

Moving forward with multilayer metastructures - simple design of complex optics [Invited]: supplement

DAVID A. B. MILLER* 

Ginzton Laboratory, Stanford University, 348 Via Pueblo Mall, Stanford, CA 94305, USA

**dabm@stanford.edu*

This supplement published with Optica Publishing Group on 22 September 2026 by The Authors under the terms of the [Creative Commons Attribution 4.0 License](#) in the format provided by the authors and unedited. Further distribution of this work must maintain attribution to the author(s) and the published article's title, journal citation, and DOI.

Supplement DOI: <https://doi.org/10.6084/m9.figshare.33722290>

Parent Article DOI: <https://doi.org/10.1364/OME.610184>

Moving forward with multilayer metastructures - simple design of complex optics: Supplemental document

DAVID A. B. MILLER

Ginzton Laboratory, Stanford University, 348 Via Pueblo Mall, Stanford CA 94305 USA
dabm@stanford.edu

S1 Overlapping nonlocality and meshes

The concept of overlapping nonlocality [1] is the idea that in the flow of the optical field from the input surface to the output surface in some linear optical device, some number C_{RL} of independent channels of information must flow from the right of the input surface to the left of the output surface, and similarly some number C_{LR} of channels must flow from the left of the input surface to the right of the output surface. The overlapping nonlocality C of this optical system is then the sum

$$C = C_{LR} + C_{RL} \quad (S1)$$

This number of channels can be thought of as the number that must be able to flow through a “transverse aperture” formed by dividing the input and output surfaces into their “left” and “right” halves, as in Fig. S1(a).

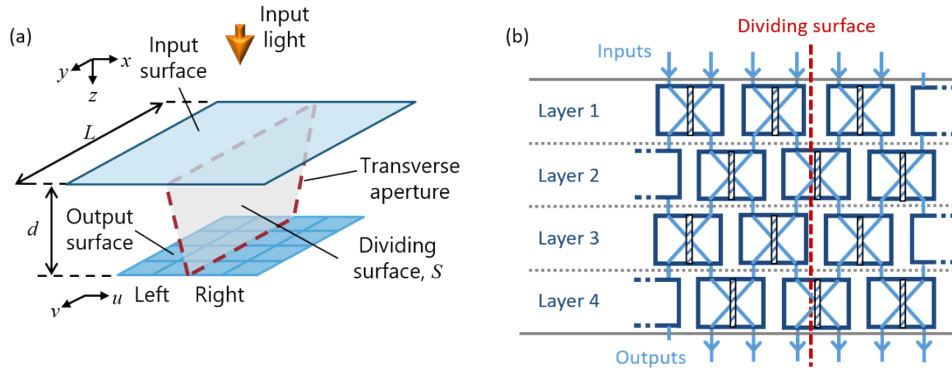


Fig. S1. Overlapping nonlocality. In (a), we divide a device mapping fields on the input surface to fields on the output surface using a dividing surface through both the input and output surfaces, defining a transverse aperture. The aperture must be large enough to support the overlapping nonlocality, the number of independent optical channels that must flow from the input on the left to the output on the right and from the input on right to the output on the left, adding the total to get the overlapping nonlocality C . (b) A portion of a four-layer mesh, with 2×2 blocks shown as beamsplitters (with additional phase shifters omitted for simplicity), showing that a dividing surface from the input to the output passes through 4 waveguides or paths, so supporting an overlapping nonlocality of up to 4.

One might think that the optics in the transverse aperture might just need to support one or other of these numbers of channels. For example, the cross-sectional area to support a number of waveguide modes flowing from inputs on the right to outputs on the left would be just the same as the cross-sectional area to support the same number of waveguide modes flowing from inputs on the left to outputs on the right. However, if our optics is reciprocal, then corresponding backward channels must exist. For example, for each such waveguide mode flowing from inputs on the left to outputs on the right, the backwards version of that waveguide mode, flowing backwards from the outputs on the right to the inputs on the left must also exist, and we note that is a channel that is flowing from right to left through the transverse aperture; the number of such backwards right-output-to-left-input channels is, by this reciprocity, also C_{LR} . So, the total number of channels that must exist for flowing from right to left through the transverse aperture is the sum as in Eq. (S1). We get the same answer for the total number of channels that must exist for flowing from left to right through the transverse aperture.

To understand what overlapping nonlocality is required from our optics, we can imagine that we start with some matrix D that represents the optical system we want to make. This is a matrix mapping input degrees of freedom – for example, fields at some discrete number of points on the input surfaces (or such samples of the input field) – to output degrees of freedom – for example pixels on some output sensor. We can immediately deduce just what overlapping nonlocality is required in the optics. We divide the matrix D into quadrants; we divide the columns of the matrix with a (vertical) line corresponding to the position of the transverse aperture on the input surface, and we divide the rows with a (horizontal) line corresponding to the position of the transverse aperture on the output surface. (See [1] Fig. S7 for an example of such a division of a matrix.) We can then perform singular value decomposition on, say, the right-inputs-to-left-outputs quadrant; the number of significant singular values (i.e., with magnitudes significantly larger than zero) tells us the number C_{RL} right-inputs-to-left-outputs channels we require. Similarly, considering the left-inputs-to-right-outputs quadrant, we deduce C_{LR} . Hence we can deduce the minimum required overlapping nonlocality we need in our optical system from Eq. (S1).

Notably, many optical operations require that the overlapping nonlocality is a significant number. In turn, this means that the operation cannot be performed just between the input and output surfaces of a single thin optical element because there are insufficient transverse channels available inside the element.

In the case of the layered structure discussed here, such as in the example in Fig. 1(c) or Fig. 4 of the main text, it is straightforward to understand just what overlapping nonlocality the structure can provide; we simply have to count the waveguides or beam paths that cross a corresponding “transverse aperture” surface, counting those that go from left to right and those that go from right to left, and adding the two to get the available overlapping nonlocality C of the structure.

In the case of the 2×2 blocks, we need to count the input and output waveguides or paths that must be crossed by a such dividing surface or line. This is easiest to see if we view the blocks as (controllable) beamsplitters as in Fig. S1(b) (with additional phase shifters, not shown) because then we do not get confused by “cutting through” the waveguides inside the MZI, which are merely there to provide the overall beamsplitting function, not to route channels from left to right or *vice versa*. Equivalently, we can view this more topologically, allowing ourselves to distort the dividing surface so that it routes round the internal waveguides; the overlapping nonlocality can then be viewed as the minimum number of paths that must be crossed by any line starting and ending at the relevant input and output positions.

We see in the example in Fig. S1(b), which corresponds to a section of the 4-layer mesh as in Fig. 1(c) and Fig. 4 in the main text, that the dividing surface cuts through exactly 4 waveguides – two going from left to right and two going from right to left – leading to this structure being able to support an overlapping nonlocality of up to $C = 4$.

One particularly important point for our proposed architectures is that they require devices (such as an MZI) that can function as 2×2 splitters, which require an overlapping nonlocality of 2. This is obvious just from such 2×2 beamsplitters in Fig. S1; any dividing surface through such a block necessarily crosses two paths. (Note that for a “Y” beamsplitter, such a surface need only cross one path.) In general, this means we should not expect to be able to fabricate a 2×2 beamsplitter just from one thin (e.g., subwavelength) layer.

