

# Electro-optical co-simulation for integrated CMOS photonic circuits with VerilogA

Cheryl Sorace-Agaskar,<sup>1,3,\*</sup> Jonathan Leu,<sup>1,3</sup> Michael R. Watts,<sup>1</sup> and Vladimir Stojanovic<sup>2</sup>

<sup>1</sup>Massachusetts Institute of Technology, Research Laboratory of Electronics, 50 Vassar St. Cambridge, MA, USA

<sup>2</sup>Electrical Engineering and Computer Sciences Department, University of California, Berkeley, CA, USA

<sup>3</sup>The first two authors contributed equally to this work.

\*ch20117@mit.edu

**Abstract:** We present a Cadence toolkit library written in VerilogA for simulation of electro-optical systems. We have identified and described a set of fundamental photonic components at the physical level such that characteristics of composite devices (e.g. ring modulators) are created organically – by simple instantiation of fundamental primitives. Both the amplitude and phase of optical signals as well as optical-electrical interactions are simulated. We show that the results match other simulations and analytic solutions that have previously been compared to theory for both simple devices, such as ring resonators, and more complicated devices and systems such as single-sideband modulators, WDM links and Pound Drever Hall Locking loops. We also illustrate the capability of such toolkit for co-simulation with electronic circuits, which is a key enabler of the electro-optic system development and verification.

©2015 Optical Society of America

**OCIS codes:** (130.0130) Integrated optics; (130.0250) Optoelectronics; (130.3990) Micro-optical devices; (130.4110) Modulators; (130.6750) Systems; (220.4830) Systems design; (250.0250) Optoelectronics.

---

## References and links

1. J. Orcutt and R. Ram, "Photonic device layout within the foundry cmos design environment," *IEEE Photonics Technol. Lett.* **22**(8), 544–546 (2010).
2. C. Sorace-Agaskar, Massachusetts Institute of Technology, Research Laboratory of Electronics, 50 Vassar St. Cambridge, MA, USA and M.R. Watts and collaborators are preparing a manuscript to be called "Integrated Silicon Photonic Single-Sideband Modulator."
3. A. F. Oskooi, D. Roundy, M. Ibanescu, P. Bermel, J. D. Joannopoulos, and S. G. Johnson, "MEEP: A flexible free-software package for electromagnetic simulations by the FDTD method," *Comp. Phys. Commun.* **181**(3), 687–702 (2010).
4. J. Morikuni and S.-M. Kang, "Optoelectronic simulation at the device and circuit level," in *Integrated Optoelectronics, Proceedings of IEEE/LEOS Summer Topical Meetings* (IEEE, 1994), pp. 8–9.
5. T. Smy and P. Gunupudi, "Robust simulation of opto-electronic system by alternating complex envelope representations," *IEEE Trans. Comp. Design* **31**(7), 1139–1143 (2012).
6. T. Smy, M. Freitas, and V. Ambalavanar, "Self-consistent opto-thermalelectronic simulation of micro-rings for photonic macrochip integration," in *Proceedings of IEEE Optical Interconnects Conference* (IEEE, 2012), pp. 68–69.
7. A. Lowery, O. Lenzmann, I. Koltchanov, R. Moosburger, R. Freund, A. Richter, S. Georgi, D. Breuer, and H. Hamster, "Multiple signal representation simulation of photonic devices, systems, and networks," *IEEE J. Sel. Top. Quantum Electron.* **6**(2), 282–296 (2000).
8. B. Wang, I. O'Connor, E. Drouard, and L. Labrak, "Bottom-up verification methodology for cmos photonic linear heterogeneous system," in *Proceedings of Forum on Specification Design Languages* (FDL, 2010), pp. 1–6.
9. A. Mekis, D. Kucharski, G. Masini, and T. Pinguet, "Computer-Aided Design for CMOS Photonics," in *Silicon Photonics for Telecommunications and Biomedicine*, 1st ed., S. Fathpour and B. Jalali eds. (CRC Press, 2011), pp. 383 –416.
10. Lumerical® INTERCONNECT circuit solver datasheet. "INTERCONNECT Circuit Solver," (Lumerical®, 2015). [https://www.lumerical.com/support/whitepaper/interconnect\\_circuit\\_solver/](https://www.lumerical.com/support/whitepaper/interconnect_circuit_solver/)

11. Photon Design® PICWave Datasheet, “PICWave: A photonic IC, laser diode and SOA simulator,” (Photon Design ®, 2015). <http://www.photond.com/products/picwave.htm>
12. P. Martin, F. Gays, E. Grellier, A. Myko, and S. Menezo, “Modeling of Silicon Photonics Devices with Verilog-A,” in *Proceedings of IEEE International Conference on Microelectronics* (IEEE, 2014).
13. K. Takeda, K. Honda, T. Takeya, K. Okazaki, T. Hiraki, T. Tsuchizawa, H. Nishi, R. Kou, H. Fukuda, M. Usui, H. Nosaka, T. Yamamoto, and K. Yamada, “Comprehensive photonics-electronics convergent simulation and its application to high-speed electronic circuit integration on a Si/Ge photonic chip,” Proc. SPIE **9388**, 938806 (2015).
14. D. Gallagher and T. Felici, “Modelling of silicon and active photonic integrated circuits,” Proc. SPIE **6996**, 69961E (2008).
15. VPICcomponentMaker Photonic Circuits Datasheet, “Design of Photonic Integrated Circuits” (VIP photonics, 2015). <http://www.vpiphotonics.com/Tools/PhotonicCircuits>
16. M. Fiers, T. van Vaerenbergh, K. Caluwaerts, D. Vande Ginste, B. Schrauwen, J. Dambre, and P. Bienstman, “Time-domain and frequency-domain modeling of nonlinear optical components at the circuit-level using a node-based approach,” J. Opt. Soc. Am. B **29**(5), 896–900 (2012) (More recent details can be found at: <http://www.intec.ugent.be/caphe/> and <http://www.lucedaphotonics.com/>).
17. E. D. Black, “An introduction to Pound–Drever–Hall laser frequency stabilization,” Am. J. Phys. **69**(1), 79–87 (2001).
18. K. Kundert, H. Chang, D. Jefferies, G. Lamant, E. Malavasi, and F. Sendig, “Design of mixed-signal systems-on-a-chip,” IEEE Trans. Comp. Design **19**, 1561–1571 (2000).
19. M. De Wilde, O. Rits, R. Bockstaele, J. M. Van Campenhout, and R. G. Baets, “Circuit-level simulation approach to analyze system-level behavior of VCSEL-based optical interconnects,” Proc. SPIE **4942**, 247–257 (2003).
20. K. Zhu, V. Saxena, and W. Kuang, “Compact Verilog-A modeling of silicon raveling-wave modulator for hybrid CMOS photonic circuit design,” in *Proceedings of IEEE MWSCAS* (IEEE, 2014), pp. 615–618.
21. B. Moss, “High-Speed Modulation of Resonant CMOS Photonic Modulators in Deep-Submicron Bulk-CMOS,” Thesis, Massachusetts Institute of Technology, September 2010.
22. J. Rhim, Y. Ban, B.-M. Yu, J.-M. Lee, and W.-Y. Choi, “Verilog-A behavioral model for resonance-modulated silicon micro-ring modulator,” Opt. Express **23**(7), 8762–8772 (2015).
23. G. Giustolisi, R. Mita, and G. Palumbo, “Behavioral modeling of statistical phenomena of single-photon avalanche diodes,” Int. J. Circuit Theory Appl. **40**(7), 661–679 (2012).
24. B. Blanco-Filgueira, P. López, and J. B. Roldán, “A Verilog-AMS photodiode model including lateral effects,” Microelectronics J. **43**(12), 980–984 (2012).
25. A. V. Oppenheim, R. W. Schafer, and J. R. Buck, *Discrete-time Signal Processing* (Prentice Hall, 1999). Ch. 2.
26. S. L. Marple, Jr., “Computing the discrete-time ‘analytic’ signal via FFT,” in *Proceedings of IEEE Asilomar Conference on Signals, Systems & Computers* (IEEE, 1997), pp. 1322–1325.
27. R. Amatya, “Optimization of tunable silicon compatible microring filters,” S.M thesis, Massachusetts Institute of Technology, February, 2008.
28. J. D. Jackson, *Classical Electrodynamics* (Wiley, 1998), Ch. 7.
29. D. J. Griffiths, *Introduction to Electrodynamics* (Prentice Hall, 1999), Ch. 9.
30. L. N. Binh, *Photonic Signal Processing: Techniques and Applications* (CRC Press, 2007), Ch. 2.
31. T. A. Ramadan, R. Scarmozzino, and R. M. Osgood, “Adiabatic couplers: design rules and optimization,” J. Lightwave Technol. **16**(2), 277–283 (1998).
32. C. T. DeRose, D. C. Trotter, W. A. Zortman, A. L. Starbuck, M. Fisher, M. R. Watts, and P. S. Davids, “Ultra compact 45 GHz CMOS compatible Germanium waveguide photodiode with low dark current,” Opt. Express **19**(25), 24897–24904 (2011).
33. S. M. Sze and K. K. Ng, *Physics of Semiconductor Devices*, (Wiley, 2006), Ch. 13.
34. E. Timurdogan, C. M. Sorace-Agaskar, J. Sun, E. Shah Hosseini, A. Biberman, and M. R. Watts, “An ultralow power athermal silicon modulator,” Nat. Commun. **5**, 4008 (2014).
35. M. Basic and M. Popovic, “Minimum drop-loss design of microphotonic microring-resonator channel add-drop filters,” in *Telecommunications Forum Proceedings* (IEEE, 2012), pp. 927–930.
36. S. Shimotsu, S. Oikawa, T. Saitou, N. Mitsugi, K. Kubodera, T. Kawanishi, and M. Izutsu, “Single side-band modulation performance of a linbo3 integrated modulator consisting of four-phase modulator waveguides,” IEEE Photonics Technol. Lett. **13**(4), 364–366 (2001).
37. M. Georgas, J. Leu, B. Moss, C. Sun, and V. Stojanovic, “Addressing link-level design tradeoffs for integrated photonic interconnects,” in *Proceedings of Custom Integrated Circuits Conference* (IEEE, 2011), pp. 1–8.
38. C. Batten, A. Joshi, J. Orcutt, A. Khilo, B. Moss, C. Holzwarth, M. Popovic, H. Li, H. Smith, J. Hoyt, F. Kartner, R. Ram, V. Stojanovic, and K. Asanovic, “Building Manycore Processor-to-DRAM Networks with Monolithic Silicon Photonics,” in *IEEE Symposium on High Performance Interconnects* (IEEE, 2008), pp. 21–30.

