optica

Perfect optics with imperfect components: supplementary material

DAVID A. B. MILLER

Ginzton Laboratory, Spilker Building, Stanford University, 348 Via Pueblo Mall, Stanford, California 94305-4088, USA (dabm@ee.stanford.edu)

Published 18 August 2015

This document provides supplementary information to "Perfect optics with imperfect components," http://dx.doi.org/10.1364/optica.2.000747, including additional figures, formal statements of algorithms required to configure the devices in the approach, and example designs for various linear optical transforms and circuits that would be constructed automatically by the overall approach. © 2015 Optical Society of America

http://dx.doi.org/10.1364/optica.2.000747.s001

1. Alternate Mach-Zehnder block configurations

In the implementations of Mach-Zehnder interferometers (MZIs), in general two phase shifters are required somewhere so that both the split ratio and the phase of at least the "Right" output can be controlled. For the automatic algorithms, as long as both of these degrees of freedom exist, various configurations are possible for the MZI blocks. Additional possible configurations are shown in Fig. S1. The configuration (a) of Fig. S1 shows a conceptual implementation based on a conventional beamsplitter with a controllable reflector and a phase shifter with phase delay ϕ in the path from the "Right" port. The waveguide MZI version (b) of Fig. S1 is functionally equivalent to configuration (a). The split ratio of the MZI in version (b) – the ratio between the "Right" and "Bottom" output powers for a power incident on the "Top" port – is the equivalent of the reflectivity of the controllable reflector in version (a). In version (b), this is controlled by two phase shifters driven differentially, which give a controlled phase difference of θ between the two interferometer arms. The configuration (c) is particularly compact and symmetric, using differential drive of the phase shifters to control the split ratio and common mode drive to vary the output phase.

These different configurations differ functionally in that they can give rise to different phase shifts of the beam in the "Bottom" output port. For the automatic alignment algorithms, it is of no consequence whether the phase of that "Bottom" output is changed or not. That "Bottom" output is fed into the next "Channel row" of the linear network, and that next row is configured later in the algorithms; that next row can configure itself to take any phase of inputs to the "Top" of its MZIs from the "Bottom" of the preceding row. Though the automatic algorithm does not care about which implementation of the MZI blocks is used, of course the calculated values of the settings of the ϕ phase shifters will be different for different MZI block configurations. In the illustrative calculated designs below in Section 5, we presume the configurations of (a) and (b) of Fig. S1. It would be straightforward to repeat the calculations for other configurations, however.

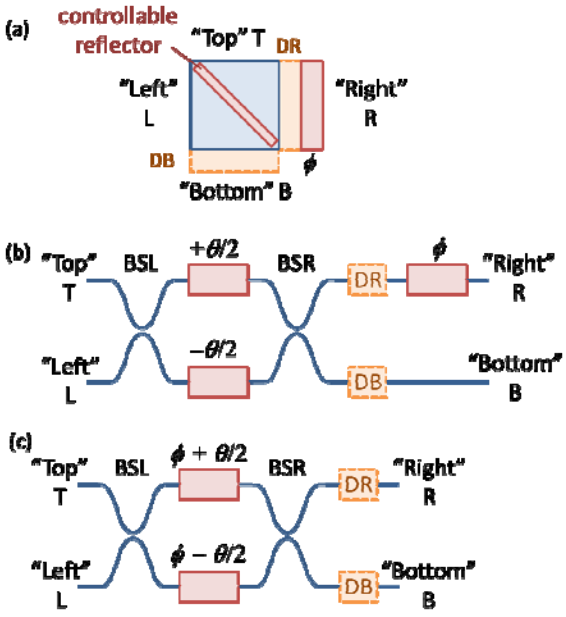

Fig. S1. Alternative configurations for MZI blocks.

2. Discussion of automatic beamsplitter adjustment

"No relative phase change" condition

In the main text, we presumed for simplicity a "no relative phase change" condition in our analysis, by which we mean that, if we shine a beam in the "top" port T in Fig. 2 (a) of the main text, the relative phase of the beams transmitted into arms C and D is not changed as we vary the magnitude of the reflectivity r_L of the left beamsplitter. That is the behavior that the Double Mach-Zehnder interferometer (DMZI) configuration (Fig. 2(b) of the main text) will show. This is not quite a necessary condition in general for a loss-less beamsplitter – an arbitrary phase shift could be added just before point C in Fig. 2(b) of the main text, for example, without violating unitarity. If there were such a relative phase change, in the algorithms below we would need to re-optimize the phase θ after each adjustment of r_L, though that step would be simple to add to the algorithms.

Graph of P_{Rmin}

The explicit graph of $P_{Rmin}(R_L, R_R)$, showing its minimum of zero along the line $\delta R_L = -\delta R_R$, is given in Fig. S2. This figure complements Fig. 3(a) of the main text.

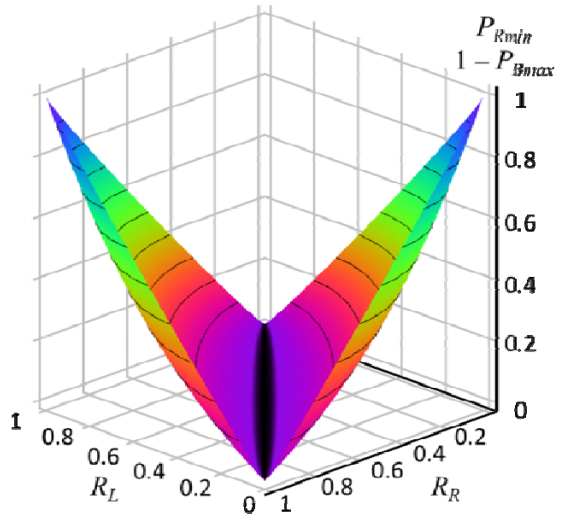

Fig. S2. Graph of $P_{Rmin}(R_L, R_R)$ (or, equivalently, $1 - P_{Bmax}(R_L, R_R)$).

3. Algorithm for setting the beamsplitter ratios in the mesh of MZI blocks

Before introducing the various algorithms formally, we can informally introduce the algorithm for setting up all the MZI blocks in a unitary mesh as in Fig. 1 of the main text so that all the blocks have their (effective) beamsplitters set to 50:50. We will call this the "mesh 50:50 setup algorithm" or MFSA for short. For the moment, we presume we have either the DR or DB (or possibly both) (mostly-transparent) detectors or power sampling points in the blocks, as shown Fig. 2 of the main text. For this setup, we presume we have sources that we can shine into inputs WI1 to WI3 in Fig. 1 of the main text, one by one. Here, we will describe the algorithm for these 3 input and 3 output unitary meshes, but the extensions to larger meshes are straightforward. We give formal general versions of all the algorithms below in section 4. BFSA below is the "beamsplitter 50:50 setup algorithm" as in the main text.

For this example case, then, the mesh 50:50 setup algorithm (MFSA) is as follows.

Shine power into WI1 only.

Run BFSA for B11. After completing BFSA, arrange that some power emerges from the lower ("bottom") port of B11 (e.g., by adjusting θ if necessary).

Run BFSA for B21, similarly leaving some power emerging from its "bottom" port.

Run BFSA for B31 (if needed – this block need only be a phase shifter)

Shine power into WI2 only.

Run BFSA for B12, leaving some power emerging from its "bottom" port.

Run BFSA for B22 (if needed – this block need only be a phase shifter)

Shine power into WI3 only.

Run BFSA for B13 (if needed – this block need only be a phase shifter)

This MFSA algorithm therefore allows us to set all the (effective) beamsplitters in all the MZI blocks to be 50:50.

Thus far, we have described how to set up the (effective) beamsplitters using detectors at one or both outputs of each MZI block. We already know it is possible to train the mesh for its ultimate function without detectors in the mesh (Appendix B of [1]). We can also run a slightly amended version of the MFSA algorithm without detectors inside each MZI block; specifically, we can set up the beamsplitter ratios in the entire mesh using only detectors D1 – D3, external to the mesh, on the outputs WC1 – WC3 respectively, effectively using them instead of the DR detectors inside the blocks. (Again, these need only sample a small amount of the power emerging from these waveguides.) We do this basically by working progressively first through all the MZI blocks on a given input row. A key point is that, after setting up the 50:50 ratios in all the MZI blocks in a row (e.g., B11 – B31) in this way, we add an algorithm, working backwards up through the row to set all those MZI blocks to the "bar" state – equivalent to perfect "reflection" from "top" to "right" and from "left" to "bottom"; effectively, this makes any such "bar" state MZI block appear as if it were not there at all. The details of this algorithm are given below in Section 4.

By this additional process of setting all the MZI blocks to the bar state, we leave the mesh in the starting state required for training the mesh for its ultimate function using only the external detectors D1 – D3 to run the "Self-configuring linear component algorithm" (SLCA) [1,2] for the final training of the mesh.