S2 Matrix for a 2x2 self-configuring block

It is straightforward to deduce the matrix represented by a 2×2 MZI block once we have self-configured it to route all the input power to, say, the upper (“Right”) output as in Fig. 1(a) of the main text. We give this algebra explicitly here.

To keep the algebra as simple as possible, as in Fig. 1(a) of the main text, we use phase shifter pairs differentially; that is, the input phase shifters ϕ_T and ϕ_L are run as a differential pair, so with relative phase shift values $\phi_L = -\phi_T$, and a relative phase shift

$$\Delta\phi = \phi_T - \phi_L \quad (\text{S2})$$

Similarly, the internal phase shifters are also run as a differential pair with phase shift values $\phi_W = -\phi_P$, with relative phase shift

$$\Delta\theta = \theta_P - \theta_W \quad (\text{S3})$$

and with no “common-mode” phase shift from either pair. (We can always add the effect of any such common-mode phase shift ϕ_{CM} as an overall multiplying factor $\exp(i\phi_{CM})$ later if needed.)

We presume the input vector of mode amplitudes is

$$|e_{in}\rangle = \begin{bmatrix} a \\ b \end{bmatrix} \quad (\text{S4})$$

with a and b being the electric field amplitudes in the “Top” and “Left” inputs respectively. The resulting corresponding output vector $|e_{out}\rangle$ of amplitudes in the “Right” and “Bottom” outputs from the MZI block is the result of passing this input through the MZI. We describe this linear operation by the total matrix M_T , so we have

$$|e_{out}\rangle = M_T |e_{in}\rangle \quad (\text{S5})$$

Our task here is to find the matrix M_T if we self-configure this MZI block to route all the power to the upper (“Right”) output. (We could similarly find the corresponding matrix if instead we chose to route to the lower (“Bottom”) output.)

We can describe the overall matrix M_T in terms of a matrix M_ϕ corresponding to passing through the input ϕ phase shifters, which is just the diagonal matrix

$$M_\phi(\Delta\phi) = \begin{bmatrix} p & 0 \\ 0 & p^* \end{bmatrix} \quad (\text{S6})$$

where

$$p = \exp(i\Delta\phi / 2) \quad (\text{S7})$$

(and $p^* = \exp(-i\Delta\phi/2)$) and the matrix M_S corresponding to passing through the rest of the MZI block (with no output phase shifters θ_R or θ_B), which is derived in detail in Ref. [2] of the main text. Presuming 50:50 beamsplitters this is given by

$$M_S(\Delta\theta) = \begin{bmatrix} s & c \\ c & -s \end{bmatrix} \quad (S8)$$

where

$$s = \sin(\Delta\theta/2) \text{ and } c = \cos(\Delta\theta/2) \quad (S9)$$

M_T is then the product

$$M_T = M_S M_\phi = \begin{bmatrix} s & c \\ c & -s \end{bmatrix} \begin{bmatrix} p & 0 \\ 0 & p^* \end{bmatrix} = \begin{bmatrix} ps & p^*c \\ pc & -p^*s \end{bmatrix} \quad (S10)$$

or, explicitly,

$$M_T(\Delta\phi, \Delta\theta) = \begin{bmatrix} \exp\left(i\frac{\Delta\phi}{2}\right)\sin\left(\frac{\Delta\theta}{2}\right) & \exp\left(-i\frac{\Delta\phi}{2}\right)\cos\left(\frac{\Delta\theta}{2}\right) \\ \exp\left(i\frac{\Delta\phi}{2}\right)\cos\left(\frac{\Delta\theta}{2}\right) & -\exp\left(-i\frac{\Delta\phi}{2}\right)\sin\left(\frac{\Delta\theta}{2}\right) \end{bmatrix} \quad (S11)$$

Hence, with the input vector of amplitudes $|e_{in}\rangle$ as in Eq.(S4), the output vector $|e_{out}\rangle$ is

$$|e_{out}\rangle = \begin{bmatrix} aps + bp^*c \\ apc - bp^*s \end{bmatrix} \quad (S12)$$

Suppose that we self-configure this MZI block, minimizing the “Bottom” output power, first by adjusting the input differential phase shift $\Delta\phi$ and then the internal differential phase shift $\Delta\theta$ to minimize the “Bottom” output power to zero. Then, the “Bottom” field amplitude must be zero, so

$$apc = bp^*s \quad (S13)$$

Writing

$$a = |a|\exp(i\alpha), \quad b = |b|\exp(i\beta) \quad (S14)$$

from Eqs. (S7) and (S13) we therefore have

$$\frac{p}{p^*} = \exp(i\Delta\phi) = \frac{b}{a} \frac{s}{c} = \frac{|b|}{|a|} \exp[i(\beta - \alpha)] \tan\left(\frac{\Delta\theta}{2}\right) \quad (S15)$$

Since $\tan(\Delta\theta/2)$ is a real number, we therefore have

$$\Delta\phi = \beta - \alpha \quad (S16)$$

and

$$\Delta\theta = 2 \tan^{-1}(|a/b|) \quad (S17)$$

So, if we self-configure an MZI block using an input $|e_m\rangle$ as in Eq. (S4) so as to put all the output in the “Right” output port, using the forms (S14) for the input amplitudes, then the resulting matrix for the block is given by M_T as in Eq. (S11), with the resulting phase shifter settings and matrix parameters $\Delta\phi$ given by Eq. (S16) and $\Delta\theta$ given by Eq. (S17). Note these mathematical results are simple arithmetic calculations from the input used to set the block by self-configuration.

Note, incidentally, that this analysis clarifies that we only need a range of a total of π for the relative phase shift $\Delta\theta$. The argument in the inverse tangent in Eq. (S17) is always positive, which means that the resulting $\Delta\theta$ only runs between 0 and π .

Note also that this analysis proves mathematically another point that is obvious physically: for self-configuration, we require that we have controllable relative phase shift $\Delta\phi$ on the inputs to the block. If we put phase shifters only on the outputs of the block, then the matrix multiplication is reversed to give a matrix

$$\tilde{M}_{TOT} = M_\phi M_S = \begin{bmatrix} ps & pc \\ p^*c & -p^*s \end{bmatrix} \quad (S18)$$

With the input vector $|e_m\rangle$ as in Eq. (S4) as before, the “Bottom” output amplitude is $ap^*c - bp^*s$. For this to be zero, because the p^* is a common factor, we deduce $a/b = \tan(\Delta\theta/2)$, which is a real number. So, we can only get such cancellation if the input vector elements are in phase with one another, which in general they are not. Hence, without input phase shift control, we cannot in general self-configure the MZI to null out the “Bottom” output.