## 1. Introduction

As integrated electronic-photonic circuits and systems increase in size and complexity, the need for a scalable design and verification infrastructure becomes imminent. In particular, the ability to jointly simulate the electrical and optical parts of large systems together, including

noise and second-order effects, becomes critical. Substantial progress has been made towards a fully integrated design environment [1], but currently available simulation tools have yet to bridge the gap, in particular enabling scalable co-simulation of nonlinear, time-variant electronic-photonic devices and systems within the electronic design environment.

In particular, there is need for a joint system simulation code. Mature specialty software such as MEEP; Lumerical Device®, Mode solutions®, and FDTD solutions®; Fimmwave®; Sentaurus®; and custom home-built scripts exist and are widely used to simulate photonic components at the physical, device level [2,3]. These programs mirror programs like Sentaurus, Atlas, and HFSS that are used to model electronic devices like MOSFETs at the physical, device level. Such programs allow for the maximum amount of detail and flexibility and the most accurate results, but are too computationally intensive to simulate the behavior of hundreds of devices, especially over “long” timescales. To solve this issue, the electronics industry has moved to code where the results from such detailed physical simulations are translated, via careful assumptions and approximations, into simple governing equations for each device – a behavioral/compact model – and used in a separate system simulation in a different programming framework – often in VerilogA or Spice. This greatly reduces the level of complexity and number of input parameters, allowing for fast simulation time and an easy way for the system designer to change the key parameters. The photonics industry is also moving in this direction. Our goal is to supplement the CMOS electronic device models with photonic device models that can be used within the IC design environment allowing for co-simulation of integrated electronic-photonic circuits and systems within the same tool. Because it is the preeminent IC design suite, we have chosen to do this in Cadence, though our code will also be compatible with any other simulator capable of using VerilogA (for example, Mentor Graphic’s Questa® ADMS in Eldo®).

Previous work has attempted to provide such behavioral macromodels for photonic devices. Several simulation tools or individual models have been presented [4–16]. Only some of these [4–7,10,11,14–16] are able to properly capture transient behavior of all photonics components, handle frequency shifts and model both amplitude and phase effects (usually while removing the optical carrier for faster simulation time) and interference. Of this subset, all are written as standalone software and are therefore difficult to adapt into existing circuit design and layout infrastructures – especially with the transistor Spice or Spectre models provided in standard process design kits (PDKs). For example, OptiSpice [5,6] is an excellent tool that correctly captures transient effects as well as amplitude and phase effects, but is written as proprietary standalone software suite. This is also true of [11,14–16]. Lumerical’s INTERCONNECT® [10] is another excellent tool that is written as proprietary standalone software. INTERCONNECT® does provide hooks into Mentor’s Pyxis® platform, allowing for a joint electronics-photonic schematics to be laid out, but the simulator simulates the electronic and photonic parts separately – first doing one and then passing the output to the other – instead of doing a true joint simulation. Thus, making it impossible to accurately capture feedback loops that incorporate both electronic and photonic components, or systems with a tight interplay between the electronic and photonic regimes. Such feedback is, for example, necessary when a monitor photodiode is being used to stabilize the frequency of a ring in a digital link, or in a Pound-Drever-Hall locking loop [17] meant to stabilize the frequency of a laser. As the electronic-photonic systems grow in complexity it becomes imperative for the modeling and simulation framework to support the co-simulation of nonlinear, time-variant electronic-photonic device interactions. This work presents a modeling and simulation framework capable of capturing these effects in a co-simulation that is not written as a standalone software suite but instead written in VerilogA [18], a language already incorporated within the standard circuit design infrastructure. By uniting these abilities it thus fills an important gap.

There have been some attempts to write photonic models into VerilogA, but all have fallen short of the designer needs listed above [8]. provides a basic framework for simulation at

fixed wavelength, which is sufficient for simple links, but cannot capture the full behavior (especially phase phenomena, cross-talk, etc.) of more complex systems and devices like single-sideband modulators, coherent optics and wavelength division multiplexed waveguide links. Here, a higher fidelity representation of the underlying physics in each basic device is needed such that physical phenomena are organically generated simply from connecting basic components [19]. provides an overview of a similar approach [9]. presents an outline of a complete PDK similar to that created here, but the VerilogA models and infrastructure presented are overly simplistic for the type of models needed in many systems. For example, power and phase of the signals, are propagated around instead of real and imaginary parts. Working in power and phase creates convergence issues and prevents appropriate modeling of interference-based systems such as rings [5]. Indeed, no rings or similar systems are shown in that work. Models in [9] also do not appropriately account for the interaction of refractive and group index. The waveguide model simply outputs the input amplitude and phase delayed by a specified amount, but in reality the light output at the end of the waveguide will have an additional phase shift due to the waveguide refractive index. This effect is not captured and will lead to incorrect phases.

Probably most advanced among previous work is [13] which provides a nice set of VerilogA models for simulating a digital link, especially when it comes to underlying electrical structure for their photodiode. However, the transfer matrix approach to passives and rings is insufficient to appropriately capture non-linear or time-variant behavior of these components that is present in many active multi-device systems.

Finally, there has been some work on models for individual devices, including [20–24]. However, it is desirable not to need to calculate the transfer functions of even moderately complicated devices by hand and program them in — it would be much nicer if such things could be automatically generated during simulation. So while [22], provides an excellent model of a ring resonator it is not clear how useful it would be in allowing a system designer who knew little of optics to simulate a two-ring filter if need arose (for example, if lower cross talk between channels was needed) [22]. also assumes that the optical frequency is passed around meaning models written in this manner cannot fully capture frequency shifts or multi-frequency interaction without additional code.

As shown in the following sections, our toolkit identifies a set of fundamental photonic components and describes each at the physical level such that the characteristics of composite devices are created automatically. It correctly captures both phase and amplitude effects and forward and backward propagating light as well as electro-optical interactions and transient effects, and can be used to model a wide variety of optical-electrical systems. Written in VerilogA, it is easily used in a wide variety of simulation types in existing electronic circuit design software and easily combined with CMOS transistor models provided by a foundry to allow for true co-simulation of electronic-photonic circuits. The next section goes through the simulation architecture and theory, and explains how the toolkit works. In section three, we demonstrate several passive device simulations and show that the simulated results match analytic solutions. Section four does the same for active devices, in particular for a resonant modulator and a single sideband modulator. Finally, section five shows how the toolkit is used to simulate more complicated mixed-electronic-photonic systems, using as examples a digital WDM link, and a Pound-Drever-Hall laser stabilization loop – an example of a nonlinear time-variant electronic-photonic system that was impossible to simulate with any of the previously developed tools.

## 2. Simulation philosophy and theory

There are a number of traits that are important for a useful, flexible, simulation platform. The first and most basic, is that it needs to be a full simulation toolkit, not simply a few device models. Strategies are therefore needed to address both how best to write *specific models* and how best to construct overall system *architecture*. On an architecture level, it is desired that

photonic models (1) easily combine with electronic standard cells, (2) be usable in a variety of simulation types (DC, ac, transient, etc), (3) be fast, (4) capture phase effects, (5) support forward and backward propagation, (6) handle multiple frequencies and WDM channels, (7) capture appropriate electro-optic interactions, (8) be modular and flexible, (9) organically generate behavior of composite devices –that is model building blocks so that, when combined, simulation outputs correctly capture complex behavior of the composite device or system without needing explicit input from the user or the creation of a new block, and (10) generally be elegant and easy to use.

The first two goals on this list are automatically satisfied by the choice of VerilogA as the modeling language and the creation of the models in the Cadence framework. However, this choice adds a constraint of its own: (11) component models must exist only in the time domain, as VerilogA is a time domain modeling language. We next tackle desired trait number three: speed. Because optical frequencies are very large ( $\sim$ 200THz), a full Nyquist rate time domain simulation would take an extremely long time to run. However, in many applications, the bandwidth of interest is only on the order of  $\sim$ 100GHz to a few THz. To exploit the relatively small bandwidth of interest, the optical responses are shifted and calculated relative to a reference frequency chosen by the user. The choice of reference frequency is arbitrary and does not need to be related to any property of the system. However, placing the reference frequency near the center of the desired bandwidth allows for the largest time steps, improving the speed of the simulation the most. Capturing the amplitude and phase effects of all the components in this smaller band is generally sufficient, and in cases where a larger bandwidth is needed that wider bandwidth can still be accurately simulated but with a corresponding increase in simulation time.

To accomplish this shift in a simple manner, the simulation is carried out using the *analytic signal* (that is only the positive frequency components of the real signal are retained), as it allows such a shift to be made solely by multiplying the signal by  $\exp(j\omega_R t)$  where  $\omega_R$  is an arbitrarily chosen reference frequency. Working with the *analytic signal* also has the added benefit of making it easy to model phase effects, and satisfy desired trait number (4) - a phase or frequency shift is then just a multiplication by an exponential. However, it also means that the signal is no longer real. Because it usually works on the full real signal, and models phase changes by via time delays, VerilogA does not inherently support complex numbers. It is therefore necessary to add complex algebra into the program. We wrote a set of complex algebra functions as VerilogA modules for use inside our component blocks. A two signal bus is used to represent the real and the imaginary part. Real and imaginary parts are used as opposed to magnitude and phase to avoid the discontinuities in the phase signal when it wraps around  $2\pi$  and the lack of a clearly defined phase at zero amplitude that can cause fatal numerical convergence issues when signals are added together during the modeling of optical inference effects. (satisfying requirement (4)) [5]. Expanding the representation into a four signal bus, two signals for the real and imaginary parts of the forward propagating signal and two for the backward propagating signal, allows us to satisfy requirement (5) and to capture behavior and errors caused by reflections as well as effects of having forward and backward propagation on the same waveguide if desired. To avoid passing the optical signals on these buses around as electrical voltages, an optical *discipline* is added to the *discipline* file.