If we are using the mostly transparent detectors DB within the mesh, it is not necessary to set the MZI blocks to the bar state before training. Indeed, one major advantage of having the (mostly-transparent) DB detectors in the blocks is that we can be continually retraining the mesh if necessary as the problem changes, without having to reset all the blocks to the "bar" state before any such retraining; hence, having the DB detectors in the blocks allows a more incremental retraining to proceed all the time, as in tracking moving physical sources, for example.

With this informal introduction to the algorithms, in the next section we present the formal algorithms for configuring both the 50:50 split ratios and for training the ultimate function of the now-"perfect" set of MZIs.

4. General formal versions of the alignment algorithms

We presume there are M_C channel rows and M_I input rows; these numbers need not be equal in general for such processors (though Fig. 1 of the main text is drawn for $M_C = M_I$) – we might be mapping M_C orthogonal M_I-dimensional vectors to M_C single-mode output waveguides (or channels), for example, where $M_C < M_I$.

For these algorithms, we use the terminology as in Figs. 1 and 2 of the main text, though for greater clarity of notation we will write B(m, n) instead of Bmn in labelling the blocks, WI(n) and WC(m) instead of WIn and WCm respectively, and D(m) instead of Dm.

Several of these algorithms run over all the blocks B(m, n) in the mesh. For simplicity in stating the algorithms, such algorithms can be written to run over $n = 1$ to M_I and $m = 1$ to M_C. Because we only actually have a "triangle" of blocks at most in a given unitary mesh (see, e.g., Fig. 1 of the main text) , or fewer if $M_C < M_I$, we test to see if the block exists using an "If block B(m, n) exists" statement. Formally, a block exists if $m \leq M_I - n + 1$, which means it fits within the "triangle", and if $m \leq M_C$, which means it is in one of the channel rows actually implemented in the device.

There are two versions of many of these algorithms depending on whether (i) we embed mostly-transparent detectors or power sampling points inside the mesh or (ii) we use only detectors or power sampling points D(1) – D(M_C) at the channel "outputs" WC(1) – WC(M_C). We presume the MZIs are all loss-less or that they all have the same loss. If the MZIs have loss, then we should add dummy blocks as discussed at the end of this section, which means that all paths from input to output go through the same number of MZI blocks. In such a case, the output is then simply multiplied by a constant corresponding to the loss in M_I successive blocks, though the device otherwise performs the desired linear operations. The algorithms are given in pseudo-code here, with a syntax that is self-evident and similar to BASIC. Non-executable commenting statements are given in italics, starting with "*Comment:*".

These algorithms are only representative and are meant to indicate that there is at least one reasonable way of implementing all of these configurations and adjustments. There are variations possible, some of which are mentioned here, and alternative approaches that could be taken.

Algorithm S1 – Beamsplitter 50:50 setup algorithm (BFSA)

Note, as discussed by example above in Section 3 and explicitly below, that this algorithm will be run as part of Algorithm S2 (MFSA) below, which will ensure powering of appropriate optical inputs to run this BFSA algorithm.

For a given block B(m, n)

Set StoppingCondition to "False"

While StoppingCondition is "False"
Comment: if the StoppingCondition variable has the value "True", this statement will exit the loop without executing further statements

Set the phase shift θ to minimize the power at DR (if present) or at D(m) or to maximize the power at DB (if present)

Adjust δR_L and δR_R together in the same sense (ideally by equal amounts) (e.g., by adjusting θ_L and θ_R together in the same sense by approximately equal amounts) to minimize the power at DR (if present) or at D(m) or to maximize the power at DB (if present)

Set the phase shift θ to maximize the power at DR (if present) or at D(m) or to minimize the power at DB (if present)

Adjust δR_L and δR_R together but in the opposite sense (ideally by equal but opposite amounts) (e.g., by adjusting θ_L and θ_R together in the opposite sense by approximately equal amounts) to maximize the power at DR (if present) or at D(m) or to minimize the power at DB (if present)

If the appropriate stopping condition is met, set StoppingCondition to "True"
Comment: There are various different stopping conditions that could be set here. If using power minimization, we can test for the power at DB being below a chosen threshold. If using power maximization, at DR or D(m), we can test to see how close we are to the previous measured maximized value to see if we are below some chosen threshold of difference. Or, we can simply run the loop a specified number of times that we presume is enough for convergence, setting the StoppingCondition to "True" after a loop counter reaches that number

Loop *Comment: loop back to the "While" statement*

Algorithm S2 – Mesh 50:50 setup algorithm (MFSA)

Version with embedded detectors DR and/or DB in each block

For $n = 1$ to M_I

Shine power into WI(n) only

For $m = 1$ to M_C *Comment: it may not be necessary to run this for $m = M_I - n + 1$ since that block may just be a phase shifter*

If block B(m, n) exists, run BFSA for B(m, n) using DR and/or DB to detect minimum and maximum powers as required

Arrange that some (possibly all) power emerges from the lower ("bottom") port of B(m, n) by adjusting θ in block B(m, n) so some (possibly all) power is detected in DB in that block or so DR power is reduced at least somewhat from its maximum (possibly to zero).
Comment: this puts the block in a partial or complete "cross" state. This gives some power into the "top" port of the block in the next channel row (next m) so we will be able to run BFSA on it next.

Next m

Comment: this next part of the algorithm works back up through the line of blocks for a given n to set them all to the "bar" state. This is optional, since it is not required just to set up the 50:50 splits, but it is a more desirable and well-defined final state of the mesh for subsequent programming.

For $m = M_I - n + 1$ to 1 step -1 *Comment: it may not be necessary to run this for $m = M_I - n + 1$ since that block may just be a phase shifter*

If block B(m, n) exists, adjust θ in block B(m, n) so DB power is minimized or DR power is maximized Next m

Next n

Version without embedded detectors

For n = 1 to M_I

Shine power into WI(n) only

For m = 1 to M_C *Comment: it may not be necessary to run this for* $m = M_I - n + 1$ *since that block may just be a phase shifter*

If block B(m, n) exists, run BFSA for B(m, n) using D(n) to detect minimum and maximum powers as required

Arrange that some (possibly all) power emerges from the lower ("bottom") port of B(m, n) by adjusting θ in block B(m, n) so D(n) power is reduced at least somewhat from its maximum (possibly to zero) *Comment: this puts the block in a partial or complete "cross" state. This gives some power into the "top" port of the block in the next channel row (next m) so we will be able to run BFSA on it next.*

Next m

Comment: this next part of the algorithm works back up through the line of blocks to set them all to the "bar" state. Though this part of the corresponding algorithm above (for the case with embedded detectors) was optional, here it is needed so that input row of blocks we have just set now appears as if it is essentially not there as we set up the blocks in the next input row.

For $m = M_I - n + 1$ to 1 step -1 *Comment: it may not be necessary to run this for* $m = M_I - n + 1$ *since that block may just be a phase shifter*

If block B(m, n) exists, adjust θ in block B(m, n) so D($m - n + 1$) power is maximized

Next m

Next n

This version has both set up all the (effective) beamsplitters in the MZIs to 50:50 and set all the MZI blocks are in their bar states (i.e., the mesh is implementing an identity matrix).

Algorithm S3 – Self-configuring linear component algorithm (SLCA)

This algorithm is described in Ref. [1], and we give a formal version here for completeness. This algorithm sets up a unitary processor, e.g., as in Fig. 1(a) of the main text, so that the total power in each specific orthogonal input "training" (column) vector $|\psi_{Um}\rangle$ of (complex) field amplitudes at inputs WI(1) – WI(M_I) is mapped to the single channel output WC(m), for all m from 1 to M_C.

For this algorithm to work perfectly, the (effective) beamsplitters in the Mach-Zehnder blocks are all presumed to be 50:50, so we presume we have already run Algorithm S2 (MFSA).

Version with embedded detectors DB in each block

For m = 1 to M_C

Comment: if $M_C = M_I$, *we do not need to run this for the last case* $m = M_C$ *because the last channel row in that case is just a 100% "reflector" (bar state) block already. Mathematically, since all the columns in the matrix being implemented are orthogonal, we know the last column anyway because it has to be orthogonal to all the others we have already programmed into the mesh*

Illuminate all the inputs WI(1) – WI(n) with fields with (complex) amplitudes given by the corresponding M_I elements of the column vector $|\psi_{Um}\rangle$

Adjust θ in block B(m, $M_I - m + 1$) to minimize power at DB in that block *Comment: since this block is at the bottom of the set of blocks, it need only be a phase shifter, not a Mach-Zehnder interferometer, in which case this step is not needed*

For $n = M_I$ to 2 step -1

Comment: i.e., starting at $n = M_I$, *then decrementing n by* 1 *each time, and executing for the last time when* n = 2

If block B(m, $n - m$) exists

adjust ϕ in block B(m, $n - m + 1$) to minimize power at DB in block B(m, $n - m$)

adjust θ in block B(m, $n - m$) to minimize power at DB in block B(m, $n - m$)

End if

Comment: this should take the power to zero at DB *in block* B(m, $n - m$)

Next n

Next m

Version with no embedded detectors in the blocks

We presume we have already also run the version of Algorithm S2 (MFSA) without embedded detectors so we have also set all the MZI blocks to their "bar" state as the starting condition. If we are retraining the mesh to implement a new unitary operation, then we can run Algorithm S5, the "Bar-State Setup Algorithm" (BSSA) below before this retraining.