For completeness, we can repeat the calculation for the case where we want all the output in the “Bottom” output port. Starting from Eq. (S12), to route all the power to the “Bottom” output by minimizing the “Right” field amplitude to be zero, we have, instead of Eq. (S13),

$$aps = -bp^*c \quad (S19)$$

So

$$\begin{aligned} \frac{p^*}{p} &= \exp(-i\Delta\phi) = -\frac{a}{b} \frac{s}{c} = -\frac{|a|}{|b|} \exp[i(\alpha - \beta)] \tan\left(\frac{\Delta\theta}{2}\right) \\ &= \frac{|a|}{|b|} \exp[i(\alpha - \beta - \pi)] \tan\left(\frac{\Delta\theta}{2}\right) \end{aligned} \quad (S20)$$

Here we have chosen to replace the minus sign with $\exp(-i\pi)$. So, again, since $\tan(\Delta\theta/2)$ is a real number, we have

$$\Delta\phi = \beta - \alpha + \pi \quad (S21)$$

and

$$\Delta\theta = \tan^{-1}(|b/a|) \quad (S22)$$

S3 Use of an MZI block to measure wavelength

In the main text, we discussed the use of a single MZI block to measure the phase difference between the two inputs to the block. Another related use of individual blocks is to measure wavelength of a monochromatic beam. In this case, we also presume the beam is incident on

the block inputs at some angle α to the vertical. Then different wavelengths will have different phase differences between the block inputs.

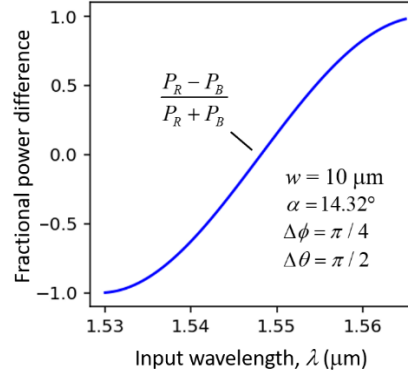


Fig. S2. Use of an MZI block to measure the wavelength of a beam incident at an angle of 14.45° . The two inputs to the block are spaced by 10 mm. Block settings are $\Delta\phi = \pi/4$ and $\Delta\theta = \pi/2$.

As an example, we choose settings of $\Delta\phi = \pi/4$ and $\Delta\theta = \pi/2$ for the block and calculate the resulting matrix M_T from Eq. (6) of the main text. Again, we choose $w = 10$ microns and now choose an input angle of 14.32° , which gives a useful coverage for the telecommunications C band. The phase difference between the two inputs $\phi_a = (2\pi/\lambda)w\sin\alpha$ as in Eq. (10) of the main text and the calculated fractional power difference between the two outputs is shown in Fig. S2, showing that such a simple block could also be used for wavelength measurement.

S4 Mesh matrices as unitary operators

There is a subtle technical point as to whether our mesh of 2×2 blocks is unitary. Though the rows of the matrix U_M describing the mesh behavior are orthogonal and normalized, U_M may not be regarded technically as a unitary matrix because it has different numbers of rows and columns; such matrices can be called semi- or partially unitary. However, as an operator that maps from one Hilbert space (the “input” space) to another (the “output” space), it is unitary in the general sense because those Hilbert spaces, spanned respectively by the Hermitian adjoints of the rows (“input” space) and the columns (“output” space) of the matrix, have the same dimensionality (here, 8). If we were to regard the input space as being 14-dimensional in our example (because there are 14 inputs) rather than 8 dimensional, so capable of representing any 14 element vectors, then indeed the operation is not unitary; in particular, physically, there could be power routed out of the other (un-numbered and unconnected) “drop” ports in the 2×2 blocks at the left and right of the structure (such as the left port of block B11), which would make the system physically non-unitary because it has loss. With this understanding, though, we can essentially view these mesh matrices as unitary.

S5 Self-configuring algorithm for setting the trapezoidal block layer mesh for arbitrary orthogonal inputs

Here we give a full version of the configuration algorithm for the mesh as in Fig. 4 in the main text for the general case; so, we presume no particular space-invariance, with each successive configuration Vector is as independent as it can be if it is to be routed to a different output. The algorithm corresponding to the matrix U_M as in Eq. (14) of the main text, is as follows.

Layer 1 settings

Shine in only Vector 1

Adjust block B11 to set the output power in its left output port to zero
Adjust block B14 to set the output power in its right output port to zero.
(Note: these steps ensure both that power is flowing “rightwards” from B11 towards Output 1 and “leftwards” from B14 towards Output 1)

Shine in only Vector 2

Adjust block B12 to set the output power in its left output port to zero
Adjust block B15 to set the output power in its right output port to zero.

Shine in only Vector 3

Adjust block B13 to set the output power in its left output port to zero
Adjust block B16 to set the output power in its right output port to zero

Shine in only Vector 4

(We do not need to adjust block B13 to set the output power in its left output port to zero because B13 has already been set in the Vector 1 steps above.)

Adjust block B17 to set the output power in its right output port to zero

Layer 2 settings

Shine in only Vector 1

Adjust block B21 to set the output power in its left output port to zero
Adjust block B23 to set the output power in its right output port to zero.
(Note: these steps ensure both that power is flowing “rightwards” from B21 towards Output 1 and “leftwards” from B23 towards Output 1)

Shine in only Vector 2

Adjust block B22 to set the output power in its left output port to zero
Adjust block B24 to set the output power in its right output port to zero

Shine in only Vector 3

(We do not need to adjust block B23 to set the output power in its left output port to zero because B23 has already been set in the Vector 1 steps above.)

Adjust block B25 to set the output power in its right output to zero

Shine in only Vector 4

(We do not need to adjust block B24 to set the output power in its left output port to zero because B24 has already been set in the Vector 2 steps above.)

Adjust block B26 to set the output power in its right output to zero

Layer 3 settings

Shine in only Vector 1

Adjust block B31 to set the output power in its left output port to zero
Adjust block B32 to set the output power in its right output port to zero.

Shine in only Vector 2

Adjust block B33 to set the output power in its right output port to zero

Shine in only Vector 3

Adjust block B34 to set the output power in its right output port to zero

Shine in only Vector 4

Adjust block B35 to set the output power in its right output port to zero

Layer 4 settings

Shine in only Vector 1

Adjust block B41 to set the power in its right output port to zero

Shine in only Vector 2

Adjust block B42 to set the power in its right output port to zero

Shine in only Vector 3

Adjust block B43 to set the power in its right output port to zero

Shine in only Vector 4

Adjust block B44 to set the power in its right output port to zero

Having set this first trapezoid of blocks this way, we then proceed with a similar but slightly simpler algorithm with successive vectors to set up any remaining blocks in a wider array. In

this case, we could continue with a Vector 5, which is shifted over by one more block to the right and which must be orthogonal to Vectors 2, 3, and 4 (it is automatically orthogonal to Vector 1 because it does not overlap with it). The algorithm for Vector 5 is

Shining in only Vector 5

Adjust block B18 to set the output power in its right output port to zero

Adjust block B27 to set the output power in its right output port to zero

Adjust block B36 to set the output power in its right output port to zero

Adjust block B45 to set the output power in its right output port to zero

Note that this algorithm for Vector 5 is like that for a diagonal line self-configuring layer, just with the constraints on Vector 5 being orthogonal to the preceding 3 Vectors (here Vectors 2, 3, and 4).

We can continue with further Vectors, each shifted to the right by one block, and each orthogonal to the preceding 3 Vectors. We see then that, from Vector 4 onwards, the mesh is being configured as a set of successive diagonal line self-configuring layers: blocks B17, B26, B35, and B44 for Vector 4; blocks B18, B27, B36, and B45 for Vector 5; and so on. In this way, we can set an arbitrarily wide set of such layers of blocks with these various 8-element shifted input Vectors of amplitudes.