The above approach to modeling signals automatically allows for multiple optical frequencies or spreads of frequencies to be modeled *without* any need to make explicit reference to these frequencies (desired trait (6)). Thus the optical signal can propagate through the system, including through frequency-selective components such as ring filters or frequency-changing elements such as single-sideband modulators, without any explicit reference made to the frequency of the light or the frequency transfer function of these elements. All the appropriate interactions will be generated naturally from the math in the component models. Finally, since VerilogA was developed for use in the electronics industry,

adding electrical structures to optical components to model electro-optic effects (and satisfy desired trait (7)) is straightforward. The details of these implementations are described below.

The next item on our list of desired traits is for the code to be modular and flexible. While the architecture above helps with this, accurately modeling the optical components is also critical. In order to allow for maximum flexibility, applicability, and adaptability, optical devices are broken down into the most basic possible components (waveguides, couplers, phase-shifters, photodetectors, heaters, light sources, etc.), which, in turn, are modeled using simple equations that capture the important physics of the devices. This is similar to how models of diodes or transistors in the electrical domain rely on basic physical equations. The use of physical equations to capture photonic behavior allows for VerilogA blocks to be easily modified to add complexity and higher order effects. The blocks include both the optical responses, and, where appropriate, the electrical responses of these base components. More complicated structures, such as ring resonators, filters, and modulators are built out of these components, thus avoiding the need to specifically calculate a transfer function and create a model for each of these structures.

Some complex devices, such as modulators and ring resonators, may be used repeatedly within the same system. It is desirable not to have to recreate these small systems from scratch every time they appear. For combinations of components that are commonly used, schematic symbols can be created once and reused as black boxes wherever needed in a larger system. These symbols can have tunable parameters, so that the work of designing a single component can be leveraged to allow the simple use of a whole class of components in a design. Each symbol is still internally built out of our basic physical blocks – thus adaptability and flexibility is maintained. Such symbols can also be useful for holding more complicated components that might not be repeated often so that the top level design looks clearer. If necessary, the designer can drill down the hierarchy of abstract symbols to view and modify their design. Figure 1 illustrates the creation and inside of such a schematic symbol for a ring resonator.

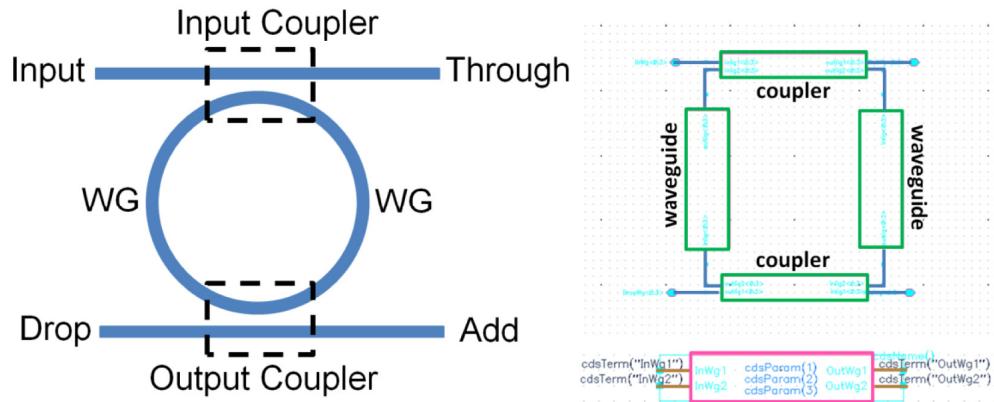


Fig. 1. A ring resonator (left) is broken into components. The VerilogA modules for these components are laid out and connected in a schematic in Cadence (right-top). A schematic symbol is created to represent this schematic block (right-bottom). This symbol can be inserted wherever a ring is needed as opposed to having to redraw the middle block. Relevant variables are passed through this symbol to the underlying schematic.

The most basic building block of optical systems is the optical waveguide. In order to represent this component in VerilogA, it is necessary to develop a time domain representation of the waveguide transfer function. As mentioned above, the physical electric field is real, but many operations are difficult to do on the full real signal. To get around this, one can take advantage of the fact that real signals are symmetric around zero in the frequency domain. We can therefore ignore all the negative frequencies without loss of information. An inverse

Fourier transform can then be done to get back to the time domain, where the result will be a complex signal. In the signal processing literature, this is called the *analytic signal* (see [25,26]) and is formally defined in Eq. (2) in [27] as:

$$Z(\omega) = \begin{cases} 2X(\omega) & \text{for } \omega > 0 \\ X(\omega) & \text{for } \omega = 0 \\ 0 & \text{for } \omega < 0 \end{cases} \quad (1)$$

where  $z(t)$  is the *analytic signal* of  $x(t)$ . The use of the *analytic signal* is often simply assumed implicitly in the electromagnetic literature (see, for example [28,29]). At the end, to recover the real, observable signal one can simply take the real part. Because our component code is very modular, we do not know where the end is until we hit some sort of detector. Therefore, the sampled, complex, analytic signal is passed through all the components. Note that, because the light source models output the time-domain *analytic signal* representation of the laser output to start with, there are no Fourier transforms in the code, nor is there any explicit tracking or specification of light frequencies; all the components are operated in the time domain as required by the VerilogA infrastructure simulator.

The *analytic* representation of the electric field at the start of the waveguide ( $z = 0$ ) is  $\tilde{E}(0, t)$ . The wave equation, derived from Maxwell's equations, says that at a location,  $z$ , in the waveguide the field is equal to:

$$\tilde{E}(z, t) = \int_{-\infty}^{\infty} \tilde{E}(0, \omega) e^{-j\beta z + j\omega t} d\omega = \int_0^{\infty} \tilde{E}(0, \omega) e^{-j\beta z + j\omega t} d\omega \quad (2)$$

where  $\beta = \beta(\omega)$  is the wavevector in the waveguide and the second equality follows because there are no negative frequency components in our *analytic* representation. In order to avoid simulation at the optical time step, a reference frequency  $\omega_R$  is chosen such that  $\omega = \omega_R + \Delta\omega$  and the signal is shifted down by this frequency. To do this,  $\beta(\omega)$  is first Taylor expanded around this frequency yielding:

$$\beta(\omega) = \beta(\omega_R) + \Delta\omega \frac{\partial\beta}{\partial\omega} \Big|_{\omega_R} \quad (3)$$

Note that,  $\frac{\partial\beta}{\partial\omega} \Big|_{\omega_R} = \frac{1}{v_g}$ , where  $v_g$  is the group velocity of light in the waveguide at the reference frequency and is equal to  $c/n_g$ , where  $n_g$  is the group index. This brings us to:

$$\tilde{E}(z, t) = \int_0^{\infty} \tilde{E}(0, \omega_R + \Delta\omega) e^{-j\beta(\omega_R)z - j\frac{\Delta\omega}{v_g}z + j(\omega_R + \Delta\omega)t} d\omega \quad (4)$$

We further note that we can define a function  $\tilde{E}_{\text{shift}}(z, \Delta\omega) = \tilde{E}(z, \omega_R + \Delta\omega)$  which re-labels the frequency to shift the amplitude coefficients by the reference frequency. While the reference frequency can be thought of as a carrier frequency and  $\tilde{E}_{\text{shift}}(z, \Delta\omega)$  as an envelope function, it is important to remember that the reference frequency is not the center (or necessarily any) frequency of the propagating light and that  $\tilde{E}_{\text{shift}}(z, \Delta\omega)$  is complex (it is not amplitude modulation) and holds for an arbitrarily (or infinitely) large frequency range. It is the Taylor expansion of  $\beta$  and simulation time constraints that will limit the workable frequency range. Using this function, the variable of integration is changed to  $\Delta\omega$ :

$$\begin{aligned}\tilde{E}(z, t) &= e^{-j\beta(\omega_R)z + j\omega_R t} \int_{-\omega_R}^{\infty} \tilde{E}_{\text{shift}}(0, \Delta\omega) e^{-j\frac{\Delta\omega}{v_g}z + j\Delta\omega t} d\Delta\omega \\ &= e^{-j\beta(\omega_R)z + j\omega_R t} \int_{-\infty}^{\infty} \tilde{E}_{\text{shift}}(0, \Delta\omega) e^{-j\frac{\Delta\omega}{v_g}z + j\Delta\omega t} d\Delta\omega\end{aligned}\quad (5)$$

where, again, the second equality follows from the fact that there are no frequency components between  $-\omega_R$  and  $-\infty$ . As can be seen, this integral is just the Fourier transform of  $\tilde{E}_{\text{shift}}(z, t)$ . Thus:

$$\tilde{E}(z, t) = e^{-j\beta(\omega_R)z + j\omega_R t} \tilde{E}_{\text{shift}}(0, t - \frac{z}{v_g}) \quad (6)$$

The field at a point,  $z$ , in the waveguide can also be expressed as:

$$\tilde{E}(z, t) = \int_{-\infty}^{\infty} \tilde{E}(z, \omega) e^{j\omega t} d\omega = \int_0^{\infty} \tilde{E}(z, \omega) e^{j\omega t} d\omega \quad (7)$$

via the definition of a Fourier transform and remembering that we are ignoring the negative frequencies. The reference frequency can be pulled out in this case as well, as was done above:

$$\begin{aligned}\tilde{E}(z, t) &= \int_0^{\infty} \tilde{E}(z, \omega_R + \Delta\omega) e^{j(\omega_R + \Delta\omega)t} d\omega = \\ &= e^{j\omega_R t} \int_0^{\infty} \tilde{E}(z, \omega_R + \Delta\omega) e^{j\Delta\omega t} d\omega = e^{j\omega_R t} \int_{-\infty}^{\infty} \tilde{E}_{\text{shift}}(z, \Delta\omega) e^{j\Delta\omega t} d\Delta\omega\end{aligned}\quad (8)$$

Again, at the result is a Fourier transform of  $\tilde{E}_{\text{shift}}(z, t)$ , yielding:

$$\tilde{E}(z, t) = e^{j\omega_R t} \tilde{E}_{\text{shift}}(z, t) \quad (9)$$

Combining Eqs. (6) and (9) and recalling that  $v_g = c/n_g$  gives:

$$\tilde{E}_{\text{shift}}(z, t) = e^{-j\beta(\omega_R)z} \tilde{E}_{\text{shift}}(0, t - \frac{zn_g}{c}) \quad (10)$$

We note that  $\tilde{E}_{\text{shift}}$  is the *analytic* signal passed around in our code, as it has had the reference frequency removed. We further note that the measureable E-field would be the real part of  $\tilde{E}(z, t)$ . So, to implement the waveguide in continuous time, a phase shifted  $\tilde{E}_{\text{shift}}$  (by  $e^{-j\beta(\omega_R)z}$ ) is output after a delay of  $n_g L/c$ , where  $L$  is the waveguide length. A propagation loss term,  $\alpha_{\text{field}}$ , is also added and thus the waveguide is modeled in VerilogA as shown in Code Sample 1 in the Appendix.

