

VANDAL: A Tool for the Design Specification of Nanophotonic Networks

Gilbert Hendry*, Johnnie Chan*, Luca P. Carloni[‡] and Keren Bergman*

*Lightwave Research Lab, Department of Electrical Engineering, Columbia University, New York, NY

[‡]Computer Science Department, Columbia University, New York, NY

Abstract— Continuing to scale CMP performance at reasonable power budgets has forced chip designers to consider emerging silicon-photonics technologies as the primary means of on- and off-chip communication. Different designs for chip-scale photonic interconnects have been proposed, and system-level simulations have shown them to be far superior to purely electronic network solutions. However, specifying the exact geometries for all the photonic devices used in these networks is currently a time-consuming and difficult manual process. We present VANDAL, a layout tool which provides a user with semi-automatic assistance for placing silicon photonic devices, modifying their geometries, and routing waveguides for hierarchically building photonic networks. VANDAL also includes SCILL, a scripting language that can be used to automate photonic device place and route for repeatability, automation, verification, and scaling. We demonstrate some of the features and flexibility of the CAD environment with a case study, designing modulator and detector banks for integrated photonic links.

I. INTRODUCTION

As performance requirements continue to scale for scientific, embedded, and commercial high-performance computing, engineers are considering new technologies for solutions to some chip- and board-level challenges. Photonics is seen by many to be a superior communication medium, as it has been the case for the telecom industry, spawning research into nano-scale devices and integration [1]–[3]. The area of on-chip photonic interconnects has emerged as a solution to the problem of chip- and board-scale communication between processing cores and external memory.

Various photonic interconnection network designs have been proposed, including networks that use the wavelength domain for routing and arbitration [4]–[6], hybrid networks that use a lightweight electronic control plane for circuit-path setup [7], [8] and broadband photonic networks using distributed time division multiplexing (TDM) control [9]. All of these network designs rely on wavelength division multiplexing (WDM), which involves transmitting different wavelengths of light in the same guiding medium without significant interference between them by using ring resonators specifically tuned to certain wavelengths for modulation, filtering, and switching [10], [11].

However, ring resonator devices are highly sensitive to their geometry, namely the optical path length of the ring or racetrack and the gap between the resonator and the waveguides coupled to it. Designing networks which use ring resonators for WDM requires precisely specifying these

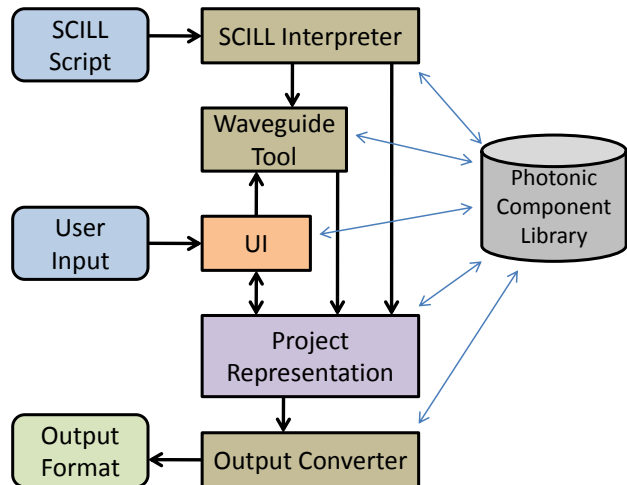


Fig. 1. VANDAL composition block diagram

geometries for hundreds, possibly thousands, of devices. This process requires tools which allow a designer to quickly and easily lay out and specify all the devices and components that make up a network. Furthermore, placing this process in an automation loop makes it possible to investigate network optimization for insertion loss, crosstalk, and power through changing spatial layout and device parameters, and simulating to measure the effects.