For all these algorithms, note that there are additional Vectors $1'$, $2'$, $3'$, $4'$, $5'$, and so on (not shown), of the same size and positions as the Vectors 1, 2, 3, 4, 5, and so on, that will be routed to the other outputs of the last block layer of blocks, i.e., outputs 2, 4, 6, 8, 10, and so on. These complementary vectors are completely determined by the mesh as set by the Vectors 1, 2, 3, 4, 5, and so on. They have no additional degrees of freedom (other than possibly their overall phase). To understand what they are, and to calculate them, once we have set the mesh with Vectors 1, 2, 3, 4, 5, and so on, we now imagine shining light back into these outputs 2, 4, 6, 8, 10, and so on, one at a time. The vector of amplitudes that emerges backwards from the corresponding set of inputs in each case will be the phase conjugate of the corresponding Vector $1'$, $2'$, $3'$, $4'$, $5'$, and so on.

We have illustrated this process of configuration of these block layers of blocks for the example case of 4 such block layers. The extension of other numbers of layers is straightforward. For m such block layers, we will be using vectors with $2m$ elements for the configuration or calculation, with each successive vector orthogonal to the preceding $m-1$ configuration Vectors.

S6 Algorithm based only on self-configuring diagonal lines

If we are prepared to discard a few inputs and outputs at the “left” of the mesh, we can self-configure the mesh just using each configuration vector once. The algorithm is illustrated in Fig. S3.

In this algorithm, we presume that we have a set of configuration Vectors, each $2m$ inputs long, and each orthogonal to every other such Vector, just as in the algorithm above in Section S5. However, now we block off the first $m-2$ inputs to the mesh (i.e., Inputs 1 through 6 in our example). Also, though we may use photodetectors in the outputs of the first $m-1$ output blocks – Outputs 1, 3, and 5 in our example – we otherwise discard the Outputs from these blocks (Outputs 1 through 6 in our example).

The algorithm then proceeds as follows:

Shine in only Vector 1

Configure the first diagonal line (which starts with Inputs 7 and 8) to route all the power to the left output at the end of the diagonal line (here Output 1)

Proceed similarly for all diagonal lines the mesh, working progressively to the right, with the n th such diagonal line configured with the n th Vector.

Note in this algorithm that we may set the blocks marked with a cross to “cross” state to start with. (Indeed, we could choose to set all blocks to “cross” state to start with, which makes the

use of output photodetectors easier as the requisite signals are already routed to them.) Even if we do not set them to “cross”, an algorithm to maximize the power from these first $m - 1$ output blocks will end up setting such blocks to “cross” anyway.

Note that, when shining in a given Vector n and maximizing the power from the bottom left output of the corresponding diagonal line n , we can simultaneously be shining any power into the inputs to the right of that Vector because none of it can be routed to that bottom left port. This could be particularly convenient if setting up a periodic mesh as in Section S8 below because we could simply shine in a complete set of replicated versions of that Vector into the successive unit cells of the structure. (So, Vector 5 is the same as Vector 1, Vector 6 is the same as Vector 2, and so on.) Then the entire periodic structure can be set up with just m such repetitive input vectors. (Note that, because of the way we have constructed this algorithm, it would be more convenient to view the unit cell of such a crystal structure as a parallelogram skewed to the left rather than skewed to the right as in Fig. S3. Either one is a possible choice of unit cell for this structure.)

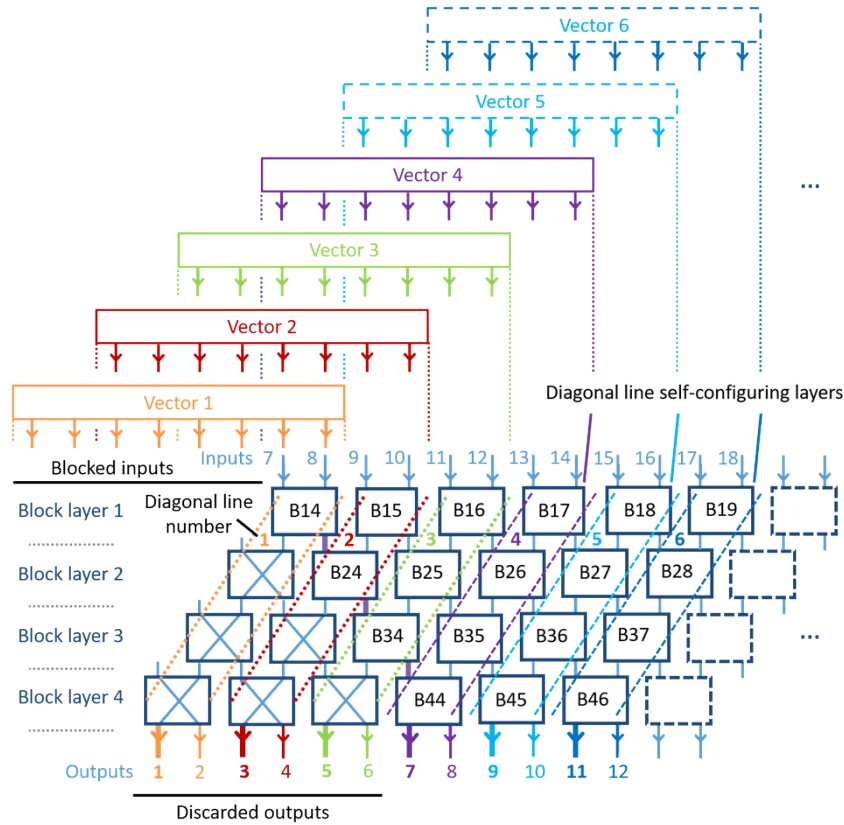


Fig. S3. Architecture for configuring the mesh using just self-configuration of a succession of diagonal lines, one per (orthogonal) input Vector, to maximize output power at the bottom left output of the diagonal line when illuminated with the corresponding Vector. In this architecture, Inputs 1 through 6 are blocked, and the signals from Outputs 1 through 6 are discarded except for their use during configuration. The mesh is then fully configured for Outputs 7 and higher and for Vectors 4 and higher.

S7 Periodic trapezoidal architectures as photonic crystals

The analysis of the trapezoidal architecture in the main text and in sections S5 and S6 above is general; for a mesh with m layers of 2×2 blocks; every one of the $2m$ element configuration Vectors 1, 2, 3, ... (or the corresponding matrix rows with $4m - 2$ non-zero elements $|\psi_1\rangle$,

$|\psi_3\rangle, |\psi_7\rangle \dots$) can be chosen quite arbitrarily to be different from any other as long as each such Vector or matrix row is orthogonal to the others; in this case potentially each 2×2 block could be configured differently from every other block. As mentioned there, the trapezoidal architecture can be arbitrarily extended sideways with more diagonal lines, each with its corresponding additional configuration Vector.

An interesting subset of such universal trapezoidal architectures is structures that are “spatially invariant” – that is, they look the same if translated sideways by some specific amount. In this case, we would have “unit cells” of blocks; the blocks within each unit cell might all be configured differently, but the configuration of every unit cell of blocks would be the same.