For m = 1 to M_C

Comment: if $M_C = M_I$, *we do not need to run this for the last case* $m = M_C$.

Illuminate all the inputs WI(1) – WI(n) with fields with (complex) amplitudes given by the corresponding M_I elements of the column vector $|\psi_{Um}\rangle$

Adjust θ in block B(m, $M_I - m + 1$) to maximize power in D($M_I - m + 1$) *Comment: since this block is at the bottom of the set of blocks, it need only be a phase shifter, not a Mach-Zehnder interferometer, in which case this step is not needed*

For $n = M_I$ to 2 step -1

Comment: i.e., starting at $n = M_I$, *then decrementing n by* 1 *each time, and executing for the last time when* n=2

If block B(m, $n - m$) exists

adjust θ in block B(m, $n - m$) somewhat away from its "bar" setting (e.g., by adding some control expected to give ~ 90° change in θ) *Comment: this step ensures that there is finite interference between the WI(n – m) and WI(n – m +1) input amplitudes at the outputs of block B(m, n – m) so that there is something to minimize as we adjust* ϕ *in the next step*

adjust ϕ in block B(m, $n - m + 1$) to minimize power in D($n - m + 1$)

adjust θ in block B(m, $n - m$) to minimize power in D($n - m + 1$) *Comment: this should take the power to zero* in D($n - m + 1$)

End if

Next n

Next m

Note: This algorithm is somewhat different from the algorithm in Appendix B of [1]. That algorithm works by maximizing output progressively in only one output detector for each input vector, on the presumption that all the MZIs are set initially to their "cross" state (i.e., "top" directly through to "bottom" and "left" directly through to "right", equivalent to the beamsplitter in Fig. 2(a) having no reflectivity at all). The algorithm here starts with the MZIs in their bar state, and works by successively minimizing power in various different output detectors. We could construct a version of Algorithm S2 (MFSA) that finished by setting all the MZIs to their cross states (or we could run Algorithm S4, the "Cross-State Setup Algorithm" (CSSA) below), and then use the previous algorithm in Appendix B of [1] here instead.

Algorithm S4 – Cross-State Setup Algorithm (CSSA)

This auxiliary algorithm can be used to set up all the blocks (except the lowest row of blocks, which are operating only as phase shifters and hence are always intended to be in their "bar" state if they even contain MZIs) in their "cross" state, where "top" is transmitted completely to "bottom" and "left" is transmitted completely to "right" in each block. This algorithm should be run only after setting all the internal (effective) beamsplitters in the blocks to 50:50, i.e., after running Algorithm S2 (MFSA). The blocks can otherwise be in any internal state, i.e., with any starting values of θ and ϕ in each block.

For n = 1 to M_I - 1

Comment: the upper limit of $n = M_I - 1$ is sufficient since the only block with $n = M_I$ is in the lowest row, and we do not want to set any of the blocks in the lowest row to the cross state since they are only operating as phase shifters.

Shine power into WI(n) only

For m = 1 to $M_I - n$

Comment: this choice of upper limit is so we do not attempt to put the lowest row of blocks (which may anyway just be phase shifters) into the cross state.

If block B(m, n) exists

Adjust θ in B(m, n) to maximize power in DB (if present) or maximize power in any of D(1) – D(M_C) in which power is present except for D(m)
Comment: Because we do not yet know the state of any blocks of larger m values, we have to consider that power from the "bottom" port of block B(m, n) could in some cases be routed to any of these other outputs. D(m) is the detector in which power will be present if power is coming out of the "right" port of block B(m, n) (which is power we are trying to minimize rather than maximize) so we do not want to maximize with respect to any power in D(m); note that this power is routed there because all of the blocks with smaller values of n have already been set to the "cross" state.

End if

Next m

Next n

We could also construct a version of this algorithm based on minimizing power in DR or in D(m).

Algorithm S5 – Bar-State Setup Algorithm (BSSA)

This auxiliary algorithm can be used to set up all the blocks (including the lowest row of blocks, which are operating only as phase shifters and hence are always intended to be in their "bar" state if they even contain MZIs) in their "bar" state, where "top" is transmitted completely to "right" and "left" is transmitted completely to "bottom" in each block. This algorithm should be run only after setting all the internal (effective) beamsplitters in the blocks to 50:50, i.e., after running Algorithm S2 (MFSA). The blocks can otherwise be in any internal state, i.e., with any starting values of θ and ϕ in each block.

Version with embedded detectors

For n = 1 to M_I

Shine power into WI(n) only

For m = 1 to M_C

If block B(m, n) exists

adjust θ in block B(m, n) so DB power is minimized or DR power is maximized

End if

Next m

Next n

Version without embedded detectors

Run Algorithm CSSA to set all blocks (except the bottom blocks, which may just be phase shifters) in their "cross" state.

For n = 1 to M_I

Shine power into WI(n) only

For $m = M_I - n + 1$ to 1 step -1

If block B(m, n) exists

adjust θ in block B(m, n) so D($m + n - 1$) power is maximized

End if

Comment: we are working back up each "Input" row of blocks, from largest m to smallest m in each row, changing them from "cross" to "bar" states

Next m

Next n

Note that the function of this algorithm is built into the version of Algorithm S2 (MFSA) for the case without embedded detectors, but is only optional in the version with embedded detectors.

Note on requirements on phase shift elements

All the algorithms presume that we have some way of changing and holding the values of the phases θ, ϕ, θ_L, and θ_R that we may adjust for each block while running the algorithms. We could be controlling the phases through heaters or through voltages on phase shift elements, in which case we would need (e.g., electronic) memories to hold the necessary drive voltages so as to retain these phases between algorithm steps and after we are finished with the algorithms. Alternatively, we might be physically trimming phase shifting elements by adding or subtracting material or physically permanently or semi-permanently changing refractive index.

To run the beamsplitter 50:50 setup algorithm, Algorithm S1 (BFSA), we have to be able to change θ by approximately π (i.e., 180°) multiple times, so it may be particularly useful to make that phase shifter voltage-controlled in some fashion, at least during the

initial training phase. Otherwise, if we only want a "set-it-and-forget-it" device that we configure only once, the phase shifts could be set by physical trimming. If we use forms of the MZIs with phase shifters in both upper and lower arms, we can increase (decrease) θ, θ_L, and θ_R by adding (subtracting) phase delay in the upper arm, and we can decrease (increase) them by adding (subtracting) phase delay in the lower arm; hence we can utilize trimming techniques that can only physically change phase delay in one sense (e.g., by adding material or by removing material). This could be particularly attractive for the θ_L, and θ_R phase shifts, which we would hope only to have to adjust once to compensate for fabrication tolerances. To adjust ϕ, however, we do need to be physically capable of both increasing and decreasing this phase. Also, having both the θ and ϕ phase shifters voltage-controlled allows the overall function of the device to be reprogrammed.

Use of dummy blocks to equalize loss

The mesh as shown explicitly in Fig. 1 of the main text has all the necessary blocks for arbitrary configuration of the network, but has different numbers of blocks in different possible "paths" through the network. For example, the path from WI1 to WC1 only passes through one block, whereas the path from WI3 to WC3 passes through three blocks. If we want to equalize background loss or overall delay on all paths, we can add dummy blocks[1,2], ultimately to be set in their "bar" state (i.e., complete transmission from "top" to "right" and from "left" to "bottom"). Such a configuration for a 4-input, 4-output mesh is shown in Fig. S3.

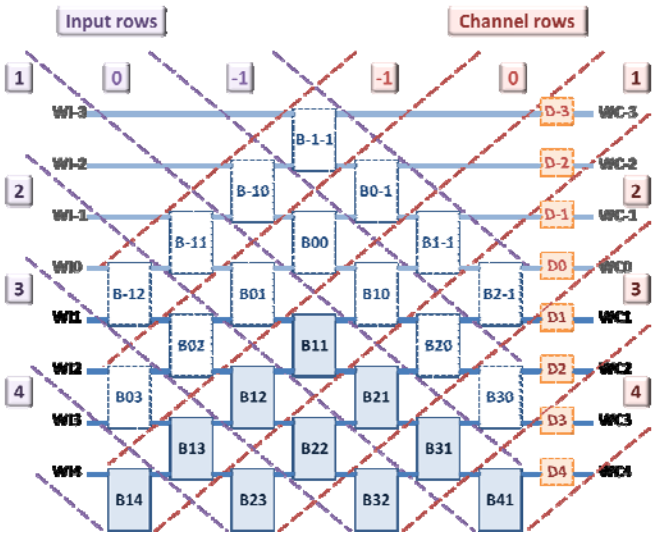

Fig. S3. 4-input, 4-output mesh with added blocks (with dashed outlines) so all paths from inputs WI1 – WI4 to outputs WC1 – WC4 through the network encounter the same number of blocks. Ultimately, all of the additional blocks here will be set to their "bar" state so they perform no mathematical function.