We present a tool for Visual Automated Nanophotonic Design And Layout, or VANDAL, which enables designers to visually place functional photonic devices, modify their parameters, route connecting waveguides between them, and hierarchically instantiate building blocks into complex networks. VANDAL also includes an interpreter for a language we also present here, Simple Component Intuitive Layout Language, or SCILL, which can be used to describe the process of placing, specifying, and routing photonic devices. Finally, VANDAL can be interfaced with a variety of other useful software tools, including industry-standard layout representation which can be used for fabrication, making VANDAL a tool central to different design and automation flows necessary to realize chip-scale photonic interconnects.

II. VANDAL: A PHOTONIC CAD TOOL

VANDAL is a fully functional photonic component place and route tool complete with GUI, written in C#. The organization of VANDAL is shown in Figure 1. The Photonic Component Library captures key knowledge of the operation

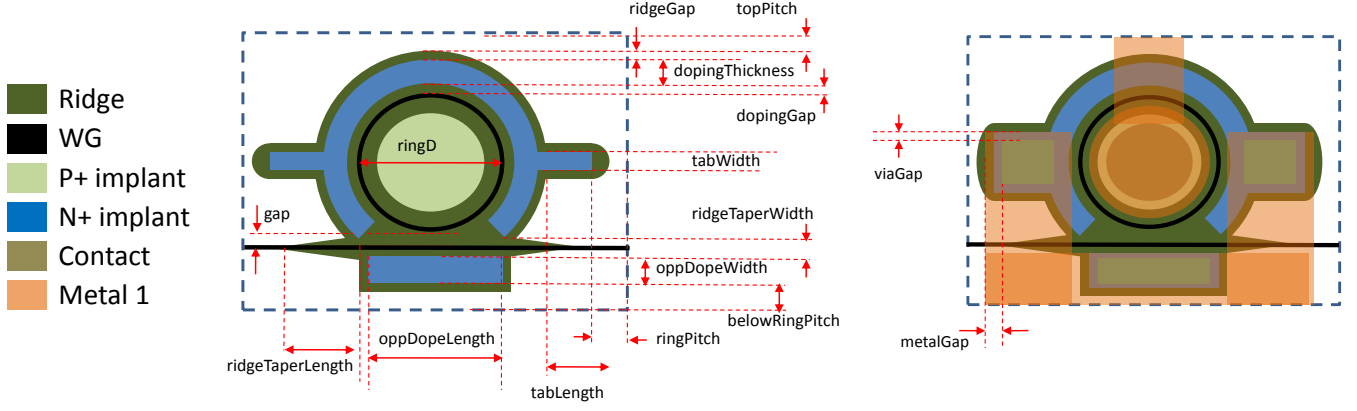


Fig. 2. Example of component parameterization - ring modulator

and geometries of various photonic devices (discussed in Section II-C), of which a user can instantiate via the graphical user interface or through SCILL (discussed in Section III). After a project consisting of devices and waveguides has been built, it can be exported to a variety of output formats used for different purposes (discussed in Section II-F). The remainder of this section describes the main components of VANDAL and the novel design methods that it supports.

A. Device Parameterization

Unlike traditional CMOS layout tools which allow a user to draw any geometry on any fabrication layer, we programmatically create functional building blocks which can be instantiated much like traditional hierarchical standard cells. We parameterize the layout of these components so that a user can control aspects of a component's behavior while retaining the geometries necessary for its correct operation. Figure 2 shows an example of the parameterization of a ring-modulator. Each geometry's position in the component can be specified using any of the parameters. For example, the y-position of the top edge of the waveguide ring relative to the top edge of the modulator is specified as $topPitch + ridgeGap + dopingThickness + dopingGap$.

Table I lists each parameter and its description for the ring-modulator example in Figure 2. Component parameterization enables significant advantages over free-form geometrical layout. First, for example, if we change the value of $ringD$, the modulator is automatically resized by maintaining the relative proportions of all other geometries. This allows a user to rapidly configure, derive, and analyze alternative implementations. This also applies to an automation tool which could instantiate many components, controlling each precisely. Finally, parameterization enables verification and repeatability, providing a medium for mapping post-fabrication device characterization to precise parameters.