The equations governing other components can be similarly derived. Phase-shifters can be modeled as waveguides where the effective group and refractive index, as well as the loss, vary, possibly nonlinearly, with applied voltage. The code for such an optically non-ideal, but electrically ideal phase shifter is, thus, very similar to that derived for a straight waveguide. The main section of this code is given in Code Sample 2 in the Appendix.

Here, the wavevector is given by  $\beta(V) = \left(\frac{2\pi}{\lambda}\right)n_r(V)$ , where  $n_r$  is the refractive index as a function of voltage, and is given, for example, by:

$n_r(V) = n_0 + n_1V + n_2V^2 + n_3V^3 + n_4V^4 + n_5V^5$ , and where the loss as a function of voltage is similarly given by  $\alpha_r(V) = \alpha_0 + \alpha_1V + \alpha_2V^2 + \alpha_3V^3 + \alpha_4V^4 + \alpha_5V^5$ . Such a phase-shifter definition is general and can be used regardless of what electrical or thermal effect is being used to generate the index and loss change. However, in practice, the response of the underlying electrical structure, in addition to the optical response, is often *critical* for simulations, particularly when driving circuitry is being designed. Reverse bias silicon diode phase shifters that rely on the plasma dispersion effect to create a phase shift are commonly used in integrated photonic platforms. These phase shifters can be modeled like the non-ideal phase shifter above, but with the addition of the underlying electrical diode structure, as shown in the Code Sample 3 code in the Appendix. This allows for the capture of electrical effects (such as speed limitations due to diode resistance and capacitance), as well as optical effects (such as speed limitations due to the optical lifetime of signals in resonances).

Resistance based phase-shifters, in which resistive heating is used to generate a phase-shift are also common. Electrically the device can, of course, be modeled as a resistor. Optically, the correct way to implement such an integrated heater is to first integrate the electrical power put into the device minus the thermal power dissipated from the device, then proceed to calculate the phase shift as a function of the device's temperature. To do this we create a *thermal discipline* that keeps track of temperature. Here the index change is given by:

$$\Delta n = n_T T \quad (11)$$

where  $n_T = \partial n / \partial T$  is the refractive index change with temperature, and  $T$  is the temperature, calculated by:

$$T(t) = T_0 + \int_0^t [Z_T P(t) - h(T(t) - T_0)] dt \quad (12)$$

where  $Z_T$  is the thermal impedance,  $P$  is the electrical power (given by  $P = V^2/R_0$  where  $V$  is the applied voltage and  $R_0$  is the electrical resistance),  $T_0$  is the temperature when the potential across the resistor is zero, and  $h$  is a heat transfer coefficient. A detailed description of integrated heater operation can be found in [27]. This method correctly takes into account the transient behavior of the heating element, which limits the thermal bandwidth of the device. The relevant lines of code are reproduced in the Code Sample 4 in the Appendix. The above phase-shifters as well as several other flavors (e.g. a forward-biased diode shifter) have been implemented in our code.

Waveguide couplers are modeled in our code as point couplers: a fraction of the light is transferred, with appropriate phase shift, to the adjacent waveguide and the rest remains behind. Both adiabatic 3dB couplers and directional or ring couplers, where the field coupling fraction is specified, have been implemented. The code for a directional coupler is shown in Code Sample 5 in the Appendix. In this case the light entering through one port is split into two output ports, with field coupling,  $kappa$ , and implements the well known equations given

in [30], which are valid in the time domain:  $E_{Out\_Top} = \frac{E_{In\_Top}}{\sqrt{2}} + \frac{E_{In\_Bottom} e^{-i\pi/2}}{\sqrt{2}}$ . More

information on the theory of directional couplers can be found in [30], and on adiabatic couplers can be found in [31]. Our model does not currently include any propagation delays or loss through the coupler, but these could easily be added either directly into the model or by placing an appropriate waveguide block before or after the device ports.

Similar principles are used to write models for photodetectors. In the ideal photodetector, the instantaneous optical intensity is simply measured and transferred to the electrical domain. Such an ideal diode can be modeled by specifying a responsivity,  $R$ , and setting  $I_{out} = RP$ , where  $I_{out}$  is the output current, and  $P$  is the input optical power. Photodiodes form a common

class of integrated photodetectors. In this non-ideal case, the underlying diode structure is included to allow for electrical responses such as frequency roll-off and dark current to be captured. Due to the short length of the device, which can be as little as  $4\mu\text{m}$  [32], it does not make sense to include the optical delays in the device model. The code for such a photodiode including electronics is shown in Code Sample 6 in the Appendix.

We note that parameters like photodiode quantum efficiency are folded into the responsivity term and therefore do not need to be specified explicitly [33]. We further note that, while the included diode limits the electrical bandwidth of the device, the above code specifies a photodiode of infinite optical bandwidth as it captures the input power at all optical frequencies. Because of this infinite optical bandwidth, it is not necessary to consider the absolute frequency in this first order model. In order to place an optical limit on the bandwidth of light detected, one can place an additional filter in the code between the original calculation of the optical power and the specification of the electrical current. This filter, which can also be shifted down in frequency and specified in relation to the reference frequency, can cut out wavelengths to which the detector is not sensitive.

Other useful components, such as continuous-wave (CW) lasers and Gaussian pulse emitting lasers, as well as debugging components, such as phase detectors, are also included in our toolkit. Once this simulation framework has been established and these simple component models, which we will now refer to as optical primitives, have been implemented, complex optical systems can be built and combined with CMOS device cells to model complicated systems in a *truly-joint* fashion.

### 3. Basic composite devices

Because the optical components are implemented as basic physical primitives they must be combined together to create full devices. Our simulation technique can be used to model very complicated devices and systems that are beyond the capabilities of more traditional optical simulation techniques. To validate our technique, however, we also modeled several devices simple enough for existing optical simulation techniques to handle and found that our results matched. Since our codes using these traditional techniques have previously been validated against experiments (see, for example, [2,34]), this is equivalent to experimentally validating our technique.

The first of these devices is a ring resonator, which is created out of a waveguide connected in feedback between two ports of a coupler. A diagram showing this system is given in Fig. 2(a). The frequency response of the ring resonator is shown in Fig. 2(c). Because this is a time domain code, to get the frequency response it was necessary to sweep the input CW laser frequency. This is done by having a voltage controlled phase ramp inside the laser (as a phase ramp in the time domain corresponds to a frequency shift in the frequency domain) and running multiple DC simulations with different voltage inputs to this phase ramp. A diagram of the full simulated system is shown in Fig. 2(b) The obtained spectral response of the ring is compared and shown to match well to the results from an existing transfer matrix code implemented in MATLAB.

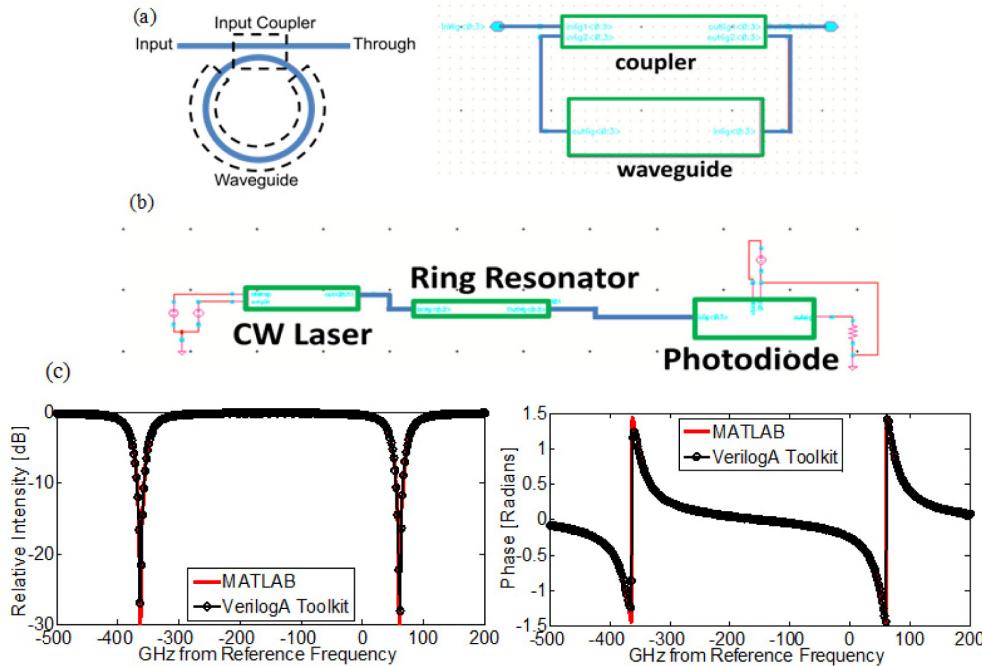


Fig. 2. (a) Diagram showing the components of a ring resonant filter and accompanying layout using our toolkit in Cadence Virtuoso. (b) A full link schematic including laser and photodetector is created to simulate the performance of the device. (c). Amplitude and phase response of the ring showing good agreement between VerilogA simulations and MATLAB transfer function code.

We can further compare the results to theoretical expectations. The free spectral range (FSR) of a ring resonator can be analytically found to be  $FSR = c/n_g L$ , where  $c$  is the speed of light in vacuum,  $n_g$  is the group index, and  $L$  is the effective waveguide path length. The results in Fig. 2(c) are for a SiN waveguide 200nm tall and 1 $\mu$ m wide that has a group index of 1.72 and is formed into a ring with a 66 $\mu$ m radius. The expected FSR is therefore 423GHz, as it is in the simulation. The effective refractive index of this waveguide is 1.516. We therefore expect the closest resonant frequency to 1550nm ( $= 193\text{THz}$ ), the reference frequency used in this simulation, to be at  $\lambda = n_{eff}L/m = 1549.5\text{nm}$  or 60.9GHz above the reference frequency. This value is also the same as that seen in Fig. 2(c). Finally, the half width at half maximum (HWHM) of the resonance was found to be 17.4GHz, also exactly what is expected given the input waveguide field loss,  $\alpha$ , or  $312.7\text{m}^{-1}$  and that  $HWHM = ca/n_g\pi$ . Moreover, not only does the amplitude response of the ring match that expected, but the phase response (measured using a debug block that outputs the instantaneous phase), shown in Fig. 2(c) also matches, going to  $\pm\pi/2$  as the laser frequency approaches the resonant frequency. Thus, it can be seen that our VerilogA toolkit also captures phase effects correctly.