To achieve the equal numbers of blocks per path, we only need to add the dummy blocks that lie horizontally on the rows connected to the actual signal paths (i.e., on the horizontal lines from WI3 to WC3, from WI2 to WC2 and from WI1 to WC1 in Fig. S3), which in this example is the set of blocks B03, B02, B01, B10, B20, B30, B-12, and B2-1. To allow training of all of these additional dummy blocks so that their internal (effective) beamsplitters are 50:50 and so that they are all set to the "bar" state, we add the rest of the dummy blocks (here B-10, B-1-1, B0-1, B1-1, and B00). We can essentially then use all the same algorithms as we would use for the simpler mesh without the dummy blocks, merely extending Algorithm S2, the mesh 50:50 setup algorithm (MFSA), to also include these dummy blocks. The simplest way to view that extension is as if we had "completed the triangle" of blocks in Fig. S3, imagining that we extended the input rows and channel rows to the bottom of the figure. In the case of Fig. S3, this would involve hypothetically adding blocks B-13, B-14, B-15 to channel row -1, B04 to channel row 0, and similar hypothetical extensions for input rows 0 and -1. Then we imagine running MFSA for this larger mesh, just as before, though of course there are no actual settings to set in these hypothetical blocks. The additional inputs (e.g., WI0, WI-1, WI-2, and WI-3 in Fig. S3) and outputs (e.g., the waveguides WC0, WC-1, WC-2, and WC-3 and detectors D0, D-1, D-2, and D-3) are used during the MFSA algorithm (which should also be run to leave all the blocks in the "bar" state at the end). Otherwise, they are not used; the main functional training Algorithm S3 (SLCA) uses only the "active" inputs (WI1, WI2, WI3, and WI4 in Fig. S3) and outputs (WC1, WC2, WC3, and WC4 and detectors D1, D2, D3, and D4 in Fig. S3).

Note now that every pair of beams that interferes at a given block has passed through the same number of blocks. So, if there is equal loss in every block, the pair of beams has experienced the same attenuation. Hence, the settings the alignment algorithm gives to a particular block are the same as if there was no loss in the blocks. This in turn leads to a conservation of the orthogonality properties of the unitary transform even if the blocks are lossy. The resulting mesh of blocks will behave the same as the lossless set of blocks except for multiplication by a factor that corresponds to the loss in passing through one complete path from input to output. Of course, different loss in different blocks does not lead to this kind of behavior. This does mean, however, that losses like waveguide loss and beamsplitter loss do not matter to the function of the system, within an overall loss factor, as long as the waveguide loss and beamsplitter loss are sufficiently uniform for all waveguides and beamsplitters.

5. Example designs

Calculating device settings

The overall method we are proposing in this work does not require we calculate anything for the settings inside the mesh because it works based on feedback loops. To see what the method will actually do in a given design, we can calculate what the various device settings will be after training. This also enables us to compare with previous designs.

To do this calculation, we use the method in Appendix A of Ref. [2]. We give a simplified and condensed version of that method here in the current notation. For definiteness, we choose the structures as in Fig. S1 (a) and (b), which are equivalent. In these, changing the reflectivity (e.g., by changing θ) makes no change in the phases of either the "right" or "bottom" exiting beams, and the phase change ϕ only acts on the "right" exiting beam; these choices keep the mathematics slightly simpler. It is straightforward to map the results onto the alternative versions in Fig. 2 of the main text and in Fig. S1 (c). Also, we choose the phase delay for any "reflected" path (i.e., from "left" to "bottom", "bottom" to "left", from "top" to "right" or "right" to "top"), not including passing through any ϕ phase shifter, to be 0 (or some multiple of 2π radians), and

we choose the phase delay for any "transmitted" path (not including passing through any ϕ phase shifter) – i.e., from "left" to "right", "left" to "right", "top" to "bottom", or "bottom" to "top" – to be $\pi/2$ (possibly plus some multiple of π radians). (To satisfy unitarity the sum of these "top" to "bottom" and "left" to "right" phase shifts has to differ by π, within an additive multiple of 2π, from the sum of the "left" to "bottom" and "top" to "right" phase shifts.) With these choices, we can write the field reflectivity of the beamsplitter or MZI in block Bmn (not including passing through any ϕ phase shifter) as a positive real number r_{mn} and the field transmission "through" such a beamsplitter or MZI (not including passing through any ϕ phase shifter), from "left" to "right", "right" to "left", "top" to "bottom", or "bottom" to "top" as

$$t_{mn} = i\sqrt{1-r_{mn}^2} \tag{1}$$

where $i = \exp(i\pi/2) = \sqrt{-1}$ and we take the positive square root. We also can calculate θ in any block as

$$\theta_{mn} = \cos^{-1} r_{mn} \tag{2}$$

and we write the phase delay setting ϕ in any given block as ϕ_{mn}.

Mathematically, we imagine that we shine light backwards into the "output" channel waveguides WCm, starting with WC1. Progressively, we calculate the device settings so that the field emerging from the "input" ports is the phase conjugate (or Hermitian adjoint, $\langle\psi_{Um}|$) of the training vector $|\psi_{Um}\rangle$ (written as a vector of unit mathematical magnitude) we would have shone into the "input" ports. This is straightforward for the first channel "row".

We write the elements of the (column) vector $|\psi_{Um}\rangle = [a_{1m}, a_{2m}, \cdots, a_{M_I m}]^T$. Generally, we will write the vector of training amplitudes that would actually arrive at the "top" ports of the mth channel row of blocks as $|\psi_{Dm}\rangle = [d_{1m}, d_{2m}, \cdots, d(M_I - m + 1, m)]^T$. (We use the notation $d(p,q) \equiv d_{pq}$ wherever it is helpful for clarity, and we will similarly use the notation $B(m,n) \equiv Bmn$). For the first channel row, $|\psi_{D1}\rangle = |\psi_{U1}\rangle$. For all subsequent channel rows, $|\psi_{Dm}\rangle$ is different from $|\psi_{Um}\rangle$ because the training vectors must pass through all the earlier channels' rows of blocks; it is however, straightforward to calculate the $|\psi_{Dm}\rangle$ from the $|\psi_{Um}\rangle$ as we calculate progressively through the rows. Note, that with each successive row, the vector $|\psi_{Dm}\rangle$ becomes shorter by one element, just as the channel rows become shorter with increasing channel number in Fig. 1 of the main text.

The settings of the B11 block are trivial; in our current notation, we want $r_{11}\exp(i\phi_{11}) = d_{11}^* (= a_{11}^*)$, so $r_{11} = |d_{11}|$ and $\phi_{11} = \arg d_{11}^*$. (Here we take the "arg" function to give a result between -π and π. For the actual physical phase shifter, we can add a positive multiple of 2π to the calculated ϕ_{mn} to keep the phase shift physically positive if we wish.) As we move along the $m = 1$ channel row, the settings of the subsequent blocks have to account for the reflectivities and phase shifts of the previous blocks in the row. Generally, in a given row, with

$$\rho_{mn} = d^*(m, n-m+1)\exp\left(-i\sum_{p=1}^{n-m}\phi_{mp}\right)\Big/\prod_{p=1}^{n-m} t_{mp} \tag{3}$$

and with the understanding that, for $n = 1$, the summation term will be zero and the product term will be 1, then

$$r_{mn} = |\rho_{mn}| \quad \text{and} \quad \phi_{mn} = \arg\rho_{mn} \tag{4}$$

(Note that, if $\rho_{mn} = 0$, we can choose any phase setting; for definiteness, we can choose $\phi_{mn} = 0$ in that case.) In Eq. (3), the "sum" term gives the sum of all the phases of the phase shifters in the preceding blocks in the channel row m, and the "product" term is the product of all the (field) transmissions of the preceding blocks in the channel row m. Note that the last block in a given row, which will be block B(m, $M_I - m + 1$), will always have a reflectivity of 1; this will follow from the unitarity or losslessness of the optics and the normalization of the input training vectors, and in the calculations, we will explicitly always set this to 1. Note that if any $r_{mn} = 1$ (or $|\rho_{mn}| = 1$), we stop calculating any further values of ρ_{mn} on that channel row; those values will not matter because no light will get to those blocks, and $t_{mn} = 0$ for that block, which would lead to a "divide by zero" error in a calculation algorithm based on Eq. (3) for any further blocks on that channel row. For definiteness, we can set all the remaining ρ_{mn} (and r_{mn}) to zero (except for the last block B(m, $M_I - m + 1$) with its reflectivity of 1).