B. Device Modeling

One significant novelty of VANDAL is that components can describe their photonic transfer functions based solely on their geometries and materials. This allows a designer to see the specific impact on device functionality when

TABLE I
RING MODULATOR PARAMETERS

Parameter	Description
ringD	Diameter of ring (middle of waveguide)
topPitch	Distance between top ridge and component's top edge.
ringPitch	Distance between N+ contact tab and component's right and left edges
belowRingPitch	Distance between N+ region opposite ring and component's bottom edge
ridgeGap	Gap between ridge edge and N+ region
dopingThickness	Width of N+ doping region around ring
dopingGap	Gap between edge of N+ region and ring
tabWidth	Width of N+ tab for contact
tabLength	Length of N+ tab for contact
ridgeTaperWidth	Width of ridge taper into exiting waveguide
ridgeTaperLength	Length of ridge taper into exiting waveguide
gap	Gap between ring and exiting waveguide
oppDopeWidth	Width of N+ region opposite ring
oppDopeLength	Length of N+ region opposite ring
viaGap	Gap from edge of N+ tab to edge of contact
metalGap	Gap from edge of contact to edge of metal1

changing different parameters, and enables high-level design automation flows. We incorporate models from PhoenixSim [12], a network simulator containing accurate physical-layer models, into VANDALs photonic components. For the case of a ring resonator, as the one illustrated in Figure 2, we capture the resonance characteristic as a function of transmission wavelength based on Yariv's equation [13]:

$$|b_1|^2 = \frac{\alpha^2 + t^2 - 2\alpha|t|\cos(\theta + \phi_t)}{1 + \alpha^2|t|^2 - 2\alpha|t|\cos(\theta + \phi_t)} \quad (1)$$

The optical power transmission coefficient, t , can be expressed from waveguide and gap dimensions from the Appendix in [14], thus providing a completely geometry and material dependent model. Capturing functional characteristics of photonic components also enables high-level design automation flows, such as reversing the modulator's equations or searching the parameter space for specific geometric dimensions that satisfy a given resonance profile.

C. Photonic Component Library

Parameterized models for the following silicon-photonics devices are available in VANDAL's component library:

a) *Modulator*: There are four types of parameterized modulators, each with different characteristics: single-order [10], second-order for hitless operation [15], PINIP for higher speeds [16], and single-order attached to a Mach-Zehnder interferometer for more athermal operation [17].

b) *Detector*: A single-wavelength filter/detector for compact layout based on a ring-resonator [18].

c) *Coupler*: A waveguide-to-waveguide hybrid mode coupler, as described in [19].

d) *Filter*: A ring-based wavelength filter with through and drop ports.

e) *1x2, 2x2 Switch*: A broadband ring-resonator switch, as described in [11].

f) *Taper*: For mode conversion, useful for coupling to off-chip fiber [20].

g) *Waveguide, bends*: The fundamental medium for guiding light, with 450nm default width for single-mode low insertion-loss propagation [21].

h) *Waveguide crossing*: Intersection of waveguides, engineered for low-loss [22].

Although many of the components currently defined are focused around the use of ring-resonator structures, VANDAL allows the addition of any type of parameterized device using multiple materials and layers.

D. Device Placement

VANDAL provides support for different methods of placing components in a plane. The first is through the GUI, by dragging and dropping instances of a component to its desired location. Components automatically snap to align with ports of other nearby components, and will auto-connect ports if dragged close enough.

The second method is by specifying the exact X-Y coordinates of the component. Coordinates are stored as unsigned positive integers, in nanometers. The third method is through the *connect* function, which aligns a component such that its indicated port is in the same location as another component's indicated port. These second and third methods are useful for automation or scripting processes, which will be discussed in more detail in Section III.

Finally, hierarchical instantiation is possible through *Compound* components which specify an existing project file, and *Port* components which specify the location of logical input and output ports of a layout.

E. Waveguide Tool

Once components are placed in a plane, we allow the user to connect components' logical ports together automatically with the Waveguide Tool (WGT). Given any two points, either unoccupied or occupied by a component's port (at the edge of that component), the WGT must find the lowest-loss path without violating the pitch requirements of any component.