With such unit cells, the structure becomes technically a kind of photonic crystal, though with the unusual characteristic that the light only flows in the forward direction; we can refer to these as ‘forward’ photonic crystals. This characteristic makes their modeling particularly simple; calculating their behavior merely requires matrix multiplications. We examine some classes of these photonic crystals below.

Mesh with transverse periodicity of m blocks for an m layer mesh

Consider a mesh of m layers of blocks that is also periodic in the transverse direction (i.e., within each layer) where the period or “unit cell size” in the transverse direction is also m blocks, or equivalently a periodicity of

$$q = 2m \quad (\text{S23})$$

inputs. We construct such a mesh physically by adding arbitrary numbers of “diagonal lines” of blocks as in Fig. 1(c) or Fig. 4 of the main text. As we discuss below, this mesh can be viewed as being made up from parallelogram unit cells stacked side by side (Fig. S4).

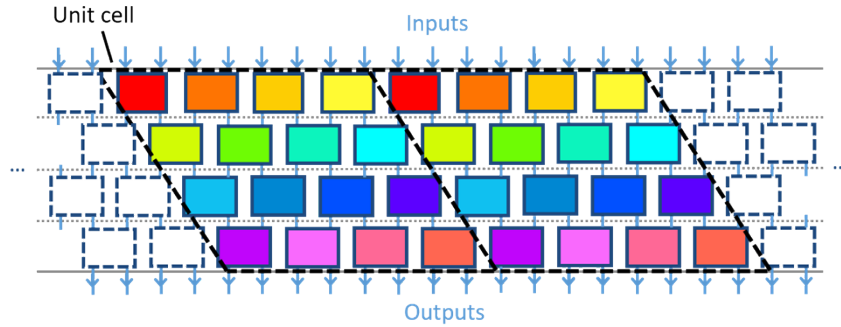


Fig. S4. Sketch of 2 unit cells of a forward photonic crystal structure made from 4 layers of blocks in a mesh. Each of the 16 blocks in a unit cell, shown in different colors, may be set differently in the design or configuration of the mesh, with these block settings replicated in other unit cells.

This approach allows the mesh to have $2m$ different functions that repeat with periodicity of m blocks. This mesh can be set up with m Vectors just as for the arbitrary mesh above, with those m Vectors shifted right by m blocks (or $q = 2m$ inputs) to set up each successive set of m blocks; equivalently, we simply replicate the settings for one such unit cell into the others.

The matrix for this mesh is made up with a set of blocks along the diagonal, with zeros elsewhere. Each such matrix block, which has dimensions $2m \times (4m - 2)$, is in the form of U_M as in Eq. (14) of the main text; so in our example with $m = 4$, these blocks are 8×14 . The resulting matrix can be written for this example as

$$M = \begin{bmatrix} \ddots & & & & \\ & U_M & & & \\ & & U_M & & \\ 0 & & & U_M & \\ & & & & \ddots \end{bmatrix} \quad (S24)$$

Note that these blocks are staggered as shown, with each block shifted down and to the right by $2m$ rows and $2m$ columns. If we ignore the grouping into the larger U_M blocks, the resulting matrix is a block diagonal matrix with blocks that each are 2 rows by $4m - 2$ (here 14) columns, just as we see inside U_M as in Eq. (14) of the main text.

Now we proceed to construct a way of looking at this system like a one-dimensional photonic crystal viewed in transmission from inputs to outputs. To analyze this system, we start by looking for eigenvectors – overall input vectors $|\psi\rangle$ that extend over the whole larger crystal and that map to output versions $|\psi_o\rangle$ of themselves just multiplied by an eigenvalue μ , i.e.,

$$|\psi_o\rangle = \mu |\psi\rangle \quad (S25)$$

Note this eigenproblem is slightly different from many typical physical eigenproblems, such as in resonators, in which the mapping is from a space back onto itself; here the mapping is from the input space to the output space, but we are still interested in functions that reproduce themselves within some factor (the eigenvalue μ) in this mapping.

Incidentally, we know immediately that, if such eigenvectors exist, because the mesh is nominally lossless, the eigenvalue must be a unit amplitude complex number, i.e., defining the corresponding “eigenphase” η , we have

$$\mu = \exp(i\eta) \quad (S26)$$

so the interesting behavior of such eigenvectors in passing through the mesh will just be a phase shift η .

We look for eigenvectors that have some kind of periodicity associated with the periodicity of the mesh “lattice”, so with a periodicity of $q = 2m$ inputs (and outputs) as in Eq. (S23). q is also the number of matrix rows associated with each period of this photonic crystal lattice. We take the individual elements of the vector $|\psi_o\rangle$ to be the complex numbers $\psi(s)$, where s indexes the mesh outputs; note s is an effective position coordinate, including within the unit cell. We take the transmitted power in such an output s to be

$$P(s) = |\psi(s)|^2 \quad (S27)$$

in some convenient units. Now we expect that, when illuminated by such an eigenvector, the transmitted power from the ports will look the same in every period of the mesh. So,

$$|\psi(s+q)|^2 = |\psi(s)|^2 \quad (S28)$$

which means that

$$\psi(s+q) = C\psi(s) \quad (S29)$$

where C is a unit complex number. Next, we indulge in the usual fallacy of periodic boundary conditions in which we persuade ourselves that the behavior of an infinitely long chain of such periods of the mesh could not really be any different from that of a very long chain joined onto itself in a loop. Taking S periods in that loop, we then also argue that the resulting amplitudes must be single-valued so these amplitudes round the loop must join onto one another. Hence, we argue that

$$\psi(s + Sq) = C^S \psi(s) = \psi(s) \quad (\text{S30})$$

So,

$$C^S = 1 \quad (\text{S31})$$

and so C is one of the S roots of unity. (We are, of course, deriving a form of the Bloch theorem.) In keeping with common ways of writing this in solid state physics, we can choose to write

$$C = \exp(i\kappa q) \quad (\text{S32})$$

where the “wavevector” κ (here, a dimensionless number) is any of the elements

$$\kappa = \frac{2\pi p}{Sq}; \quad p = 0, \pm 1, \pm 2, \dots, \pm \frac{S}{2} \quad (\text{S33})$$

(where technically we have to drop one or other of the end points $+S/2$ or $-S/2$ to get the correct number S of distinct values). From Eqs. (S29) and (S32) we therefore have one form of the Bloch theorem

$$\psi_\kappa(s + q) = \exp(i\kappa q) \psi_\kappa(s) \quad (\text{S34})$$

where we are now explicitly writing the ψ for a given κ by subscripting it. Defining

$$u_\kappa(s) = \psi_\kappa(s) \exp(-i\kappa s) \quad (\text{S35})$$

we have the more common form of the Bloch theorem

$$\psi_\kappa(s) = u_\kappa(s) \exp(i\kappa s) \quad (\text{S36})$$

where

$$u_\kappa(s + q) = u_\kappa(s) \quad (\text{S37})$$

So, $u_\kappa(s)$ can be thought of as a “unit cell” function that is the same in every “unit cell” – here every set of q inputs or $m = q/2$ blocks. The overall eigenfunction is then written in the form of Eq. (S36), which is this unit cell function multiplied by an envelope complex exponential “wave” $\exp(i\kappa s)$. As usual with the Bloch theorem, we are not solving the full problem yet, but we are saying that any eigen solution of the problem must have the form as in Eq. (S36).