To further expand on this example we can consider the case where the ring has a drop port and the case of a two ring filter. Both modifications are very simple to make using our toolkit, making it easy for designers to change systems on the fly. We compare both cases below to the results of transfer matrix codes implemented in MATLAB. No new calculations or equations were required when using our VerilogA toolkit; however, the new transfer matrices needed to be calculated and input into our MATLAB codes.

To add a drop port, we simply add an output coupler as shown in Fig. 3(a). One can then use this model in a system or to explore how to best optimize the component for a desired application. For example, in a single ring resonator with drop port, it has been shown [35]

that, given a target HWHM, the drop port output intensity is maximized when the input and output coupler coefficients are equal. On the other hand, the through port intensity is minimized at critical coupling where the input coupler coefficient equals the sum of the waveguide and output coupler loss. VerilogA simulations of both cases are shown in Fig. 3(b) and compared to transfer matrix codes, where we observe excellent agreement.

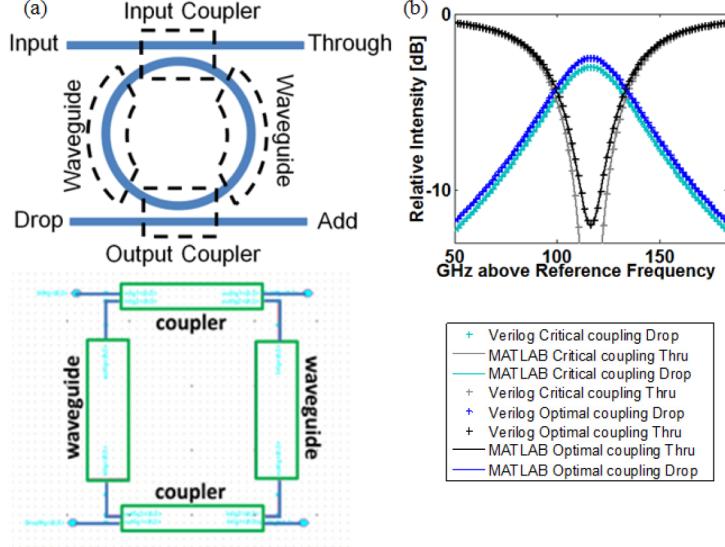


Fig. 3. (a). Diagram showing the components of a ring resonant filter with drop port and accompanying layout using our toolkit in Cadence Virtuoso. (b). Amplitude response of the ring showing output intensity in the through and drop port. One can see that the critical coupling has better extinction in the through port but less power in the drop port output, while matched coupling has worse extinction and higher drop port output power, as expected. Note that the waveguide loss was exaggerated in these cases to highlight the difference.

The two-ring series filter case is shown in Fig. 4(a). As can be seen, creating this case with our toolkit is similarly straightforward. One simply adds a second ring to the bottom of the first. Further, basic schematic symbol of this and other ring filters exist in our tool kit making it even easier for a designer to switch filters to tune the roll-off, ripple, passband width, or other figures of merit, and re-simulate. Two-ring filters are often helpful in systems such as switching matrices where having a wider and flatter passband allows for a wider bandwidth signal to propagate through. However, because two resonances are combined to make the larger resonance, ripples are often introduced into the passband. All these effects are captured in the simulation output, which is shown in Fig. 4(b) and also matches exactly the expected output calculated via transfer matrices in MATLAB.

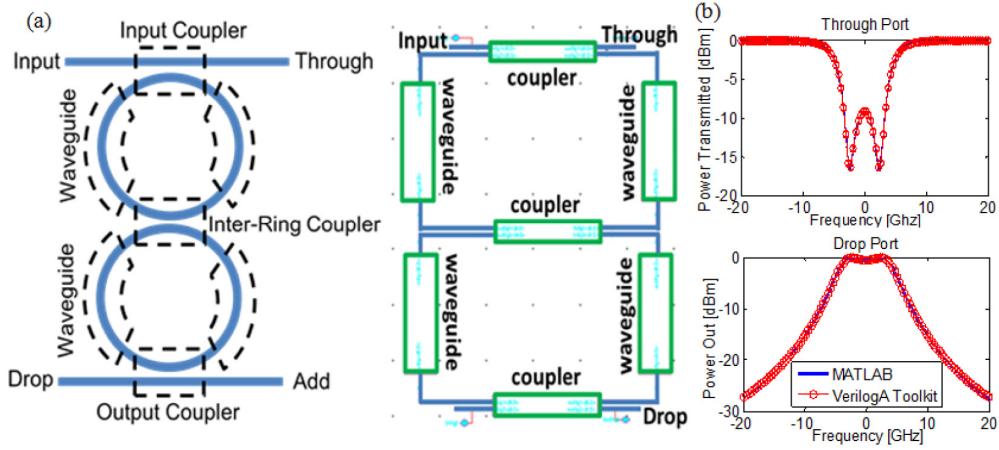


Fig. 4. (a). Diagram showing the components of a two ring series filter with drop port and accompanying layout using our toolkit in Cadence Virtuoso. (b). Amplitude response of the ring showing output intensity in the through and drop port showing good agreement between VerilogA simulations and MATLAB transfer function code.

The above demonstrations all involve passive devices, but simulating active devices in our toolkit is equally straightforward as is illustrated in the next section.

#### 4. Active devices

Active devices can be built in our tool kit in a way exactly analogous to the passive devices described above. In this section we will demonstrate the design and simulation of a ring modulator of the type often proposed for digital communication links [34]. We will also demonstrate the simulation of a single sideband modulator. Single sideband modulators are often used to create frequency shifts, and are important for both analog applications, such as arbitrary waveform generators, and digital applications, like dense wavelength division multiplexing systems [2].

A ring modulator can be made in our toolkit by replacing the waveguide component with a phase shifter in the resonant ring (see Fig. 5(a)). As described above, the phase shifter allows an input voltage to change the refractive index, group index, and loss of the waveguide, and thus the location of the resonance frequencies in a ring modulator. The shift in frequency of the resonance location of the modulator with voltage is shown in Fig. 5(b). The simulation is done for a five micron radius disk modulator made in a 220nm tall silicon layer. The output is compared to calculations done in MATLAB. Again, it matches the expected output.

However, the transient output, and not the DC output, is often what is of real interest in modulators. It is easy to simulate this using our toolkit: the input laser is simply set at one wavelength and the input voltage to the phase shifter changed from a DC to a pseudo-random-bit-sequence (PRBS) source (using a standard electrical PRBS block from the “ahdLib” library). The layout of this link is shown in Fig. 5(c), and the output in Fig. 5(d). As can be seen, when the input voltage is low, the optical amplitude inside the ring modulator is high and the through port output is low. Conversely, when the input voltage is high, and the resonance is shifted away from the laser frequency, the amplitude inside the ring is low, and the through port output is high, as expected. Further, we can clearly see the usefulness of a transient simulation by noting the roll-off in the output through the modulator.

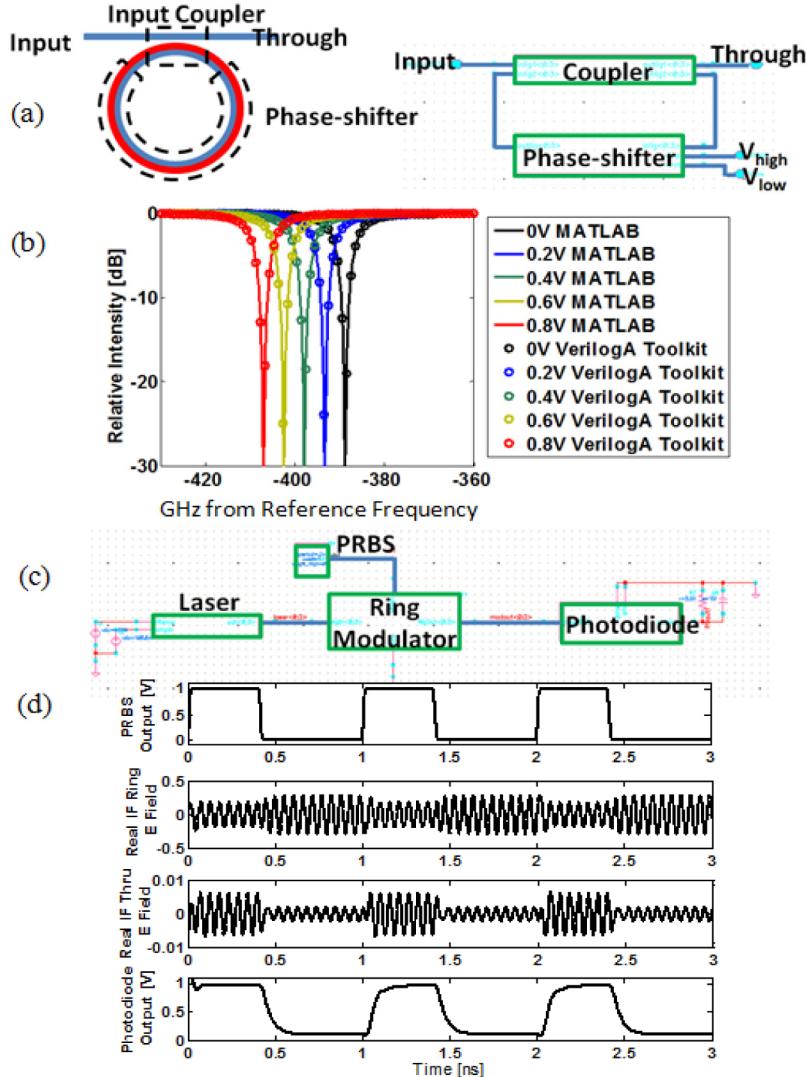


Fig. 5. (a). Diagram showing the components of a single ring modulator and accompanying layout using our toolkit in Cadence Virtuoso. (b). Modulator amplitude transfer function as a function of applied phase-shifter bias, showing good agreement between VerilogA simulations and MATLAB transfer function code. (c). Schematic of full single-wavelength digital link. Output from a CW laser is fed into a ring modulator driven by a PRBS source. The output of the modulator is sent to a photodiode. (d). Simulation output, showing electrical driving voltage, field amplitude inside the ring, through port field amplitude, and output of photodiode.

