# PhoenixSim: Crosslayer Design and Modeling of Silicon Photonic Interconnects

Sébastien Rumley, Meisam Bahadori, Ke Wen, Dessislava Nikolova, Keren Bergman
Department of Electrical Engineering, Columbia University, New York, NY 10027
rumley@ee.columbia.edu

## ABSTRACT

Silicon Photonics is emerging as a key technology for high-performance computing interconnects. Yet few tools are available to investigate how to best leverage this technology in current or future computer architectures and, furthermore, how this technology will impact real application workloads. In this paper, we present a multi-layer simulation and modeling software solution – PhoenixSim. PhoenixSim enables integrated and interactive design space exploration over the physical, networking and application layers. In this paper, we report its general organization and constituting models. We show how the different layers of the tool can be utilized to design and analyze an optical interconnect network for supporting the HPCG (High Performance Conjugate Gradient) benchmark.

## Keywords

Silicon photonics, compact models, interconnects, software tools.

## 1. INTRODUCTION

Performance scaling of computing systems is increasingly limited by data communication over distances, even short ones. The prospect of low-cost, compact Photonic Integrated Circuits (PIC), able to provide ultra wide and dense bandwidth links, is therefore particularly alluring for computer architects. One of the main technologies underlying the PIC emergence is Silicon Photonics. Most basic operations required to build an optical interconnect have been shown realizable with Silicon photonics components: modulators, filters, waveguides, low-loss waveguide crossings and junctions, as well as photo receivers (direct detection). Multiple private research labs (e.g. Oracle, IBM) have progressed toward incorporating all these elements in a single integrated transceiver chip. In addition, these nano-scale components can be mass produced using conventional CMOS lithography processes [8], an aspect that should enable low-cost fabrication. Finally, monolithic integration of both photonic and electronic circuits in the same process has been shown feasible [2]. Photonic network interfaces might thus be directly integrated within Chip Multi-Processors (CMPs) or System-on-chip in the future, helping these to more efficiently communicate with memory modules and/or among themselves.

*AISTECS 2016, January 18 2016, Prague, Czech Republic*
© 2016 ACM. ISBN 978-1-4503-4084-7/16/01…$15.00
DOI: http://dx.doi.org/10.1145/2857058.2857061

The emerging possibility of placing photonic components in the core of computing systems is expected to drastically change current architectures [6]. Silicon Photonics links carrying 320 Gbps have been demonstrated [10], and recent predictions indicate that with fully mature components, several Terabit/s could be attained [3]. This constitutes a ten time increase compared to current electronic network interfaces and even a 100x increase compared to the peak signaling speed (25GHz) in use today. More generally, these numbers also indicate that future integrated photonics-based interconnects will be capable of a wide diversity of bandwidths: from a few tens of Gbps to several Tbps.

Integrated photonics thus opens new horizons. They also trigger a whole new set of questions and challenges. At the deepest level, the size, geometries and materials of each nano-device must be appropriately chosen to ensure proper operation of the system, and guarantee best performance in terms of optical signal quality and integrity, and power consumption. Several "Photonic Design Automation" (PDA) tools have been developed to ease and accelerate this device design process. Using FDTD or similar numerical methods, the behavior of each component, taken separately, can be assessed. By combining several of these components, an end-to-end communication system with optimized properties can be assembled.

If PDA tools are clearly required to effectively design and develop PIC, other tools are required alongside, however. Realizing a full link design within PDA tools is a time consuming process. Thus, it is worth investing this time to perfect an initial strawman design, but not to select this strawman design among many. For example, a link design integrating 50 parallel wavelengths, each modulated at 10G, can be finely optimized with PDA tools. However, prior to realizing this optimization, the confidence that this transmission format (50×10G) constitutes the most effective way to obtain 500 Gbps must be attained. To this aim, building tens of models corresponding to various products within PDA tools is possible but likely ineffective. Other approaches should thus be employed to rapidly *prescreen* and pinpoint designs of interest that would later be perfected with PDA tools.

PDA tools also show limited capacity in electrical circuit modeling. Unfortunately, electrical circuits (drivers, amplifiers) attached to modulators and photo-receivers are major performance determinants, particularly in terms of power dissipation [14]. As later shown in this paper and reported in [20], there are cases where compromises made on the optical signal quality side can result in overall energy savings when considering the link as a whole. Therefore, hybrid *EPDA (Electronic and Photonic Design Automation)* design tools incorporating the behavior (to the least at the level of the trend or rule-of-thumb) of both electrical and optical circuits are highly required.

Yet will such hybrid design tools be sufficient? In our opinion, they will not. In the above-mentioned example, the choice of the ideal 500Gbps transmission format hides yet more fundamental questions: 1) is 500 Gbps the appropriate value? 2) Will the resulting link, once fully optimized, fit in the target computing system? EPDA tools, combined with *prescreening* tools, may deliver sound designs of 500 Gbps (or any another value) links. They will provide, however, no guarantee on the practical usability of these links once inserted in the host computing system. The adequacy of a photonic interconnect with its host system is not only determined by the hardware parameters of its constituting devices. The presence of photonics at the core of the system will also affect the way network elements are controlled and arbitrated: if optical spatial switching is leveraged, typically, circuits will be the norm as long as credible optical buffers will be unavailable. This means selecting (and developing) appropriate circuit arbitration mechanisms and scheduling strategies. Additional software tools – network simulators in this case – are hence required to support the system designer in this decision process as well.