In the current version of VANDAL, components are considered to be in the same silicon substrate because of typical waveguide-fabrication techniques [23]. Waveguide crossings are fabricated for low-loss and low crosstalk [22], but can still contribute a significant part of total network loss [24]. Nevertheless, the geometry and pitch of waveguide crossings must also be taken into account in the path-finding process.

We implement this process with A* (A-Star) search [25], a common graph-search technique. This method requires that any node must be able to generate a finite set of successor nodes, and that traveling from one node to another has a defined cost metric (in our case, insertion loss of the waveguide added from node n_1 to node n_2). Here, we define the search space as a graph of nodes which can represent any x-y coordinate. The basic process is as follows, for any starting node n :

- 1) Check finish condition - if n is at the destination, exit;
- 2) Check quick solution;
- 3) Generate n 's successors, adding them to the list of possible next moves;
- 4) Sort the list by n 's cost plus heuristic;
- 5) Repeat with next node, n' .

More often, the *quick solution* will be employed, which attempts a Manhattan-distance waveguide (containing 0 or 1 bend) from n directly to the destination. The quick solution fails if the previously-added waveguide is not in the correct orientation to perform a Manhattan-distance connection, or if there are any components blocking the path.

Successor nodes are generated by considering all possible cardinal directions that can be reached from n by either continuing straight for a given unit distance or turning with a 90-degree bend. One key element in A* search is the implementation of the *heuristic*, which attempts to estimate the remaining cost between n and the destination. For this we use the Manhattan distance, adding the cost of any waveguide crossings that must be placed along the path. The next node considered, n' , is the node that has the least cumulative insertion loss since the starting node (its total cost) plus the expected insertion loss to the destination (heuristic function).

Consider the example in Figure 3, which shows the process of connecting two North-facing port of adjacent components. The iterations go as follows:

- 1) The only valid successor (1-N) to the starting point at the port.
- 2) North, East, and West successors are generated (2-N, 2-E, 2-W). Since bends have slightly higher insertion loss than straight waveguides, the method minimizes the cost by continuing North-ward, even though the East-facing successor would be closer to the destination.
- 3) North, East, and West successors are generated (3-N, 3-E, 3-W). Now the straight North-ward waveguide has overcome the loss by bending East-ward in Step (2), so 2-E is the new current node.
- 4) The *quick solution* directly to the destination becomes valid, and the process is completed.

TABLE II
SCILL PRIMITIVE STATEMENTS

Statement	Description	Format/Example
Assignment	Modifies the namespace by either declaring a new variable or naming an existing one, and updating its value. <code><TYPE></code> is required when declaring a previously-unused variable name. <i>value</i> can be a constant, variable, or result of a function.	<code><TYPE> [name] = [value]</code>
New	Creates a new component of the type specified, and must be the name of a valid VANDAL component type (see Section II-C).	<code>COMP foo = new modulator</code>
If	The usual control statement, ended with <code>ENDIF</code> . <i>[condition]</i> can use the usual <code>==</code> , <code>>=</code> , <code><=</code> , and <code>!=</code> comparison operators.	<code>IF [condition]</code> ... <code>ENDIF</code>
For	The typical looping mechanism, simplified for integer-only, increment-by-1 operation. <i>var</i> is the loop variable (implied integer), which ranges from <i>min</i> to <i>max</i> , inclusive.	<code>FOR [var] in [min]..[max]</code> ... <code>ENDFOR</code>
Foreach	A mechanism for iterating over an <code>ARRAY</code> variable. End with <code>ENDFOR</code> .	<code>FOREACH <TYPE> [var] in [array]</code>

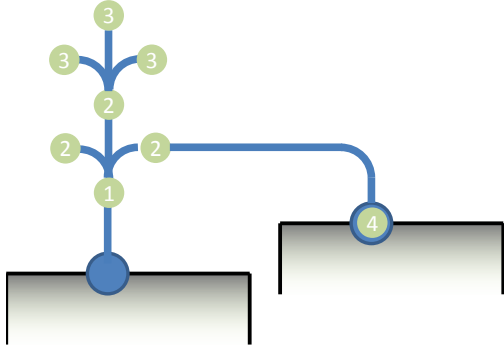


Fig. 3. Example connection two North-facing ports

In this way, our waveguide path-finding method can optimize the insertion loss between any two ports, though it does encounter some difficulty for layouts with large numbers of devices that it must route through. In these cases, the user has the option to manually route the waveguides for the most complex paths, thereby helping the tool achieve an optimized layout sooner.

