceedings of the 24th Edition of the Great Lakes Symposium on VLSI*, May 2014, pp. 267–272.

[4] C. Chen *et al.*, "Sharing and placement of on-chip laser sources in silicon-photonic nocs," 09 2014.

[5] M. Nikdast *et al.*, "Crosstalk noise in wdm-based optical networks-on-chip: A formal study and comparison," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, pp. 2552–2565, Nov 2015.

[6] Y. Xie *et al.*, "Formal worst-case analysis of crosstalk noise in mesh-based optical networks-on-chip," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, pp. 1823–1836, 2013.

[7] M. Nikdast *et al.*, "Fat-tree-based optical interconnection networks under crosstalk noise constraint," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, pp. 156–169, 2015.

[8] Y. W. Kim *et al.*, "Extended worst-case osnr searching algorithm for optical network-on-chip using a semi-greedy heuristic with adaptive scan range," *IEEE Access*, pp. 125 863–125 873, 2020.

[9] M. Xiao *et al.*, "Crosstalk-aware automatic topology customization and optimization for wavelength-routed optical nocs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2022.

[10] T.-T. Lu *et al.*, "Inverses of 2 × 2 block matrices," *Computers & Mathematics with Applications*, pp. 119–129, 2002.

[11] M. Ortín-Obón *et al.*, "Contrasting laser power requirements of wavelength-routed optical noc topologies subject to the floorplanning, placement, and routing constraints of a 3-d-stacked system," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, pp. 2081–2094, July 2017.

[12] A. Truppel *et al.*, "Psion 2: Optimizing physical layout of wavelength-routed onocs for laser power reduction," in *Proceedings of the 39th International Conference on Computer-Aided Design*, 2020.

[13] A. Peano *et al.*, "Design technology for fault-free and maximally-parallel wavelength-routed optical networks-on-chip," in *2016 IEEE/ACM International Conference on Computer-Aided Design*, Nov 2016, pp. 1–8.

[14] B. Little *et al.*, "Microring resonator channel dropping filters," *Journal of Lightwave Technology*, pp. 998–1005, 1997.

[15] S. Xiao *et al.*, "Modeling and measurement of losses in silicon-on-insulator resonators and bends," *Opt. Express*, pp. 10 553–10 561, Aug 2007.

[16] R. Ramaswami *et al.*, *Optical Networks A Practical Perspective 3ed.* Elsevier, 2010.

[17] Y. Xie *et al.*, "Crosstalk noise and bit error rate analysis for optical network-on-chip," in *Proceedings of the 47th Design Automation Conference*, Jan 2010, pp. 657–660.

[18] H. Gu *et al.*, "A low-power low-cost optical router for optical networks-on-chip in multiprocessor systems-on-chip," in *2009 IEEE Computer Society Annual Symposium on VLSI*, May 2009, pp. 19–24.

[19] ——, "A low-power fat tree-based optical network-on-chip for multi-processor system-on-chip," in *2009 Design, Automation Test in Europe Conference Exhibition*, 2009, pp. 3–8.

[20] A. von Beuningen *et al.*, "Proton+: A placement and routing tool for 3d optical networks-on-chip with a single optical layer," *J. Emerg. Technol. Comput. Syst.*, pp. 44:1–44:28, Dec 2015.

[21] A. Truppel *et al.*, "Psion+: Combining logical topology and physical layout optimization for wavelength-routed onocs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pp. 5197–5210, 2020.

[22] Eigen Development Team, "Eigen," 2022, version 3.4.0. [Online]. Available: https://eigen.tuxfamily.org

[23] M. Ortín-Obón *et al.*, "A tool for synthesizing power-efficient and custom-tailored wavelength-routed optical rings," in *Asia and South Pacific Design Automation Conference*, Jan 2017, pp. 300–305.