To complete the calculation method, we need to calculate each successive $|\psi_{Dm}\rangle$ from its corresponding training vector $|\psi_{Um}\rangle$ using the known values of the r_{mn} and ϕ_{mn} in the preceding channel rows. Generally, this mathematical transformation from the "input" into the "top" ports of the mth row to the "input" into the "top" ports of the (m + 1)th can be accomplished using an $(M_I - m)\times(M_I - m + 1)$ matrix $\mathsf{C}^{(m)}$, which we can write[2] as

$$\mathsf{C}^{(m)} = \begin{bmatrix} t_{m1} & c_{12}^{(m)} & c_{13}^{(m)} & \cdots & c_{1(M_I-m)}^{(m)} & c_{1(M_I-m+1)}^{(m)} \\ 0 & t_{m2} & c_{23}^{(m)} & & & \\ \vdots & 0 & t_{m3} & \ddots & \vdots & \vdots \\ & \vdots & \ddots & \ddots & & \\ 0 & 0 & 0 & \cdots & t_{m(M_I-m)} & c_{(M_I-m)(M_I-m+1)}^{(m)} \end{bmatrix} \tag{5}$$

where

$$c_{sj}^{(m)} = r_{mj} r_{ms}\left[\prod_{p=s+1}^{j-1} t_{mp}\right]\exp\left[i\sum_{p=s+1}^{j}\phi_{mp}\right] \tag{6}$$

is understood physically as the (complex) factor relating the input field on the "top" of block Bmj to its contribution to the input field on the "top" of block $B(m+1, s)$. Here, as we can see from the matrix, $j \geq s+1$. For the cases where $j = s+1$, the product term is understood to be 1 (there is no transmission term involved because the light reflects from one block into the next where it also reflects).

So, once we have calculated the settings in a given row m from its training vector $|\psi_{Um}\rangle$, we calculate the matrix $\mathsf{C}^{(m)}$. Then, from the next training vector $|\psi_{U(m+1)}\rangle$ we calculate the vector $|\psi_{D(m+1)}\rangle = \mathsf{C}^{(m)}\mathsf{C}^{(m-1)}\cdots\mathsf{C}^{(1)}|\psi_{U(m+1)}\rangle$, which we use to calculate all the $r_{(m+1)n}$ and $\phi_{(m+1)n}$ in channel row m+1, and so on.

In this way, we can calculate all the settings in the mesh for a given set of orthogonal, normalized training vectors, and hence for any desired unitary or loss-less transform between the inputs and the channel outputs.

Unitary linear operations

Three-way beamsplitter ("tritter")

A three-way beamsplitter or "tritter" splits each of its three inputs so that the input power in any one input is divided equally among its three output (see, e.g., Ref. [3] for a recent discussion). This is a good example to illustrate the method here; it is just complicated enough to be a non-trivial example, it has been demonstrated in real systems (e.g., [3]), and is of specific interest for quantum applications. We will work through this example in detail to clarify definitions, notations, and the overall process.

One unitary matrix D for such a device is[3]

$$\mathrm{D}_T = \frac{1}{\sqrt{3}}\begin{bmatrix} 1 & 1 & 1 \\ 1 & e^{i2\pi/3} & e^{i4\pi/3} \\ 1 & e^{i4\pi/3} & e^{i8\pi/3} \end{bmatrix} \tag{7}$$

Other matrices could also describe such a device[4]. Columns or rows in this matrix could be interchanged, which would correspond to a re-ordering of the inputs or outputs, respectively; any column could be multiplied by an arbitrary unit complex number, which corresponds to changing the phase delay from a given input to the whole set of outputs, and similarly any row could be multiplied by a unit complex number, which would change the relative phase of a given output compared to the other outputs.

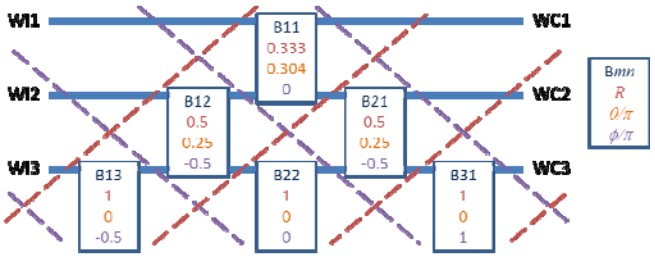

Fig. S4. Example calculated settings to implement a tritter. Settings for intensity reflectivity R, as set in a MZI with θ radians of phase lead in one arm compared to the other, and phase lead ϕ radians in the phase shifter (using configurations as in Fig. 2 (a) or (b)) are given for each block for a configuration as in Fig. 1 of the main text. (The numbers shown for phases are θ/π and ϕ/π, as indicated in the legend block on the right of the figure.)

The input mathematical vectors correspond to sets of field amplitudes in the input waveguides WI1 – WI3. Obviously, such a device takes the set of input vectors corresponding to illuminating the waveguides WI1 – WI3 one at a time,

$$|\psi_{I1}\rangle = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix},\ |\psi_{I2}\rangle = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix},\ \text{and}\ |\psi_{I3}\rangle = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{8}$$

and turns them, one by one, into the output vectors of sets of field amplitudes in the channel waveguides WC1 – WC3

$$|\psi_{O1}\rangle = \frac{1}{\sqrt{3}}\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix},\ |\psi_{O2}\rangle = \frac{1}{\sqrt{3}}\begin{bmatrix} 1 \\ e^{i2\pi/3} \\ e^{i4\pi/3} \end{bmatrix},\ |\psi_{O3}\rangle = \frac{1}{\sqrt{3}}\begin{bmatrix} 1 \\ e^{i4\pi/3} \\ e^{i8\pi/3} \end{bmatrix} \tag{9}$$

In our method, we train the FPLA to perform this function using input vectors that are the columns $|\psi_{Um}\rangle$ of the matrix $\mathrm{U} = \mathrm{D}_T^{\dagger}$ (or, equivalently, that are the Hermitian adjoints of the rows of D_T). Specifically, for D_T we train with the vector of input amplitudes

$$|\psi_{U1}\rangle = \frac{1}{\sqrt{3}}\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \text{ to give output vector } \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \mathrm{D}_T |\psi_{U1}\rangle \tag{10}$$

(Note: though we show normalized training vectors here, the actual power in the training vector makes no difference to the ultimate settings of the devices.) With this incident beam, we run Algorithm S3, the SLCA algorithm along the first channel row of blocks so all the power emerges from channel waveguide WC1 to give the desired output vector (within some overall phase delay for propagating through the device). Similarly, we train with

$$|\psi_{U2}\rangle = \frac{1}{\sqrt{3}}\begin{bmatrix} 1 \\ e^{-i3\pi/4} \\ e^{-i3\pi/2} \end{bmatrix} \text{ to give output vector } \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = \mathrm{D}_T |\psi_{U2}\rangle \tag{11}$$

Here we can use the convention that a positive phase ϕ (as in the factor $\exp(i\phi)$) corresponds to a phase lead and a negative one corresponds to a phase lag or delay. So, the training vector here has relative phase lags of $3\pi/4$ and $3\pi/2$ radians in the inputs to waveguides WI2 and WI3 respectively. Similarly, we train with

$$|\psi_{U3}\rangle = \frac{1}{\sqrt{3}}\begin{bmatrix} 1 \\ e^{-i4\pi/3} \\ e^{-i8\pi/3} \end{bmatrix} \text{ to give output vector } \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \mathrm{D}_T |\psi_{U3}\rangle \tag{12}$$

Though our method does not require any calculations to set it up, we can use the calculation method above to calculate what the settings of the phase shifters and reflectivities would be. The results are shown in Fig. S4.

Linear optics quantum computing CNOT gate

The linear optics quantum computing CNOT gate has been investigated extensively following its original proposal [5]. The version proposed in Ref. [6] has been demonstrated in discrete [7] and integrated [7,8] optical versions, including a generalized interferometer mesh with fixed couplings of two different kinds [8]. Using the relations between the input and output photon annihilation operators for the waveguide modes (Eq. (2) of Ref. [6]), we can directly write down the unitary device matrix for such a gate as

$$\mathrm{D}_G = \frac{1}{\sqrt{3}}\begin{bmatrix} -1 & \sqrt{2} & 0 & 0 & 0 & 0 \\ \sqrt{2} & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & -1 \\ 0 & 0 & 0 & 1 & -1 & -1 \end{bmatrix} \tag{13}$$

where the field variables for the input column vector, in the notation of Ref. [6], are in the order, from top to bottom, c_H, c_V, t_H, t_V,

v_c, and v_t, and similarly for the resulting output column vector, with variables in the order, from top to bottom, c_{H0}, c_{V0}, t_{H0}, t_{V0}, v_{c0}, and v_{t0}. For the physical mesh, the training vectors are the columns of the matrix $\mathsf{U}_G = \mathsf{D}_G^{\dagger}$.