F. Layout Output

VANDAL is capable of exporting its internal representation to a variety of file formats for different uses.

a) *VANDAL* - *.phc*: VANDAL's own file format, which consists of all component objects serialized using standard IO.

b) *NED*: The language used by OMNeT++ [26], and therefore PhoenixSim [12] to describe the instantiation and connectivity of simulation modules.

c) *CIF* - *Caltech Interchange Format*: A standard format used to describe simple geometries [27], which can be imported into fabrication machines and standard CMOS layout tools.

d) *Hotspot*: A thermal modeling tool, which takes simple geometry coordinates and size as inputs [28]. This tool is compatible with PhoenixSim representation, enabling simulation of thermal fluctuations on the photonic components.

e) *MEEP*: A tool for performing finite-difference time-domain (FDTD) electromagnetic simulations [29].

TABLE III
SCILL VARIABLE TYPES

Type	Description	Example
BOOL DOUBLE INT STRING	Standard C-style variable types	<code>INT foo = 10</code>
COMP	Represents a component in the layout being created. See <i>new</i> primitive. Attributes of the component can be accessed through <code>""</code> notation	<code>COMP myMod</code> <code>myMod.X = foo</code>
FILE	A file in the system.	<code>FILE fooFile</code>
DIR	A directory in the system. Use with <code>FOREACH</code> to iterate over all files in a directory	<code>DIR fooDir</code> <code>FOREACH FILE f</code> <code>in fooDir</code>
PORT	Logical port of a project, used in hierarchical placement	<code>PORT p =</code> <code>comp.port[0]</code>
ARRAY	Any variable can be declared as <code>ARRAY</code> , which stores multiple integer-indexed values	<code>ARRAY INT foos</code> <code>foos[0] = 10</code>

III. SCILL: A LAYOUT SCRIPTING LANGUAGE

We developed SCILL (Simple Component Intuitive Layout Language) to automate the layout of photonic components such that the process is precise, convenient, flexible, repeatable, and insertable into an automation loop for optimization. SCILL is a language with C-like sequential execution, modifying a *namespace* containing variable definitions and values. SCILL also contains script-like directives to use features of VANDAL, such as the `WGT`. Unlike other representations which simply describe the *locations* of every element in the system, a SCILL script can describe the *process* by which elements are placed. This allows us to implement powerful automation methods, such as instantiating many elements with little code using loops.

VANDAL includes an interpreter for SCILL, and automatically creates a new *project* for any SCILL script run. A SCILL script is typically structured as follows:

- 1) Include - specify any external scripts that are called from this one;
- 2) Parameters - specify user-level parameters using the *param* keyword that can be input at run-time through

VANDAL;

3) Execution - all other executable lines.

Example code follows in Section IV. See Table II for descriptions of the primary statements defined in SCILL. SCILL variables are typed, according to the descriptions found in Table III. SCILL also contains some useful built-in functions, as well as a mechanism for calling other scripts.

a) *moveTo*: This function is useful for specifying the position of a component relative to another component by moving the first such that one of its ports is connected to a port on the other. This function takes the form of `<moveTo [port1] [port2]>`, where *port1* and *port2* are PORT variables, and *port1* is a port on the component to be moved.

b) *connect*: Invokes the waveguide tool to connect two ports together with waveguides, in the form of `<connect [port1] [port2]>`, where *port1* and *port2* are PORT variables.

c) *rotate*: Simply rotates the given COMP variable 90 degrees clockwise. See Script 1 for an example.

d) *Other scripts*: SCILL code can call other SCILL scripts, by invoking the *include* keyword. The script can then be called like a function which processes user-provided input parameters to produce a returned value.