The final step in our analysis is to deduce the eigen equation for $u_\kappa(s)$, which will in general be different for each choice of “wavevector” κ . For this, we presume, without loss of generality, that we are interested in outputs 1 through q (1 through 8 here), as being a representative unit cell block of the crystal. To calculate that output with an input in Bloch form, we note that the fields in all the inputs 1 through $2q - 2$ (1 through 14 here) influence the final output fields in this set of q output ports. If the inputs 1 through q (1 through 8 here)

have amplitudes $\psi_\kappa(1), \psi_\kappa(2), \dots, \psi_\kappa(q)$, then, because the input wave is presumed to be in Bloch form as in Eq. (S34), the fields in inputs $q+1$ to $2q-2$ will be of the form

$$\psi_\kappa(q+1) = e_\kappa \psi_\kappa(1), \psi_\kappa(q+2) = e_\kappa \psi_\kappa(2), \dots, \psi_\kappa(2q-2) = e_\kappa \psi_\kappa(q-2) \quad (\text{S38})$$

where we are using a shorthand

$$e_\kappa = \exp(ikq) \quad (\text{S39})$$

So, the matrix multiplication that determines the outputs 1 through q to give the output vector for these q elements is

$$\begin{bmatrix} \psi_{o\kappa}(1) \\ \vdots \\ \psi_{o\kappa}(q) \end{bmatrix} = \begin{bmatrix} u_{1,1} & u_{1,2} & u_{1,3} & u_{1,4} & u_{1,5} & u_{1,6} & u_{1,7} & u_{1,8} & 0 & 0 & 0 & 0 & 0 & 0 \\ u_{2,1} & u_{2,2} & u_{2,3} & u_{2,4} & u_{2,5} & u_{2,6} & u_{2,7} & u_{2,8} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & u_{3,3} & u_{3,4} & u_{3,5} & u_{3,6} & u_{3,7} & u_{3,8} & u_{3,9} & u_{3,10} & 0 & 0 & 0 & 0 \\ 0 & 0 & u_{4,3} & u_{4,4} & u_{4,5} & u_{4,6} & u_{4,7} & u_{4,8} & u_{4,9} & u_{4,10} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & u_{5,5} & u_{5,6} & u_{5,7} & u_{5,8} & u_{5,9} & u_{5,10} & u_{5,11} & u_{5,12} & 0 & 0 \\ 0 & 0 & 0 & 0 & u_{6,5} & u_{6,6} & u_{6,7} & u_{6,8} & u_{6,9} & u_{6,10} & u_{6,11} & u_{6,12} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & u_{7,7} & u_{7,8} & u_{7,9} & u_{7,10} & u_{7,11} & u_{7,12} & u_{7,13} & u_{7,14} \\ 0 & 0 & 0 & 0 & 0 & 0 & u_{8,7} & u_{8,8} & u_{8,9} & u_{8,10} & u_{8,11} & u_{8,12} & u_{8,13} & u_{8,14} \end{bmatrix} \times \begin{bmatrix} \psi_\kappa(1) \\ \vdots \\ \psi_\kappa(q) \\ e_\kappa \psi_\kappa(q+1) \\ \vdots \\ e_\kappa \psi_\kappa(2q-2) \end{bmatrix} \quad (\text{S40})$$

Now we note a mathematical trick. We first move the triangle of matrix elements in the last $2m-2$ columns on the right over to fill in the zeros on the left, multiplying them by e_κ . (Formally this is equivalent to multiplying the last $2m-2$ columns by e_κ and adding them to the first $2m-2$ columns.) Then we remove the last $2m-2$ columns to get a new $q \times q$ matrix. In our example, this matrix becomes

$$\Psi_{C\kappa} = \begin{bmatrix} u_{1,1} & u_{1,2} & u_{1,3} & u_{1,4} & u_{1,5} & u_{1,6} & u_{1,7} & u_{1,8} \\ u_{2,1} & u_{2,2} & u_{2,3} & u_{2,4} & u_{2,5} & u_{2,6} & u_{2,7} & u_{2,8} \\ e_{\kappa}u_{3,7} & e_{\kappa}u_{3,8} & u_{3,1} & u_{3,2} & u_{3,3} & u_{3,4} & u_{3,5} & u_{3,6} \\ e_{\kappa}u_{4,7} & e_{\kappa}u_{4,8} & u_{4,1} & u_{4,2} & u_{4,3} & u_{4,4} & u_{4,5} & u_{4,6} \\ e_{\kappa}u_{5,5} & e_{\kappa}u_{5,6} & e_{\kappa}u_{5,7} & e_{\kappa}u_{5,8} & u_{5,1} & u_{5,2} & u_{5,3} & u_{5,4} \\ e_{\kappa}u_{6,5} & e_{\kappa}u_{6,6} & e_{\kappa}u_{6,7} & e_{\kappa}u_{6,8} & u_{6,1} & u_{6,2} & u_{6,3} & u_{6,4} \\ e_{\kappa}u_{7,3} & e_{\kappa}u_{7,4} & e_{\kappa}u_{7,5} & e_{\kappa}u_{7,6} & e_{\kappa}u_{7,7} & e_{\kappa}u_{7,8} & u_{7,1} & u_{7,2} \\ e_{\kappa}u_{8,3} & e_{\kappa}u_{8,4} & e_{\kappa}u_{8,5} & e_{\kappa}u_{8,6} & e_{\kappa}u_{8,7} & e_{\kappa}u_{8,8} & u_{8,1} & u_{8,2} \end{bmatrix} \quad (\text{S41})$$

If we now start with a mathematical input vector with just the q elements corresponding to the unit cell, the resulting matrix-vector multiplication leads to exactly the same output results as in Eq. (S40),

$$\begin{bmatrix} \psi_{o\kappa}(1) \\ \vdots \\ \psi_{o\kappa}(q) \end{bmatrix} = \Psi_{C\kappa} \begin{bmatrix} \psi_{\kappa}(1) \\ \vdots \\ \psi_{\kappa}(q) \end{bmatrix} \quad (\text{S42})$$

Now we write the various ψ_{κ} and $\psi_{o\kappa}$ using the form as in Eq. (S36), giving

$$\begin{bmatrix} u_{\kappa}(1)\exp(i\kappa) \\ \vdots \\ u_{\kappa}(q)\exp(i\kappa q) \end{bmatrix} \equiv D_{\kappa} \begin{bmatrix} u_{\kappa}(1) \\ \vdots \\ u_{\kappa}(q) \end{bmatrix}, \quad \begin{bmatrix} u_{o\kappa}(1)\exp(i\kappa) \\ \vdots \\ u_{o\kappa}(q)\exp(i\kappa q) \end{bmatrix} \equiv D_{\kappa} \begin{bmatrix} u_{o\kappa}(1) \\ \vdots \\ u_{o\kappa}(q) \end{bmatrix} \quad (\text{S43})$$

where D_{κ} is the diagonal matrix

$$D_{\kappa} = \begin{bmatrix} \exp(i\kappa) & 0 & \cdots & 0 \\ 0 & \exp(i2\kappa) & & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & \exp(i\kappa q) \end{bmatrix} \quad (\text{S44})$$