Using this matrix and our algorithms here, we can automatically generate the design shown in Fig. S5 for a 6-input, 6-output mesh. The result is has both similarities to and differences from the layouts deduced in the original proposal [5]. (By "essentially identical" here we mean within phase shifts of various multiples of $\pi/2$ that can arise because of differences of definitions and choices of phases associated with beamsplitter operation.) First, the present mesh obviously contains more beamsplitter blocks than the original, quite economical and symmetrical design [5]; many of additional blocks are performing only simple "bar" or "cross" routing through this network, though (in some cases with $-\pi/2$ or $\pi/2$ phase shifts on the "upper" output arm).

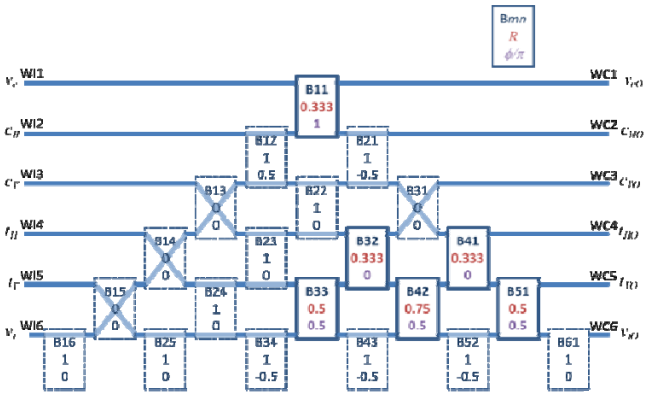

Fig. S5. Calculated settings of the mesh for the CNOT gate, showing the reflectivity R and the phase shift setting (phase lead) ϕ for each block. Blocks that in either the "bar" or "cross" state are shown with dashed outlines

The kind of mesh of Fig. S5 is capable of much more complicated operations with more fully populated matrices, which is why these additional blocks are present. Nonetheless, we can see that this automated design has separated out the 2x2 block between the v_c and c_H inputs and the v_{c0} and c_{H0} outputs and the 4x4 block connecting the other inputs and outputs. The arrangement for the 2x2 block is essentially identical to that of the original proposal [5]. The setup for handling the lower, 4x4 block is similar to the original proposal [5] but employs 5 rather than 4 beamsplitters. Note, though, that, unlike the original proposal, all the paths through this network pass through the same number of blocks, which equalizes loss (hence preserving orthogonality in the presence of finite but equal loss in the blocks) and the optical path length for all paths is nominally the same, which allows the device to function identically over some range of wavelengths. Furthermore, the present mesh approach can work just as well for any rearrangement of the inputs or outputs (equivalent to swapping rows or columns, respectively, in the matrix D_G), and is not restricted to the particular ordering of the original proposal [5].

We can "prefactor" the network into a 2x2 unit and a 4x4 unit, then our automated algorithm run on each of these separately, using the device matrix

$$\mathsf{D}_{G22} = \frac{1}{\sqrt{3}}\begin{bmatrix} -1 & \sqrt{2} \\ \sqrt{2} & 1 \end{bmatrix} \tag{14}$$

to construct the training vectors to shine into the top two waveguides, WI1 and WI2, and the matrix

$$\mathsf{D}_{G44} = \frac{1}{\sqrt{3}}\begin{bmatrix} -1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & -1 \\ 0 & 1 & -1 & -1 \end{bmatrix} \tag{15}$$

to construct the training vectors for the bottom four waveguides, WI3 – WI6. This process produces results shown in Fig. S6. We omit showing blocks that would be performing no function, i.e., operating in the "bar" (zero-reflection) state blocks with no additional phase shift. These additional blocks are not trained in this process, and should be preset in the bar state (by, for example, running Algorithm S5 (BSSA) before training.) The results now are essentially identical with the original proposal [5].

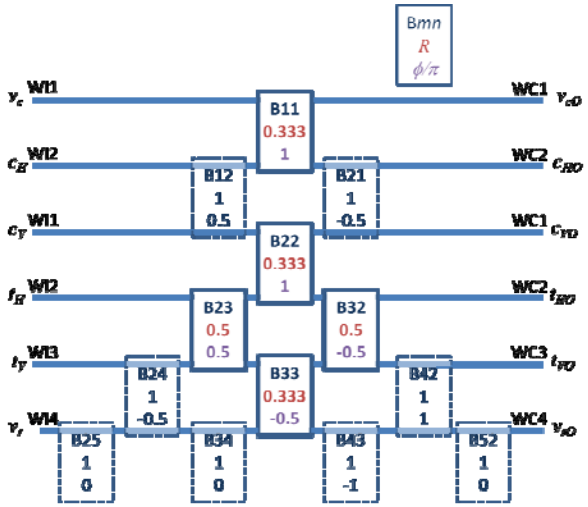

Fig. S6. Results of automatic design for the CNOT gate when run separately as a 2x2 unit (top two waveguides) and a 4x4 unit (bottom four waveguides).

Hadamard transform

The Hadamard transform is an example of a unitary linear transform with a broad range of applications in signal processing, encryption and data compression, and Hadamard transforms also have applications in quantum computing. The mth Hadamard transform H_m transforms an input vector of 2^m elements to an output vector of the same dimension. For example, the operator H_3 would be described by a device matrix

$$\mathsf{D}_{H3} \equiv \mathsf{H}_3 = \frac{1}{\sqrt{2^3}}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix} \tag{16}$$

The training vectors are the complex conjugates of the rows of D_{H3}; since these rows are real then the rows themselves are the training vectors, which are known as the Walsh functions. Note this matrix is symmetric (and Hermitian). The resulting calculated design is shown in Fig. S7. In this case, all the blocks (except the

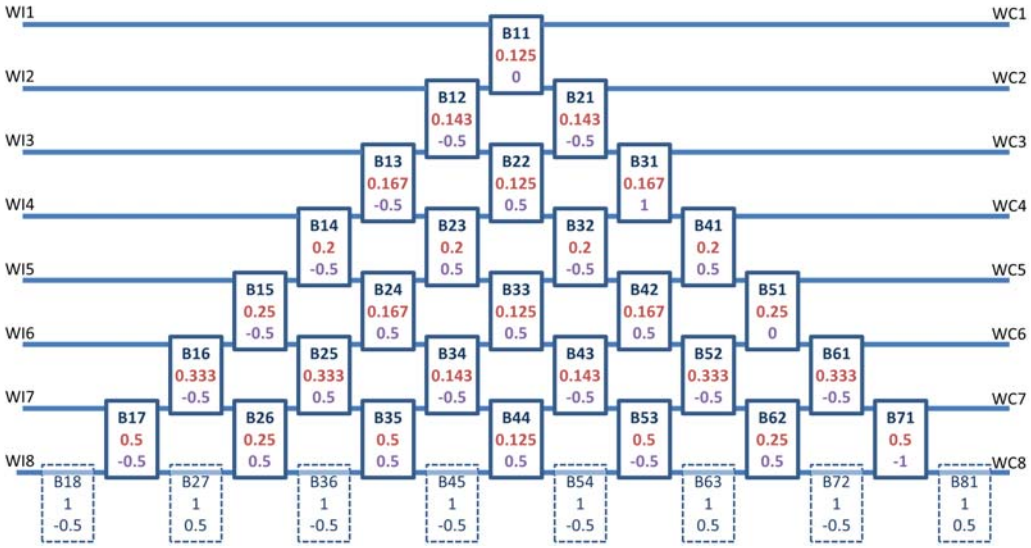

Fig.S7. Results of automatic design for the H_3 Hadamard transform, using the same notation as in Fig. S6

bottom row) have non-trivial reflectivities. The concept of making Hadamard transforms using sets of MZIs and phase shifters has been known in principle for some time [9]

Fourier transform

The concept of performing discrete Fourier transforms using combinations of Mach-Zehnder interferometers and phase shifts has also been known for some time [9]. An example form of discrete Fourier transform for a dimensionality $N=8$ is given by the matrix

$$\mathsf{D}_{F8}=\frac{1}{4}\times\begin{bmatrix} 1+i & \sqrt{2}\,i & -1+i & -\sqrt{2} & -1-i & -\sqrt{2}\,i & 1-i & \sqrt{2} \\ \sqrt{2}\,i & -\sqrt{2} & -\sqrt{2}\,i & \sqrt{2} & \sqrt{2}\,i & -\sqrt{2} & -\sqrt{2}\,i & \sqrt{2} \\ -1+i & -\sqrt{2}\,i & 1+i & -\sqrt{2} & 1-i & \sqrt{2}\,i & -1-i & \sqrt{2} \\ -\sqrt{2} & \sqrt{2} & -\sqrt{2} & \sqrt{2} & -\sqrt{2} & \sqrt{2} & -\sqrt{2} & \sqrt{2} \\ -1-i & \sqrt{2}\,i & 1-i & -\sqrt{2} & 1+i & -\sqrt{2}\,i & -1+i & \sqrt{2} \\ -\sqrt{2}\,i & -\sqrt{2} & \sqrt{2}\,i & \sqrt{2} & -\sqrt{2}\,i & -\sqrt{2} & \sqrt{2}\,i & \sqrt{2} \\ 1-i & -\sqrt{2}\,i & -1-i & -\sqrt{2} & -1+i & \sqrt{2}\,i & 1+i & \sqrt{2} \\ \sqrt{2} & \sqrt{2} & \sqrt{2} & \sqrt{2} & \sqrt{2} & \sqrt{2} & \sqrt{2} & \sqrt{2} \end{bmatrix} \tag{17}$$

where the matrix elements for such an *N*-dimensional transform are given by the expression

$$\mathsf{D}_{FNp,q}=\frac{1}{\sqrt{N}}\exp\left(2\pi ipq/N\right) \tag{18}$$

Using our FPLA architecture, the resulting calculated design would be as shown in Fig. S8.