IV. CASE STUDY: LINK INSTANTIATION

We demonstrate some of the features of VANDAL through a simple case study that consists of instantiating some photonic gateways, verifying their resonance profile, and connecting them with waveguides. The network gateway is a key module of a nanophotonic network which consists of modulators and detectors to convert from the electronic domain to optical signaling. For ring resonator-based modulators and detectors, the gateway design is critical because the devices must be fabricated such that their resonant wavelengths are exactly in tune with any switches or filters in the network.

In this case study, we first write a SCILL script to automatically generate an arbitrary number of modulators and detectors, which can be found in Script 1. This script takes as parameters the number of modulators to instantiate, the minimum ring diameter, the change in diameter for adjacent rings, and whether to make modulators or detectors. We test our script by instantiating four modulators with minimum diameter of $5\mu m$ and a diameter change of $10nm$. The automatically generated resonance profile of the result can be seen in various screenshots in Figure 4, which also indicates important features such as free spectral range (FSR) and extinction ratio (ER).

Next, we write a script to use these gateways to instantiate photonic links, to connect cores or other IP across a chip. Script 2 shows the example we use here, making 2 links spanning about $0.12mm^2$ using six and four wavelengths. Figure 5 shows the resulting screenshot, with the modulator banks in the upper-left corner, and the detector banks at the bottom and on the right side. Besides the transmission profile of the links, we can also automatically calculate the insertion loss for any link, which is 0.34 and 0.48 dB for this example.

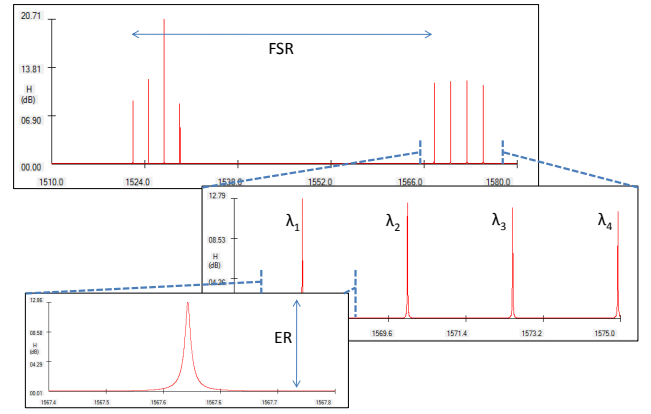


Fig. 4. Screenshots of response of four-modulator WDM gateway, with freespace wavelength on the x-axes and loss (dB) on the y-axes

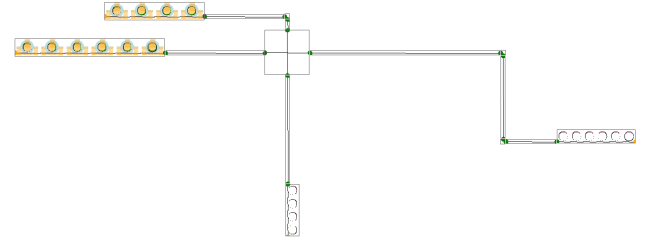


Fig. 5. Screenshot of two photonic links

V. CONCLUSION

In response to the growing interest in silicon photonic network design, we present VANDAL, a visual place and route tool, and SCILL, a simple domain-specific language, for the specification, design, and optimization of nanophotonic networks and their components. Users can write a SCILL script and process it with VANDAL to automatically generate a network by combining any number of devices (waveguides, couplers, modulator and detector banks, etc.). They can also analyze the network through the automatic derivation of its key characteristics from those of its components (such as gateway transmission profile and link insertion loss.) Future work includes advanced network-synthesis methods from higher levels of abstraction, the refinement of the component models through the characterization of fabricated components, and a complete demonstration of the design flow supported by VANDAL.