Hence, Eq. (S42) becomes

$$D_{\kappa} \begin{bmatrix} u_{o\kappa}(1) \\ \vdots \\ u_{o\kappa}(q) \end{bmatrix} = \Psi_{C\kappa} D_{\kappa} \begin{bmatrix} u_{\kappa}(1) \\ \vdots \\ u_{\kappa}(q) \end{bmatrix} \quad (\text{S45})$$

Noting that, since D_{κ} is unitary, its inverse $D_{\kappa}^{-1} \equiv D_{\kappa}^{\dagger}$, which is its Hermitian adjoint, and introducing the short-hand notations

$$\mathcal{U}_{\kappa} \equiv \begin{bmatrix} u_{\kappa}(1) \\ \vdots \\ u_{\kappa}(q) \end{bmatrix}, \quad \mathcal{U}_{o\kappa} \equiv \begin{bmatrix} u_{o\kappa}(1) \\ \vdots \\ u_{o\kappa}(q) \end{bmatrix} \quad (\text{S46})$$

we can rearrange Eq. (S45) to obtain

$$\mathcal{U}_{o\kappa} = \mathcal{U}_{C\kappa} \mathcal{U}_{\kappa} \quad (\text{S47})$$

where

$$U_{C\kappa} = D_{\kappa}^{\dagger} \Psi_{C\kappa} D_{\kappa} \quad (S48)$$

is now the “unit cell” matrix.

The corresponding eigenequation for the unit cell functions (or vectors) is

$$U_{C\kappa} U_{\kappa p} = \mu_{\kappa p} U_{\kappa p} \equiv \exp(i\eta_{\kappa p}) U_{\kappa} \quad (S49)$$

where we introduce a “band” index p since we expect in general there will be q different eigensolutions for each choice of κ , and have indexed all eigenvectors and eigenvalues with their corresponding values of κ and p .

Note that, in this periodic case, the resulting full matrix for the mesh can be written as a block diagonal matrix with square blocks $U_{C\kappa}$, as

$$M = \begin{bmatrix} \ddots & & & \\ & U_{C\kappa} & & 0 \\ & \vdots & \ddots & \\ q \text{ rows} & & & U_{C\kappa} \\ & 0 & & \vdots \\ & & q \text{ columns} & U_{C\kappa} \\ & & & \ddots \end{bmatrix} \quad (S50)$$

The problem of evaluating eigenvectors and eigenvalues is now reduced just to solving for the eigenvectors and eigenvalues of a square matrix $U_{C\kappa}$ of size $2m$, where m is the number of rows of blocks in the periodic mesh. This will enable us to calculate a “band structure” of phase shifts η as a function of the “wavevector” κ . Note that, if the physical inputs to the mesh are equally spaced by some amount a , we can evaluate the corresponding physical wavevector component k along the input surface of the mesh by multiplying by a , i.e.,

$$k = \kappa a \quad (S51)$$

hence establishing physical eigenvectors for the incoming waves.

The self-configuring algorithm for setting up the mesh in this periodic case is given in detail below in Supplemental document section S8. For the simpler special case where the configuration Vectors are all just correspondingly shifted versions of Vector 1, which we could refer to as simple periodicity, the resulting algorithm is given below in Supplemental document section S9. The spatial filter based on Daubechies wavelets discussed in the main text actually corresponds to this simpler case.

The eigenfunctions for this forward photonic crystal are the transverse functions that reproduce themselves with a phase factor between the input – the “top” of the multilayer structure – and the output – the “bottom” of the multilayer structure as in Fig. S4. (The output functions may be moved slightly sideways compared to the input functions, by $m-1$ “positions” if we draw the mesh with rectangular blocks and vertical connecting lines or “waveguides” between the blocks as in Fig. 4 of the main text, but if we use the numbering system for inputs and outputs as in Fig. 4 of the main text, the eigenfunctions reproduce themselves between inputs and outputs.)

It would also be possible to stack multiple such sets of m layers one on top of the other as sketched in Fig. S5, showing three such sets of layers.

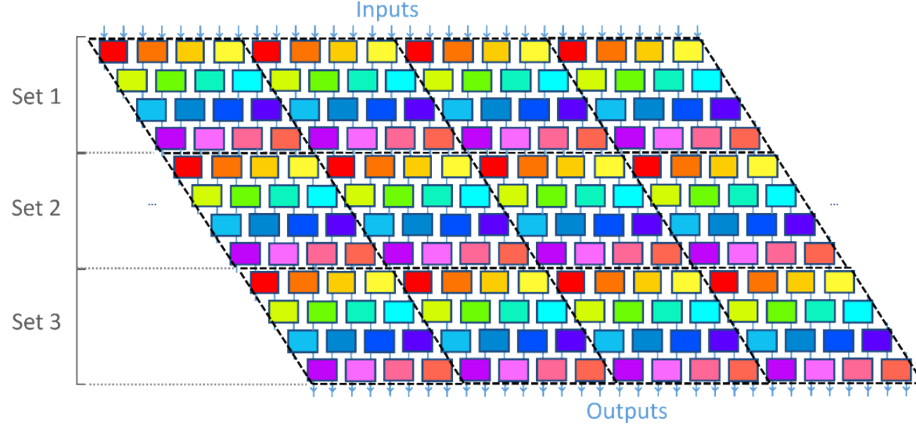


Fig. S5. Stacking of multiple sets of m layers of blocks, forming a “forward photonic crystal” also now in the vertical direction.

In this case the structure would also become periodic in the “vertical” direction (or, more precisely, along the direction of the sides of the parallelogram unit cells), and the eigenfunctions introduced here would reproduce themselves in each successive set of m layers, with a phase factor of $\exp(i\eta_\kappa)$ between successive sets of m layers. So, if we index the successive sets of m layers of blocks by index Q , we have

$$|\psi_{Cp\kappa, Q+1}\rangle = \exp(i\eta_{p\kappa}) |\psi_{Cp\kappa, Q}\rangle \quad (\text{S52})$$

which is a Bloch-like form as in Eq. (S34), but now in this “vertical” direction. In this sense, we would now have a 2-dimensional photonic crystal structure.

The effect of using n such sets of layers in sequence would be simple to magnify the eigenvalue phase shifts by n – i.e.,

$$|\psi_{Cp\kappa, Q+n}\rangle = \exp(in\eta_{p\kappa}) |\psi_{Cp\kappa, Q}\rangle \quad (\text{S53})$$

Note that this is a kind of “logical” crystal structure in the vertical direction, being periodic in the sets of blocks. Those sets of blocks would not have to be precisely spaced vertically to get this kind of Bloch-like behavior with respect to the index p , at least for the relative phase between eigenfunctions of different κ and p . Such overall spacing of these sets of layers would affect the absolute phase of light propagating through the structure, but not the relative phases of the different eigenstates.

Non-unitary periodic structure

We could use the non-unitary construction as in Fig. 9 in the main text in a lateral forward photonic crystal configuration, following on from the unitary case of Fig. S4. We would use a parallelogram of blocks to feed through the amplitude modulators and phase shifters to another parallelogram of blocks, as in the example unit cell in Fig. S6.