In this simulation, it is the optical time constant (the speed at which the optical mode in the resonator can ring down), and not the electrical time constant of the modulator or photodetector diodes (the speed at which the index can be changed or the intensity detected) that ultimately limits device speed. If the optical time constant were shorter (the ring were lower Q), or the electrical circuit slower, the frequency roll-off would be caused by the electronics instead, and this would also be captured by the simulation. Because the code models the optical interference inside the ring with each time step, all these transient phenomena are neatly captured without any explicit modeling. No changes to the inside of the blocks or the overall system code are needed; we just have to draw the schematic for the desired systems and run the preferred simulation.

An example of a device where phase and frequency effects are of importance is a single sideband modulator (SSB). SSBs are important for many analog applications as well as for high density WDM communication systems as they allow for closer channel spacing and lower power [36]. Additionally, the same topology can be used to transmit data using more complicated modulation formats (BPSK, QAM, etc). Correctly modeling the phase effects and delays of components is vital to simulating single sideband operation, and as we will demonstrate below, our VerilogA Toolkit is capable of doing this. We note again that the transfer function of the SSB is never calculated explicitly; instead the correct output is generated automatically from the combination of basic optical blocks.

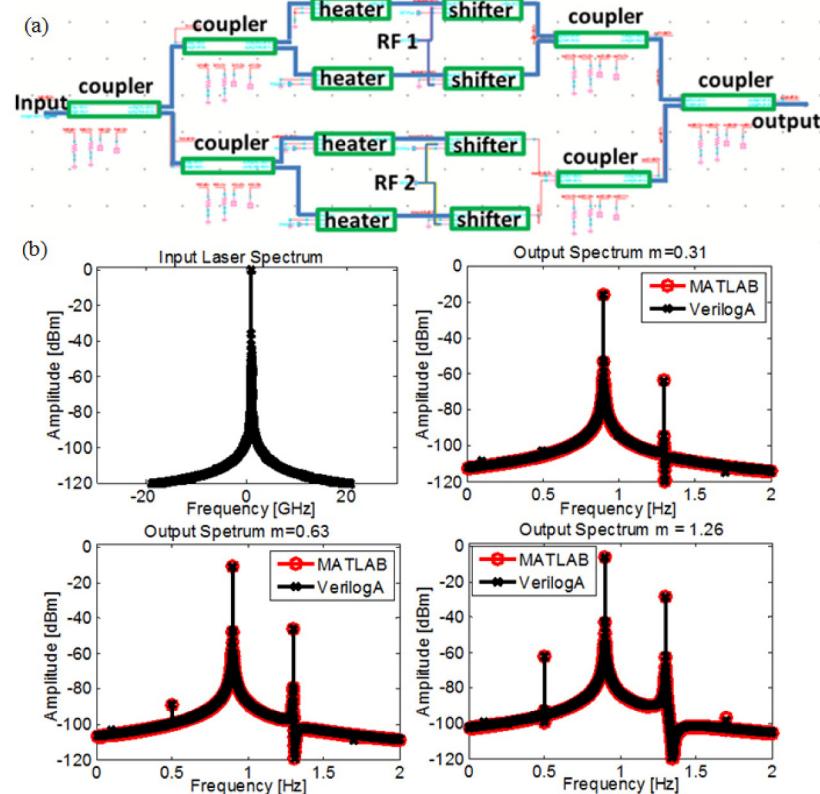


Fig. 6. (a.) Diagram of single side band modulator component system (from reference [2]) and accompanying layout using our toolkit in Cadence Virtuoso using basic components (couplers, heaters, phase-shifters). (b.) Fourier transform of SSB output as a function of modulation amplitude. Input CW signal is shown in black in upper left. Following graphs show steadily increasing modulation depth. Increase in power transferred to desired sideband and accompanying increase in power in undesired harmonics is seen. VerilogA and MATLAB code show good agreement.

A schematic layout of the SSB topology using our VerilogA toolkit is shown in Fig. 6(a). The SSB is made from two nested Mach-Zehnder modulators (MZMs), with each MZM composed of the coupler and phase-shifter primitives, and generates a single sideband carrier suppressed signal. To do so each individual MZM is biased using a heater at the zero output point (thereby suppressing the carrier), and driven in push-pull. The electrical RF input signals to each modulator are out of phase by  $\pi/2$ , such that one arm receives a cosine modulation and the other a sine modulation. An extra  $\pi/2$  phase added by heaters to one of the MZM signals therefore allows one sideband to cancel and the other to add. Which sideband is kept depends on which arm receives the extra  $\pi/2$  phase relative to which receives the sine

(vs. the cosine) RF input signal. For more detail about SSB operations see [2,36]. This surviving sideband is the output of the device, and is given by  $f_{out} = f_{opt} - f_{rf}$  or  $f_{out} = f_{opt} + f_{rf}$ , where  $f_{opt}$  is the original optical frequency and  $f_{rf}$  is the modulation frequency. We choose to keep the lower sideband.

The output of the simulation using our VerilogA toolkit is shown in Fig. 6(b). Here a CW laser block inputs a CW signal to the SSB. Since our code operates entirely in the time domain, the Fourier transform was computed in a post-processing step. In a real system model, the time domain output samples would simply be passed on to the next block with no transform necessary. To verify that the VerilogA model is in fact working correctly, we compare it to the output of our MATLAB system code for this device. As above, we have previously verified this MATLAB code against experimental results, and it has been found to match well. Figure 6 shows the case of a 0dBm laser output that is 1GHz from the reference frequency and is modulated at 100MHz to yield a sideband frequency of 900MHz. The presence of a single, frequency shifted tone at the output can clearly be seen -the carrier and unwanted sideband are well suppressed in all cases. As the modulation depth,  $m$ , increases the amount of power transferred to the desired sideband increases as does the relative height of the higher harmonic distortions. These last are caused by the increasing interaction with the nonlinear portion of the MZM transfer function and are given by the Bessel function expansion for the sine within the sine [2,36]. All these expected effects can be clearly seen in the output spectra shown in Fig. 6(b). Further, there is very good agreement between the MATLAB and the VerilogA models in all cases, indicating that our VerilogA framework is working correctly. It is worth noting that writing and running the MATLAB® code took more time and was more complicated than setting up and running the SSB modulator schematic in VerilogA, thus demonstrating the convenience and one of the benefits of having a single, flexible simulation platform.

## 5. Simulating systems

In the preceding section, we demonstrated the ability of our system to correctly simulate simple, composite components, and basic devices. The ultimate goal of our work is to be able to simulate complex systems, particularly ones that include the electrical driving circuitry and noise models. In this section we will take the basic and composite device blocks and use them to build full heterogeneous electrical and optical systems. Note how similar the process is to building circuits out of transistors, by simply instantiating components, setting parameters, and connecting the waveguides and electrical wires, further allowing for easy combination with existing electrical simulation infrastructure and ease of use by designers familiar with such. While the toolkit can of course be used to build a multitude of different systems and simulate them at numerous operating points under a myriad of conditions, we will focus on two examples here: a WDM link and a Pound-Drever-Hall laser stabilization loop.

We begin with the WDM link. One of the driving forces for integrated silicon photonics is to use WDM over optical waveguides to replace copper wires and solve the bandwidth density/energy consumption problem for data transmission links [37]. A multi-frequency laser source provides a comb of frequencies, each of which can be modulated with independent data streams. Since there are many optical and electrical components involved, tradeoffs in many dimensions need to be made to optimize the desired figure of merit for the link. Architecture-level analysis [38] gives us useful insight to identify major bottlenecks, but for any deeper understanding, a physical simulation tool is necessary.

The schematic layout of a WDM link simulation using our toolkit is shown in Fig. 7(a). To simulate the link we pass the output of a WDM laser block, which simultaneously outputs five CW frequencies, through five modulators and five ring filters in series on a single waveguide. The drop port of each ring filter is connected to a photodiode. Four of the ring modulators are driven by four different PRBS data streams. The fifth is driven by a clock

stream. The simulated input and output electrical waveforms are shown in Fig. 7(b). Note that the optical rise and fall times and signal delays are neatly captured.

Since all five channels are represented as one complex field signal in the waveguide, very subtle crosstalk can also be seen. That is because in this design, the channels are spaced to be 100GHz apart, while the ring modulator FWHM is 13.8GHz. Depending on other tradeoffs, the optimal channel spacing might be even closer, and this simulation toolkit will allow the designer to inspect the resulting bit-error-rates (BER) in the presence of channel crosstalk. Other non-idealities, such as thermal effects, thermal noise, transistor noise, can be added to the simulation to provide a more realistic prediction of a systems behavior.

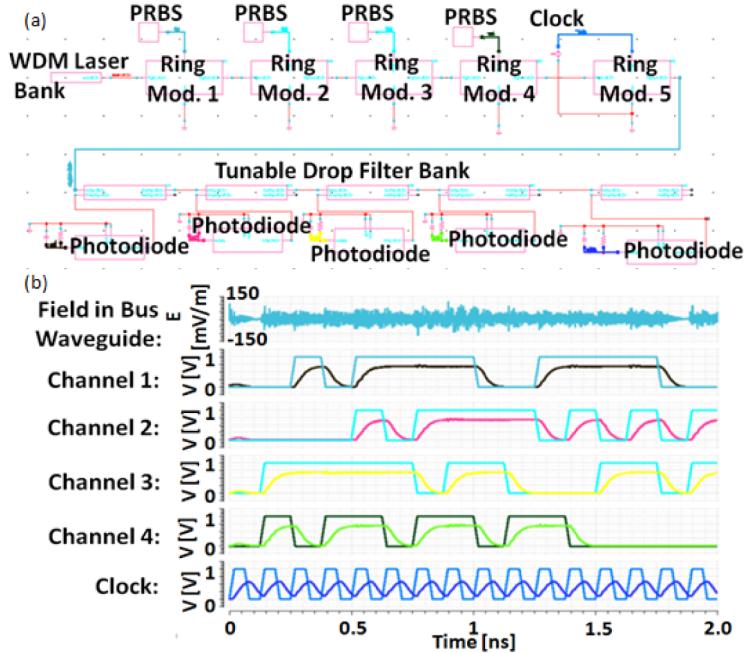


Fig. 7. (a). Schematic of WDM link with four PRBS channels and one clock channel. (b) real part of signal on WDM waveguide (top), input and output of modulator channels (middle four channels) and clock (bottom).