Lens

With this FPLA approach, discrete Fourier transforms are not restricted to dimensionalities that are powers of 2 (as is the case for fast Fourier transforms, for example), and we can recenter them as desired. Another example of a Fourier transform is a device configured to emulate the action of a lens in forming a Fourier transform in its back focal plane of the field in its front focal plane. We specifically want a uniform input to result in a "spot" in the middle of the output. We can correspondingly center the transform by using the expression

$$\mathsf{D}_{LNp,q}=\frac{1}{\sqrt{N}}\exp\left[\frac{2\pi i}{N}\left(p-\frac{(N+1)}{2}\right)\left(q-\frac{(N+1)}{2}\right)\right] \tag{19}$$

Choosing *N* to be odd allows an output in the middle of the set of output waveguides for a uniform input. For $N=5$ the resulting calculated device matrix would be (to three significant figures)

$$\mathsf{D}_{L5}=\begin{bmatrix} a-bi & c+fi & g & c-fi & a+bi \\ c+fi & a+bi & g & a-bi & c-fi \\ g & g & g & g & g \\ c-fi & a-bi & g & a+bi & c+fi \\ a+bi & c-fi & g & c+fi & a-bi \end{bmatrix} \tag{20}$$

with $a\simeq 0.138$, $b\simeq 0.425$, $c\simeq -0.362$, $f\simeq 0.263$, and $g\simeq 0.447$. This leads to the calculated design as in Fig. S9.

Non-unitary linear operations

Many common and useful linear operations are non-unitary, and so in an FPLA architecture can be implemented using the configuration discussed in Ref. [2]. Such an approach formally decomposes the device matrix D by singular value decomposition into

$$\mathsf{D}=\mathsf{V}\mathsf{D}_{diag}\mathsf{U}^{\dagger}=\sum_{m=1}^{M}s_m\left|\psi_{Vm}\right\rangle\left\langle\psi_{Um}\right| \tag{21}$$

involving two unitary blocks $\mathsf{U}^{\dagger}$ and V respectively, separated by a set of modulators whose field transmissions implement the singular values s_m. Each unitary block is then trained separately with the appropriate training vectors. Now to find the two sets of training vectors, $\left|\psi_{Um}\right\rangle$ to train the $\mathsf{U}^{\dagger}$ block from the left and

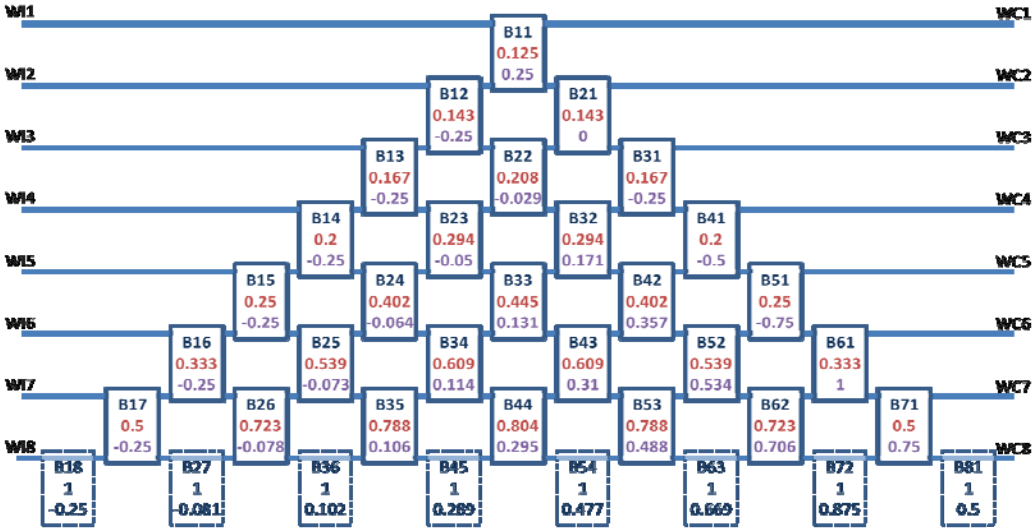

Fig. S8. Results of automatic design for the $N = 8$ discrete Fourier transform, using the same notation as in Fig.S6.

$|\psi_{Vm}\rangle$ to train the V block from the right, we can therefore formally perform the singular value decomposition as in Eq. (21) of the device matrix D. The vectors $|\psi_{Um}\rangle$ and $|\psi_{Vm}\rangle$ are, of course, just the columns of the resulting matrices U and V, respectively.

One minor mathematical point here relates to the choice of signs for the singular values. The eigen equations for $\mathrm{D}^\dagger\mathrm{D}$ and $\mathrm{D}\mathrm{D}^\dagger$ that we solve in singular value decomposition give us the numbers $|s_m|^2$ as eigenvalues, so the singular values themselves, s_m, are ambiguous within unit complex factors. The eigenvectors of $\mathrm{D}^\dagger\mathrm{D}$ and $\mathrm{D}\mathrm{D}^\dagger$ that we will calculate numerically when performing the singular value decomposition of D are themselves also ambiguous within unit complex factors and different, valid eigenvector algorithms might generate different resulting eigenvectors. In constructing the full singular value decomposition, we will have to add some process to make sure we have chosen these unit complex numbers consistently.

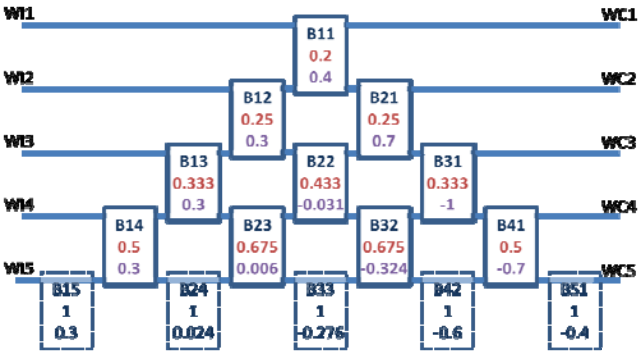

Fig. S9. Results of automatic design for the $N = 5$ discrete "lens" Fourier transform, using the same notation as in Fig. S6.

For simplicity, for example, we might choose all of the s_m to be real and positive, and we could leave the vectors $|\psi_{Um}\rangle$ (the columns of U) in whatever (normalized) form is generated by the eigenvector algorithm for the matrix $\mathrm{D}^\dagger\mathrm{D}$. Then we should construct the final versions of $|\psi_{Vm}\rangle$ (the columns of V) by multiplying the initial (normalized) results generated by the eigenvector algorithm for the matrix $\mathrm{D}\mathrm{D}^\dagger$ by whatever (unit) complex factor leads to the results $|\psi_{Vm}\rangle = s_m \mathrm{U}^\dagger |\psi_{Um}\rangle$. With this set of choices, the product $\mathrm{V}\mathrm{D}_{diag}\mathrm{U}^\dagger$ will correctly reconstruct the original device matrix D.

This is actually only a mathematical point because the absolute phase of the individual training vectors has no meaning physically. In the end with this approach for a physical device we are going to have to choose the phase relationship between an input training vector and its corresponding output vector manually, for example by setting the phase of the modulators in the middle, by some other process, if we care about the relative phases or signs of these different outputs.