VI. ACKNOWLEDGMENTS

This work was partially supported by the Interconnect Focus Center (IFC), the NSF (Award #: 0811012), and the SRC under grant ECCS-0903406 SRC Task 2001

REFERENCES

- [1] L. Chen, K. Preston, S. Manipatruni, and M. Lipson, "Integrated GHz silicon photonic interconnect with micrometer-scale modulators and detectors," *Optics Express*, vol. 17, no. 17, August 2009.
- [2] B. G. Lee *et al.*, "All-optical comb switch for multiwavelength message routing in silicon photonic networks," *IEEE Photonics Technology Letters*, vol. 20, no. 10, pp. 767–769, May 2008.
- [3] A. Alduino *et al.*, "Demonstration of a high speed 4-channel integrated silicon photonics WDM link with hybrid silicon lasers," in *Proceedings of Hot Chips*, Aug 2010.

Script 1 SCILL code for modulator instantiation

```
param INT numW = 4
param DOUBLE D_min UNITS um = 10
param INT D_delta UNITS nm = 20
```

```
COMP c1
COMP prev
prev = new port
rotate prev
INT loopMax = numW - 1
INT temp
```

```
FOR i in 0..loopMax
IF doMods
c1 = new modulator
ELSE
c1 = new detector_int
ENDIF
c1.ringD = D_min * 1000
temp = i * D_delta
c1.ringD = c1.ringD + temp
moveTo c1.port[1] prev.port[0]
connect c1.port[1] prev.port[0]
prev = c1
ENDFOR
```

```
COMP out = new port
out.isIN = false
rotate out
rotate out
rotate out
moveTo out.port[0] prev.port[0]
connect out.port[0] prev.port[0]
```

Script 2 SCILL code for link instantiation

```
include makeGW
```

```
COMP link1mods = new makeGW 6 10 20 true
COMP link1dets = new makeGW 6 10 20 false
```

```
link1mods.X = 1000
link1mods.Y = 40000
link1dets.X = 600000
link1dets.Y = 140000
connect link1mods.port[1] link1dets.port[0]
```

```
COMP link2mods = new makeGW 4 10 20 true
COMP link2dets = new makeGW 4 10 20 false
```

```
link2mods.X = 100000
link2mods.Y = 0
rotate link2dets
link2dets.X = 300000
link2dets.Y = 200000
connect link2mods.port[1] link2dets.port[0]
```

- [4] D. Vantrease *et al.*, “Corona: System implications of emerging nanophotonic technology,” in *Proceedings of 35th International Symposium on Computer Architecture*, Aug 2008.
- [5] Y. Pan *et al.*, “Firefly: Illuminating future network-on-chip with nanophotonics,” in *Proceedings of the International Symposium on Computer Architecture*, 2009.
- [6] N. Kirman *et al.*, “Leveraging optical technology in future bus-