We next explore the operation of a purely analog, and extremely phase sensitive system, a Pound-Drever-Hall (PDH) locking loop for laser frequency stabilization, using our code. This set of simulations is also more complex as it additionally makes use of noise blocks and of standard CMOS cells to model more complicated electrical driving circuits. The theory of PDH operation is described in [17]. To summarize the concept briefly: the output of a laser is sent through a phase modulator modulated by a sine wave from an RF oscillator and then past a resonant cavity, in our case a ring resonator. The optical carrier picks up a different phase-shift in relation to the modulation sidebands depending on its frequency relative to the ring reference frequency. The carrier and the sidebands are beat together on a photodiode and the resulting RF signal is mixed down to baseband using the same, equally delayed, signal from the RF oscillator. The DC component of the output is the PDH error function and its sign indicates whether the laser frequency needs to be adjusted up or down to match that of the cavity. The error function is fed back on the laser and decreases laser phase noise by “locking” the laser frequency to that of the resonant cavity. The PDH scheme has two operation regimes: a low modulation frequency regime where both the laser and the modulation sidebands fall within the cavity resonance, and a high modulation frequency regime where the sidebands remain outside of the cavity.

A schematic layout of this system, using ideal photonic and electronic components, is shown in Fig. 8(a). The error function output in both the low modulation and high modulation regimes is shown in Fig. 8(b), where they are compared to, and show excellent agreement with, results generated using hand calculated transfer-functions and joint time and frequency domain simulations in MATLAB. As in the SSB case writing and running the MATLAB code to simulate the output of the PDH took more time and was more complicated than setting up the schematic and running the VerilogA code. Note that the VerilogA output closely matches the MATLAB output.

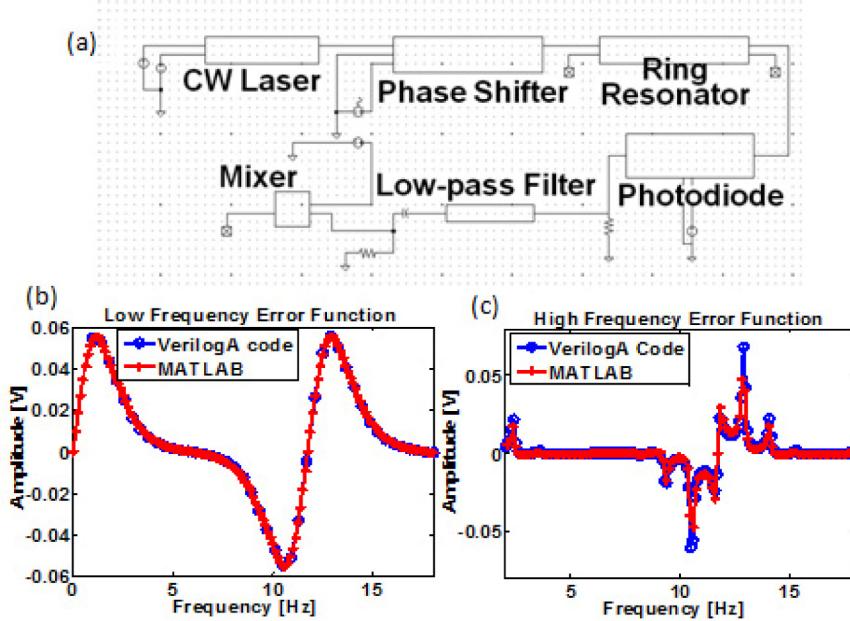


Fig. 8. (a) Schematic layout in Cadence Virtuoso of PDH loop. (b) PDH error function in low and high frequency regimes, showing excellent agreement between MATLAB and VerilogA code.

To predict the performance of a real system with non-ideal components, a new schematic is created using accurate, non-idealized blocks for the photonics components and the actual CMOS driving circuitry in a 65nm 10LPe CMOS process. The parameter values in blocks for the photonics components are based on measured results from devices our group has designed and tested in the lab. The driving circuitry blocks shown in the schematic are designed in-house using the CMOS component models provided by the CMOS foundry, to satisfy the needs and constraints of the photonic components and of the full system. Figure 9(a). shows a schematic of this full system. As can be seen, this schematic is significantly more complex than the idealized schematic in Fig. 8.

Simulations of the whole system can then be run, and performance achievable in the lab can be predicted. Furthermore, sensitivity to operating point, operating conditions, fabrication variations, various optical and electronic noise sources, and changes in component type or design can be explored. This allows for a much better understanding of system performance and robustness than previously possible. A closed loop simulation shows the PDH in transient operation, as the feedback signal from the PDH loop changes the laser frequency to the cavity frequency (Fig. 9(b).) To further demonstrate the code, we add phase noise to our laser using standard VerilogA noise blocks. A noise analysis is performed in Cadence Virtuoso, as would be done for purely electrical circuits. The resulting laser phase noise in both the open loop (unsuppressed) and closed loop (suppressed) case are shown in Fig. 9(c). For this particular

set of loop parameters, the PDH lock is successful and mid-frequency phase noise is suppressed by >50dB.

Thus, our VerilogA toolkit allows us to successfully and usefully model the performance of complex hybrid systems.

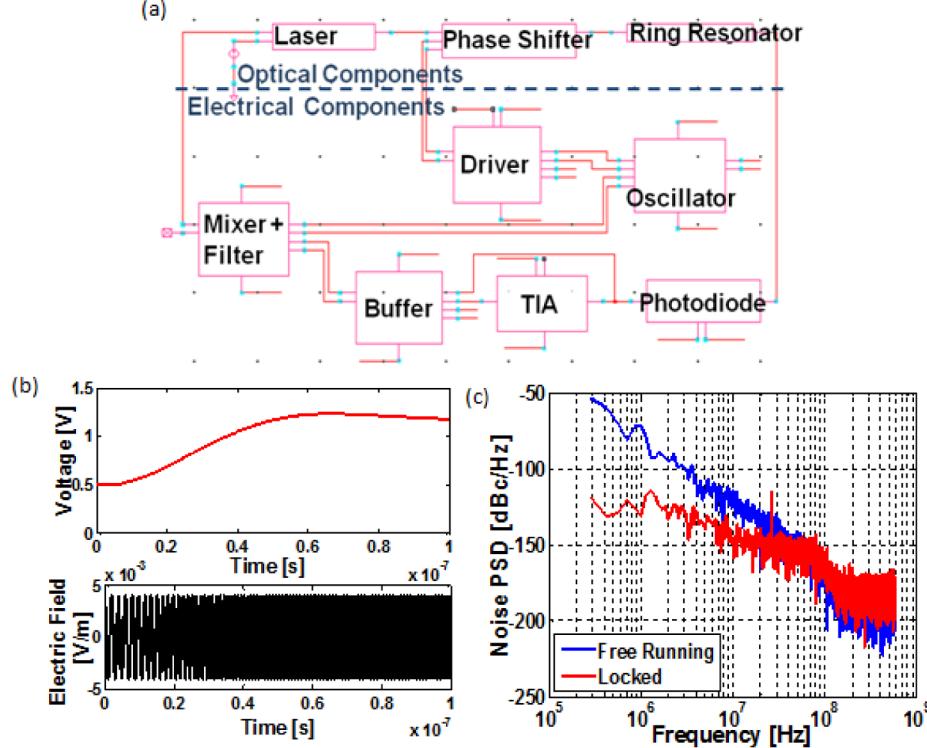


Fig. 9. (a) Schematic layout of PDH loop using non-ideal photonics and full driving circuitry in Cadence Virtuoso. (b) Laser frequency locks to cavity reference in 100ns (red), real part of laser electric field as locking occurs (black). (c). Phase noise simulation done using our toolkit showing noise of the free-running CW laser (blue) and of the laser locked to a cavity (red) showing successful phase-noise suppression.

## 6. Conclusions

A Cadence toolkit, written in VerilogA, for simulating optical devices in conjunction with electronic circuits has been presented and verified. To correctly capture phase-phenomena and interference effects while avoiding numerical instabilities, the optical electric field is represented by the real and imaginary parts of both a forward and backward propagating signal. Relevant optical electrical interactions are captured. To minimize run time and simplify coding, the analytic signal representation is used, and the simulations are carried out relative to an arbitrarily chosen reference frequency. A set of fundamental photonic components are identified, and each is described at the physical level, such that the characteristics of any composite device will be created organically. We show that resonant effects in composite devices such as ring filters and modulators, and phase-effects in composite devices like single sideband modulators are correctly, organically captured by these models. We further show the ability to jointly simulate optics and electronics for entire systems including a WDM link, and a Pound-Drever-Hall laser stabilization loop complete with standard cell models of all the CMOS driving circuitry and noise. Our toolkit opens the possibility for system designers to easily build and simulate complex heterogeneous electronic-photonic circuits.