Differentiation

One generally useful non-unitary problem is differentiation or, at least, its approximation by finite differences. We can work through a particular example here to show the method of tackling non-unitary problems with the FPLA architecture. We could imagine that the set of amplitudes at the input waveguides corresponds to a set of amplitudes of some function at a set of equally spaced points; then we could construct an approximation to the spatial first derivative of that function based on the differences in the function values at adjacent points (i.e., the amplitudes in adjacent waveguides). We can consider a system with 5 input waveguides, which will have a device matrix we could write as

$$\mathrm{D}_D = \frac{1}{\sqrt{8}} \begin{bmatrix} -1 & 1 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & -1 & 1 \end{bmatrix} \qquad (22)$$

Note, first, that this matrix is not square; there are only four differences between five adjacent values, so here we obtain a 4×5 matrix (and we will correspondingly only have 4 output waveguides). Note also that the rows of this matrix are not orthogonal. Hence, no matter what we choose for the prefactor in front of this matrix (here $1/\sqrt{8}$), D_D is not unitary (and it is already not technically unitary because it is not square). The choice of the prefactor here is to some extent arbitrary because it depends

on what we regard as the "distance" between adjacent points in our derivative. The choice of $1/\sqrt{8}$ here conveniently makes the sum of the squares of the matrix elements equal to 1, which also ensures that none of the singular values is greater than 1 (and hence can be implemented by a modulator element without gain), but other values could be chosen instead.

In the usual fashion for singular value decomposition, the columns of the matrix U are the eigenvectors of

$$\mathsf{D}_D^\dagger \mathsf{D}_D = \frac{1}{8}\begin{bmatrix} -1 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} -1 & 1 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & -1 & 1 \end{bmatrix}$$
$$= \frac{1}{8}\begin{bmatrix} 1 & -1 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & -1 & 1 \end{bmatrix} \quad (23)$$

Note, incidentally that, once we get away from the edges of the matrix, the rows of this matrix are taking on the character of the second difference or second derivative with the form "-1, 2, -1". The eigenfunctions of such a second derivative are, of course, just the various Fourier functions (sines, cosines, exponentials with imaginary exponents). Solving for the eigenvectors numerically (and normalizing them) therefore gives the matrix

$$\mathsf{U} = \begin{bmatrix} -0.602 & 0.512 & -0.372 & 0.195 \\ -0.372 & -0.195 & 0.602 & -0.512 \\ 0 & -0.632 & 0 & 0.632 \\ 0.372 & -0.195 & -0.602 & -0.512 \\ 0.602 & 0.512 & 0.372 & 0.195 \end{bmatrix} \quad (24)$$

The eigenvalues corresponding to the columns, from left to right, are $|s_1|^2 = 0.048$, $|s_2|^2 = 0.173$, $|s_3|^2 = 0.327$, and $|s_4|^2 = 0.452$. We choose the singular values as the positive real square roots, which become the diagonal values in

$$\mathsf{D}_{diag} = \begin{bmatrix} 0.219 & 0 & 0 & 0 \\ 0 & 0.416 & 0 & 0 \\ 0 & 0 & 0.572 & 0 \\ 0 & 0 & 0 & 0.672 \end{bmatrix} \quad (25)$$

There is technically a fifth column (with all elements equal) that we could add to U, which would make it square, but it corresponds to an eigen value of zero; as a result it will play no further part in the singular value decomposition, so we can drop it (and the corresponding column we would have had in D_{diag}). Without this column, the matrix U is technically not unitary, though this will not matter in practice for the subsequent mathematics. Similarly, we can find the eigen vectors of

$$\mathsf{D}_D \mathsf{D}_D^\dagger = \frac{1}{8}\begin{bmatrix} -1 & 1 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & -1 & 1 \end{bmatrix}\begin{bmatrix} -1 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (26)$$
$$= \frac{1}{8}\begin{bmatrix} 2 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 2 \end{bmatrix}$$

which leads to

$$\mathsf{V} = \begin{bmatrix} -0.372 & 0.602 & -0.602 & 0.372 \\ -0.602 & 0.372 & 0.372 & -0.602 \\ -0.602 & -0.372 & 0.372 & 0.602 \\ -0.372 & -0.602 & -0.602 & -0.372 \end{bmatrix} \quad (27)$$

where we have applied the procedure discussed above to choose the signs of the columns of V. (Note that the eigenvalues of $\mathsf{D}_D \mathsf{D}_D^\dagger$ are identical to those of $\mathsf{D}_D^\dagger \mathsf{D}_D$, as is always the case in this singular value decomposition procedure.) The reader can now verify that indeed the resulting product $\mathsf{V}\mathsf{D}_{diag}\mathsf{U}^\dagger$ constructs the original matrix D_D.

Hence we have found all the training vectors for setting up this non-unitary transformation in the FPLA architecture. Additionally, we have to manually set the modulators in the middle to implement the required singular value amplitudes, and we may have to additionally set a phase also in these to get the correct relative phases of the different input-vector-to-output-vector mappings (which is not set by the training procedures). Depending on the precise training vectors, such relative phases could be set, for example, based on shining two input training vectors on the device at once and adjusting the appropriate (singular value) modulator phase to maximize (minimize) the output power in a particular output port where the two corresponding output vectors are meant to be in (out of) phase, and repeating this for different pairs of beams until all appropriate relative phases have been set. This process will work for the present differentiation example since all the vectors are meant to be real.

Power splitters and integration

The self-aligning beam coupler previously described[1] can take a set of inputs of arbitrary amplitudes and phases and combine all the power to one output. This device corresponds to just one row of MZIs (for example, channel row 1 in Fig. 1 in the main text). If we train such a device with a vector of inputs of equal amplitudes and phases (here for $N = 3$ such inputs as a concrete example), then it will correspond to the following device matrix

$$\mathsf{D}_I = \frac{1}{\sqrt{3}}\begin{bmatrix} 1 & 1 & 1 \end{bmatrix} \quad (28)$$

This matrix is sufficiently simple that the singular value decomposition is trivial. The matrix $\mathsf{U}^\dagger = \mathsf{D}_I$, the one singular value $s_1 = 1$, and the matrix $\mathsf{V} = [1]$ (i.e., the trivial 1×1 unit matrix). The power reflectivities of the beamsplitter blocks B_{11}, B_{12}, and B_{13} will be 1/3, 1/2 and 1, respectively. This matrix is not unitary since it is not square. It will be lossless for this one specific

input of equal amplitudes with equal phases. Otherwise, it will add the (complex) field amplitude at each input to present the result at the output (i.e., in waveguide WC1). Such an operation corresponds to a discrete approximation to an integral (i.e., a summation).

Note that, while it adds the fields correctly, it does not in general add the powers in the different waveguides. For example, if the input had one unit of power in WI1, 2 units of power in WI2 (with both of these beams in phase), and no power in WI3, the corresponding input (field amplitude) vector would be $[1 \quad 4 \quad 0]^T$ (in units where power is the modulus squared of the field amplitude). The output would be $5/\sqrt{3}$ units of field in WC1, corresponding to a power of 25/9 units. Note, though, that the total input power was 3 units, which is greater than 25/9. The device has correctly added the input fields (with a scale factor of $1/\sqrt{3}$) but has not added the powers. (The remaining power will be dumped out of the lower right ports of blocks B11 and B12.)

A device such as this run backwards, with an input beam into WC1, will operate as an equal power splitter, with the "outputs" appearing at waveguides WI1 – WI3. The extension to larger number of waveguides WI*n* is straightforward, for both the field integration (summation) and the power splitting.

References

1. D. A. B. Miller, "Self-aligning universal beam coupler," Opt. Express **21**, 6360-6370 (2013)
2. D. A. B. Miller, "Self-configuring universal linear optical component," Photon. Res. **1**, 1-15 (2013).
3. N. Spagnolo, C. Vitelli, L. Aparo, P. Mataloni, F. Sciarrino, A. Crespi, R. Ramponi, and R. Osellame, "Three-photon bosonic coalescence in an integrated tritter," Nature Communications 4, 1606 (2013)
4. M. Zukowski, A. Zeilinger, and M. A. Horne, "Realizable higher-dimensional two-particle entanglements via multiport beam splitters," Phys. Rev. A 55, 2564-2579 (1997)
5. E. Knill, R. Laflamme, G. J. Milburn, "A scheme for efficient quantum computation with linear optics," Nature 409, 46 - 52 (2001)
6. T. C. Ralph, N. K. Langford, T. B. Bell, and A. G. White, "Linear optical controlled-NOT gate in the coincidence basis," Phys. Rev. A 65, 062324 (2002)
7. J. L. O'Brien, G. J. Pryde, A. G. White, T. C. Ralph and D. Branning, "Demonstration of an all-optical quantum controlled-NOT gate," Nature 426, 264-267 (2003)
8. J. Mower, N. C. Harris, G. R. Steinbrecher, Y. Lahini, and D. Englund, "An integrated programmable quantum photonic processor for linear optics," CLEO 2014, San Jose, CA, 8 – 13 June 2014, Paper FM2A.3
9. M. E. Marhic, "Discrete Fourier transforms by single-mode star networks," Opt. Lett. **12**, 63-65 (1987)