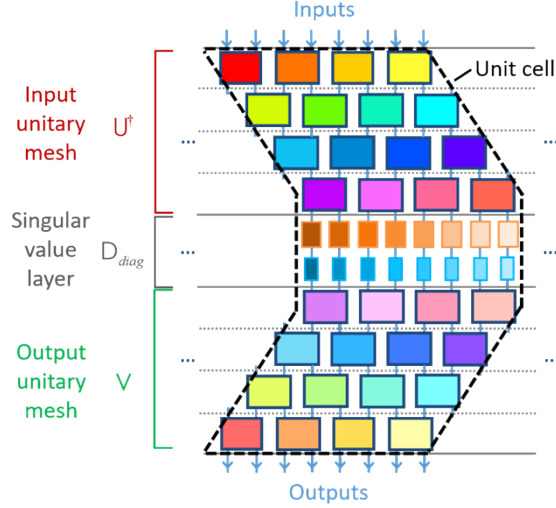


Fig. S6. Example unit cell for a non-unitary forward photonic crystal mesh, here for 4 layers in each unitary mesh. In this example the output unitary mesh is oriented in the opposite direction to the input unitary mesh to realign the Inputs and Outputs.

S8 Self-configuring algorithm for periodicity of m blocks for an m block layer mesh

If the mesh is to be space-invariant in groups of size equal to the number of layers of 2×2 blocks (here 4 layers), so block $B_{i(j+4)}$ is set to be the same as block B_{ij} (e.g., B25 is set the same as B21), then we can use the following algorithm. This algorithm presumes that the blocks are calibrated or identically fabricated so we can replicate the settings from blocks in one unit cell to those in the next. If this is not the case, then we can configure with the algorithm in Supplemental document Section S5 above using shifted versions of the configuration Vectors for successive unit cells or diagonal lines.

Layer 1 settings

Shine in only Vector 1

Adjust block B11 to set the output power in its left output port to zero

Replicate settings to B15

Adjust block B14 to set the output power in its right output port to zero.

Comment: these steps ensure both that power is flowing “rightwards” from B11 towards Out 1 and “leftwards” from B14 towards Out 1

Shine in only Vector 2

Adjust block B12 to set the output power in its left output port to zero

Replicate settings to B16

Comment: We could then adjust block B15 to set the output power in its right output port to zero, but B15 will already have been set as the replica of B11.

Shine in only Vector 3

Adjust block B13 to set the output power in its left output port to zero

Replicate settings to B17

Comment: We could then adjust block B16 to set the output power in its right output port to zero, but B16 will already have been set as the replica of B12.

Comment: We do not need to shine in Vector 4 because there is nothing left for it to set in this layer.

Layer 2 settings

Shine in only Vector 1

Adjust block B21 to set the output power in its left output port to zero
Replicate settings to B25
Adjust block B23 to set the output power in its right output port to zero.
Comment: these steps ensure both that power is flowing “rightwards” from B21 towards Out 1 and “leftwards” from B23 towards Out 1

Shine in only Vector 2
Adjust block B22 to set the output power in its left output port to zero
Replicate settings to B26
Adjust block B24 to set the output power in its right output port to zero

Layer 3 settings

Shine in only Vector 1
Adjust block B31 to set the output power in its left output port to zero
Replicate settings to B35
Adjust block B32 to set the output power in its right output port to zero.

Shine in only Vector 2
Adjust block B33 to set the output power in its right output port to zero

Shine in only Vector 3
Adjust block B34 to set the power in its right output port to zero

Layer 4 settings

Shine in only Vector 1
Adjust block B41 to set the power in its right output port to zero

Shine in only Vector 2
Adjust block B42 to set the power in its right output port to zero

Shine in only Vector 3
Adjust block B43 to set the power in its right output port to zero

Shine in only Vector 4
Adjust block B44 to set the power in its right output port to zero

This process will have set all the blocks B11 to B44. For this horizontally periodic system, we could then proceed to replicate these blocks periodically to the right, stepping by 4 blocks to the right to get a periodic structure with period 4 blocks in the “horizontal” direction. Note that the blocks B15, B16, B17, B25, B26 and B35 have already been set this way by the replications mentioned in the algorithm above.

S9 Self-configuring algorithm for setting the block layer mesh for simple shifted inputs

This algorithm can configure a mesh that has simple periodicity – that is, every block in a given layer is set the same so the mesh is space-invariant as we move by any integer number of blocks.

Comment: perform “collection triangle” settings for Output 1

Layer 1 settings

Shine in only Vector 1

Adjust B11 to set the output power in its left output port to zero

Comment: This is required because we need all the power to flow towards the bottom block B14, which it can only do if B11 routes its output power to its right port.

Comment: At this point we could adjust B14 to set the output power in its right output port to zero; this is required because we need all the power to flow towards the bottom block B14, which it can only do if B14 routes its output power to its right port. However, for simplicity of the algorithm we do not need to do this here because we can set this same block later using Vector 4. The fact that there are two algorithmic ways of setting this block is actually telling us there is a necessary constraint on the last two elements of Vector 1, which is that that two element

vector is orthogonal to the two element vector that is the first two elements of Vector 1

Shine in only Vector 2 (which is just a shifted version of Vector 1 in this case)

Adjust B12 to set the output power in its left output port to zero

Comment: This is required because we need all the power to flow towards the bottom block B14, which it can only do if B11 routes its output power to its right port.

Shine in only Vector 3 (which is just a shifted version of Vector 1 in this case)

Adjust B13 to set the output power in its left output port to zero

Shine in only Vector 4 (which is just a shifted version of Vector 1 in this case)

Adjust B14 to set the output power in its left output port to zero

Layer 2 settings

Shine in only Vector 1

Adjust B21 to set the output power in its left output port to zero

Shine in only Vector 2

Adjust B22 to set the output power in its left output port to zero

Shine in only Vector 3

Adjust B23 to set the output power in its left output port to zero

Layer 3 settings

Shine in only Vector 1

Adjust B31 to set the output power in its left output port to zero

Shine in only Vector 2

Adjust B32 to set the output power in its right output port to zero

Layer 4 settings

Shine in only Vector 1

Adjust B41 to set the output power in its right output port to zero

Comment: this completes the setting if the “collection triangle” of blocks for Vector 1 to be routed to Output 1. We can now proceed to set the remaining blocks, one successive diagonal line at a time with one successive input Vector.

Comment: successive diagonal line settings

Shine in only Vector 2

Adjust B15 to set the output power in its right output port to zero

Adjust B24 to set the output power in its right output port to zero

Adjust B33 to set the output power in its right output port to zero

Adjust B42 to set the output power in its right output port to zero

Shine in only Vector 3

Adjust B16 to set the output power in its right output port to zero

Adjust B25 to set the output power in its right output port to zero

Adjust B34 to set the output power in its right output port to zero

Adjust B43 to set the output power in its right output port to zero

and so on for successive (shifted) input vectors and additional corresponding diagonal lines of blocks on the right in Fig. 4 of the main text.

If the blocks are physically calibrated, then we can simply replicate the settings of the first block in a block layer to all subsequent blocks in that layer.

References

1. D. A. B. Miller, "Why optics needs thickness," *Science* **379**, 41–45 (2023).
2. D. A. B. Miller, "Analyzing and generating multimode optical fields using self-configuring networks," *Optica* **7**, 794–801 (2020).