## 7. Appendix

```

`include "../../../../cad/constants.vams"
`include "../../../../cad/disciplines.vams"

module Optical_Waveguide(inLight, outLight);
    inout [0:3] leftLight; // [0:1] is left side input, [2:3] is left side output
    inout [0:3] rightLight; // [0:1] is right side output [2:3] is right side input. optical
    [0:3] rightLight, leftLight;

    // Physical Design Parameters, all units in SI
    parameter real L = 0.0005;           // length
    parameter real ng = 4.1963;          // group index
    parameter real np = 2.1;             // refractive index
    parameter real alphaA = 287.6;       // loss
    parameter real G_freq = 1.93e14;     // reference frequency parameter passed in from
    top level simulation

    // initialize intermediate points
    optical [0:1] rightOutput;
    optical [0:1] leftOutput;
    optical [0:1] transfer; // for calculated phase-shift and amplitude change
    optical [0:1] transferCart; // for above in real and imaginary forms

    pol2cart convs1(transfer, transferCart); // convert waveguide effect to real and
    //imaginary parts
    cartmul mulout1(transferCart, leftLight[0:1], rightOutput); //Calc outputs
    cartmul mulout1B(transferCart, rightLight[2:3], leftOutput);

    analog begin
        //calculate phase-shift and amplitude change in polar coordinates
        E(transfer[0]) <+ (-L*np*2*M_PI*G_freq/P_C)%((2*M_PI);
        E(transfer[1]) <+ exp(-alphaA*L);

        // Output delayed signals
        E(rightLight[0]) <+ absdelay(E(rightOutput[0]), L*ng/P_C);
        E(rightLight[1]) <+ absdelay(E(rightOutput[1]), L*ng/P_C);
        E(leftLight[2]) <+ absdelay(E(leftOutput[0]), L*ng/P_C)
        E(leftLight[3]) <+ absdelay(E(leftOutput[1]), L*ng/P_C);
    end
endmodule

```

Code Sample 1. VerilogA waveguide model code. “pol2cart” is used to convert the waveguide transfer function from the amplitude and phase representation to the real and imaginary one. Note that the input and output optical *signals* are *never* converted to the amplitude and phase representation, thus avoiding any convergence issues. “cartmul” is used to do complex multiplication on the real and imaginary signals.

```

...
analog begin
    //calculate effective indices and effective loss based on applied voltages
    Va = V(vtop, vbot);
    neff = n0+n1*Va+n2*Va*Va+n3*pow(Va,3)+n4*pow(Va,4)+n5*pow(Va,5);
    ng = ng0+ng1*Va+ng2*Va*Va+ng3*pow(Va,3)+ng4*pow(Va,4)
        +ng5*pow(Va,5);
    aeff = a0+a1*Va+a2*Va*Va+a3*pow(Va,3)+a4*pow(Va,4)+a5*pow(Va,5);
    //calculate phase-shift and amplitude change in polar coordinates
    E(transfer[0]) <+ ((-1*neff*L*2*M_PI*G_freq)/P_C)%(2*M_PI);
    E(transfer[1]) <+ exp(-aeff*L);
    // output delayed signals
    E(rightLight[0]) <+ absdelay(E(rightOutput[0]), L*ng/P_C);
    E(rightLight[1]) <+ absdelay(E(rightOutput[1]), L*ng/P_C);
    // and set backward propagation
    E(leftLight[2]) <+ absdelay(E(leftOutput[0]), L*ng/P_C);
    E(leftLight[3]) <+ absdelay(E(leftOutput[1]), L*ng/P_C);
end
...

```

Code Sample 2. VerilogA phase-shifter model code.

V<sub>a</sub> is the voltage applied across the phase shifter, and n0...n5, ng0...ng5, and a0...a5 are the coefficients governing the voltage to effective refractive index, effective group index, and effective loss respectively (as either calculated from device simulations or measured from experiment).

```

// set-up of code analogous to above, also see photodiode code below...
...
// electrical part:
Vt = 'P_K*Temp/'P_Q;                                // kT/q = Thermal voltage [V]
q_diode = - 2 * Cjo * Vbi * sqrt(1 - V(cap) / Vbi); // diode charge
if (V(cap) > Vbi) begin                            // diode must be used in reverse bias
    $strobe("photodiode voltage V(cap)>Vbi");
    $finish;
end
I(res) <+ Is * (limexp(V(res) / Vt) - 1);          // current through resistive branch [A]
I(cap) <+ ddt(q_diode);                            // current through capacitive branch [A]
V(rseries) <+ Rseries*(I(res)+I(cap));             // ohm's law equation for series

// optical part:
Va = V(cap);
neff = n0+n1*Va+...
...
// end of code analogous to above...

```

Code Sample 3. VerilogA electrical reverse bias phase-shifter model code.

```

...
parameter real L = 0.0005;           // length
parameter real ng = 4.1963;          // group index
parameter real n0 = 2.1;             // refractive index with no heating
parameter real Zt = 57000;           // thermal impedance
parameter real R0 = 2000;            // Resistance
parameter real h = 1.9E-4;           // heat transfer coefficient
parameter real nT = 0.0001;          // change in index with temperature
parameter real T0 = 293;             // ambient temperature
parameter real a0 = 0.0;             // waveguide loss
parameter real a1 = 0.0;             // voltage dependence of waveguide loss (dα/dV)
...
optical [0:1] heaterPol;
optical [0:1] heaterCart;
thermal Htemp;
...
pol2cart convs1(heaterPol, heaterCart);
cartmul multout1(heaterCart, leftLight[1:0], rightOutput);
cartmul multout1B(heaterCart, rightLight[2:3], leftOutput);
...
analog begin
  Va = V(vtop, vbot);
  Temp(Htemp) <+ T0 + idt(Zt*Va*Va/R0-h*(Temp(Htemp)-T0),0); // Waveguide temperature
  neff = n0+nT*Temp(Htemp);                                     // Temperature induced phase change
  alphaeff = a0+a1*Va;                                         // Loss
  // Calculate phase shift and amplitude change in polar coordinates
  E(heaterPol[0]) <+ (-1*neff*L*2*M_PI*(G_freq)/P_C)%(2*M_PI);
  E(heaterPol[1]) <+ limexp(alphaeff*1e-4*L);
end
...

```

Code Sample 4. VerilogA thermal phase-shifter model code.

```

`include "../../../../cad/constants.vams"
`include "../../../../cad/disciplines.vams"

module Waveguide_Coupler(leftLight1, leftLight2, rightLight1, rightLight2);
  inout [0:3] leftLight1, leftLight2, rightLight1, rightLight2; // [0:1] is left side input, [2:3] is
  left side output
  optical [0:3] leftLight1, leftLight2, rightLight1, rightLight2;

  // Physical Design Parameters, all units in SI
  parameter real kappa = 0.5; // kappa is FIELD coupling
  parameter real G_freq = 1.93e14; // reference frequency parameter passed in from top level
  simulation

  // Initialize intermediate points
  optical [0:1] Xcoup;
  optical [0:1] Xthru;
  optical [0:1] leftLig1X;
  optical [0:1] leftLig2X;
  optical [0:1] leftLig1T;
  optical [0:1] leftLig2T;
  optical [0:1] Xcoup2;
  optical [0:1] rightLig1X;
  optical [0:1] rightLig2X;
  optical [0:1] rightLig1T;
  optical [0:1] rightLig2T;

analog begin
  // Calculate coupling matrix
  E(Xcoup[0]) <+ 0.0;
  E(Xcoup[1]) <+ -kappa;
  E(Xcoup2[0]) <+ 0.0;
  E(Xcoup2[1]) <+ -kappa;
  E(Xthru[0]) <+ sqrt(1-kappa*kappa);
  E(Xthru[1]) <+ 0;
end
// Forward propagating wave:
  cartmul in1X(leftLig1[0:1], Xcoup2, leftLig1X);
  cartmul in2X(leftLig2[0:1], Xcoup, leftLig2X);
  cartmul in1T(leftLig1[0:1], Xthru, leftLig1T);
  cartmul in2T(leftLig2[0:1], Xthru, leftLig2T);
  cartadd comb1(leftLig1T, leftLig2X, rightLig1[0:1]);
  cartadd comb2(leftLig1X, leftLig2T, rightLig2[0:1]);
// Backward propagating wave
  cartmul Bin1X(rightLig1[2:3], Xcoup2, rightLig1X);
  cartmul Bin2X(rightLig2[2:3], Xcoup, rightLig2X);
  cartmul Bin1T(rightLig1[2:3], Xthru, rightLig1T);
  cartmul Bin2T(rightLig2[2:3], Xthru, rightLig2T);
  cartadd Bcomb1(rightLig1T, rightLig2X, leftLig1[2:3]);
  cartadd Bcomb2(rightLig1X, rightLig2T, leftLig2[2:3]);
endmodule

```

Code Sample 5. VerilogA optical coupler model code. “cartadd” performs complex addition on the real and imaginary signals.

```

`include "../../../../cad/constants.vams"
`include "../../../../cad/disciplines.vams"

module PD_Electrical(p, n, inLight);
    inout [0:3] inLight;
    inout p, n;
    electrical p, n;
    optical [0:3] inLight;

    //Declare an internal node and branches
    electrical inter;
    branch (p.inter) cap, res, photo;
    branch (inter,n) rseries;
    //Physical Design Parameters
    parameter real Temp=300;           // In Kelvins
    //Derived Parameter Declarations
    parameter real Is=1e-12;          // Reverse-bias saturation current [A]
    real Vt;                      // Thermal voltage [V]
    parameter real Vbi=1;            // Built-in voltage [V]
    parameter real Cjo=1e-15;         // Unbiased junction capacitance [F]
    parameter real tau=1e-100;        // response time constant 10ps
    parameter real Rseries = 1.0;      //series resistance of diode [Ohms]
    real q_diode;                  // Charge in diode [C]
    //Optical Constants
    parameter real responsivity = 0.8;
    analog begin
        //Calculate Derived Parameters
        Vt = `P_K*Temp/P_Q;           // Thermal voltage [V]
        q_diode = - 2 * Cjo * Vbi * sqrt(1 - V(cap) / Vbi); //Charge
        if (V(cap) > Vbi) begin           // Diode must be used in reverse bias
            $strobe("photodiode voltage V(cap)>Vbi");
            $finish;
        end
        //Calculate photocurrent, filter by time constant
        I(photo) <+ laplace_nd(-responsivity*(pow(E(inlig[1]),2)+pow(E(inlig[0]),2)), {1}, {1,1/tau}); //Photocurrent
        I(res) <+ Is * (limexp(V(res) / Vt) - 1);           // Current through resistive branch [A]
        I(cap) <+ ddt(q_diode);                            // Current through capacitive branch [A]
        V(rseries) <+ Rseries*(I(res)+I(cap)+I(photo)); // Ohm's law equation for series resistance
        // This model assumes no reflections
        E(inLight[2]) <+ 0.0;
        E(inLight[3]) <+ 0.0;
    end
endmodule

```

Code Sample 6. VerilogA photo-detector model code.

### Acknowledgments

This work was supported by the Defense Advanced Research Projects Agency (DARPA) of the United States under the E-PHI project, grant no. HR0011-12-2-0007 and DARPA MTO Program Manager Dr. Josh Conway.