- based chip multiprocessors,” in *MICRO 39: Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture*. Washington, DC, USA: IEEE Computer Society, 2006, pp. 492–503.
- [7] A. Shacham, K. Bergman, and L. P. Carloni, “Photonic networks-on-chip for future generations of chip multiprocessors,” *IEEE Transactions on Computers*, vol. 57, no. 9, pp. 1246–1260, 2008.
- [8] M. Petracca, B. G. Lee, K. Bergman, and L. Carloni, “Design exploration of optical interconnection networks for chip multiprocessors,” in *16th IEEE Symposium on High Performance Interconnects*, Aug 2008.
- [9] G. Hendry *et al.*, “Silicon nanophotonic network-on-chip using TDM arbitration,” in *Proceedings of IEEE Symposium on High-Performance Interconnects*, Aug. 2010.
- [10] K. Preston, S. Manipatruni, A. Gondarenko, C. B. Poitras, and M. Lipson, “Deposited silicon high-speed integrated electro-optic modulator,” *Opt. Express*, vol. 17, no. 7, pp. 5118–5124, 2009.
- [11] B. G. Lee, A. Biberman, N. Sherwood-Droz, C. B. Poitras, M. Lipson, and K. Bergman, “High-speed 2×2 switch for multiwavelength silicon photonic networks-on-chip,” *Journal of Lightwave Technology*, vol. 27, no. 14, July 2009.
- [12] J. Chan *et al.*, “Phoenixsim: A simulator for physical-layer analysis of chip-scale photonic interconnection networks,” in *DATE: Design, Automation, and Test in Europe*, Mar. 2010.
- [13] A. Yariv, “Universal relations for coupling of optical power between microresonators and dielectric waveguides,” *Electronics Letters*, vol. 36, no. 4, pp. 321–322, Feb. 2000.
- [14] B. E. Little, S. T. Chu, H. A. Haus, J. Foresi, and J.-P. Laine, “Microring resonator channel dropping filters,” *Journal of Lightwave Technology*, vol. 15, no. 6, June 1997.
- [15] H. L. R. Lira, S. Manipatruni, and M. Lipson, “Broadband hitless silicon electro-optic switch for on-chip optical networks,” *Opt. Express*, vol. 17, no. 25, pp. 22 271–22 280, 2009.
- [16] S. Manipatruni, Q. Xu, and M. Lipson, “PINIP based high-speed high-extinction ratio micron-size silicon electrooptic modulator,” *Opt. Express*, vol. 15, no. 20, pp. 13 035–13 042, 2007.
- [17] B. Guha, B. B. C. Kyotoku, and M. Lipson, “CMOS-compatible athermal silicon microring resonators,” *Opt. Express*, vol. 18, no. 4, pp. 3487–3493, 2010.
- [18] J. K. Doyle, P. E. Jessop, and A. P. Knights, “Silicon photonic resonator-enhanced defect-mediated photodiode for sub-bandgap detection,” *Opt. Express*, vol. 18, no. 14, pp. 14 671–14 678, 2010.
- [19] T. Tamir and E. Garmire, *Integrated optics*. Springer-Verlag, Berlin ; New York :, 1975, ch. 8.
- [20] V. R. Almeida, R. R. Panepucci, and M. Lipson, “Nanotaper for compact mode conversion,” *Opt. Lett.*, vol. 28, no. 15, pp. 1302–1304, 2003.
- [21] J. Cardenas *et al.*, “Low loss etchless silicon photonic waveguides,” *Opt. Express*, vol. 17, no. 6, pp. 4752–4757, 2009.
- [22] W. Bogaerts, P. Dumon, D. V. Thourhout, and R. Baets, “Low-loss, low-cross-talk crossings for silicon-on-insulator nanophotonic waveguides,” *Opt. Lett.*, vol. 32, no. 19, pp. 2801–2803, 2007.
- [23] B. G. Lee *et al.*, “Ultra-high-bandwidth WDM signal integrity in silicon-on-insulator nanowire waveguides,” *IEEE Photonics Technology Letters*, vol. 20, no. 6, pp. 398–400, May 2007.
- [24] J. Chan, G. Hendry, A. Biberman, and K. Bergman, “Architectural exploration of chip-scale photonic interconnection network designs using physical-layer analysis,” *Lightwave Technology, Journal of*, vol. 28, no. 9, pp. 1305–1315, may. 2010.
- [25] P. Hart, N. Nilsson, and B. Raphael, “A formal basis for the heuristic determination of minimum cost paths,” *Systems Science and Cybernetics, IEEE Transactions on*, vol. 4, no. 2, pp. 100–107, July 1968.
- [26] A. Varga, “OMNeT++ discrete event simulation system,” online: <http://www.omnetpp.org>.
- [27] R. W. Hon and C. H. Sequin, *A Guide to LSI Implementation*. Xerox Palo Alto Research Center, 1980, ch. 7: A CIF Primer, pp. 79–123.
- [28] W. Huang *et al.*, “An improved block-based thermal model in hotspot 4.0 with granularity considerations,” in *Proceedings of the Workshop on Duplicating, Deconstructing, and Debunking*, 2007.
- [29] A. F. Oskooi *et al.*, “MEEP: A flexible free-software package for electromagnetic simulations by the FDTD method,” *Computer Physics Communications*, vol. 181, pp. 687–702, Jan 2010.