Finally, the answers to these various questions found at component and networking levels influence or even determine the whole system architecture. Reciprocally, host level decisions, as the number of processor cores present in one CMP, the number of CMPs, etc., will characterize the bandwidth requirements. Furthermore, optimizations made at the optical component level may trigger changes in the system architecture, which in turn may require further adaptations at both component and networking levels. In summary, the PIC emergence will shake traditional designs up, and widely novel architectures will be required. Granted the importance of data movement in future architectures, communication and computation facets of the full system will have to be defined jointly. This implies the existence of software tools making that codesign effort possible.

In summary, incorporating emerging silicon photonics links inside computing systems involves many steps, and solely optimizing the devices constituting a link to optimize its optical properties is clearly not sufficient. However, finding answers to the various questions raised by the PIC emergence implies exploring an extremely wide design space. Silicon photonic device parameters (and the (E)PDA tools permitting their exploration) are part of this design space. However, other tools are required to finely explore the other dimensions (network level, for instance), and more importantly, to perform holistic, crosslayer explorations.

The Lightwave Research Laboratory at Columbia University is investigating for more than a decade now how optical technologies can be leveraged to improve the performance of large-scale computing systems. In this context, the PhoenixSim crosslayer modeling and simulation framework has been developed. Over the years, PhoenixSim took various shapes and evolved significantly from its origins as lessons have been learned in the process. Our current version 2 shares only its name and goals with the initial version our group presented in [4]. The initial version was C++ written and used Omnet++ as simulation engine. Our current version, in contrast, is 100% java written and integrates its own event driven simulation engine. PhoenixSim version 2 recently reached a maturity point permitting us to obtain a large set of interesting (and sometimes intriguing) results. This paper aims at describing our framework in its current state, and at showing the type of optimizations and studies it now supports. The next Section provides an overview of the framework organization and main components. In Section 3, we focus on the hardware level and show how PhoenixSim can be used as early design exploration tool, as well as hybrid optical/electrical optimization tool. In Section 4, we present the network simulation capabilities of PhoenixSim. In Section 5, we detail how parallel applications can be emulated on top of the network simulator. In Section 6, we show how the different "layers" constituting PhoenixSim can be combined. In particular, we report the design of a medium-scale distributed system organized around a wavelength routed optical interconnect.

## 2. PHOENIXSIM OVERVIEW

PhoenixSim is organized around three "layers" meant to capture respectively: 1) the behavior, consumption and performance of physical devices (mainly optical ones) – *physical layer* 2) the latencies experienced by the messages sent over communications links – *network layer* 3) the progression of distributed (message passing based) application – *application layer*.

As shown in Figure 1, the physical layer connects to the network layer. Based on the device characteristics as well as the link or network architecture selected, bandwidth and power consumption parameters are calculated and propagated into the network simulation model constituting the network layer. The application layer is similarly connected to the network layer. As the execution of an application is emulated, messages are being emitted into the network and received from it.

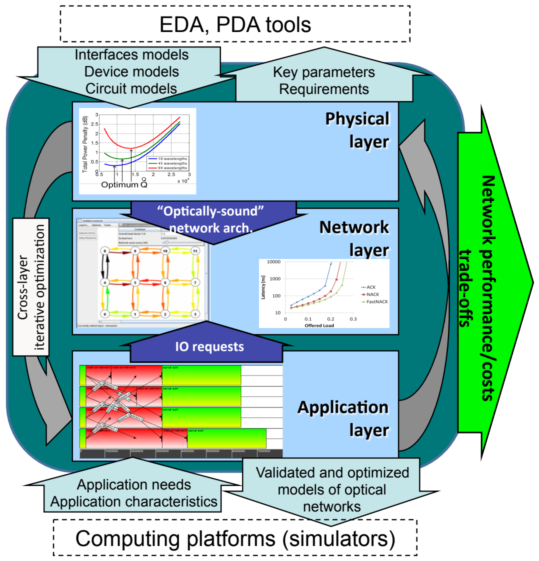

EDA, PDA tools
Interfaces models
Device models
Circuit models
Key parameters
Requirements
Physical layer
Optimum Q
"Optically-sound" network arch.
Network layer
IO requests
Application layer
Cross-layer iterative optimization
Network performance/costs trade-offs
Application needs
Application characteristics
Validated and optimized models of optical networks
Computing platforms (simulators)

**Figure 1: General organization of PhoenixSim.**

Each layer can also be used separately. Calculations and/or optimizations realized at the physical layer can be performed separately from any network simulation. In that case, the resulting link or interconnect figures of merit (bandwidth and power consumption in most cases) are simply recorded for each studied case. Network simulation can be run over "virtual" network architectures disconnected from any hardware reality using classical random traffic generators. Finally, parallel applications can be executed over an ideal communication substrate. Being able to consider a single layer at the time is mandatory for debugging, validation and verification purposes. Single layer studies are also crucial for the user to seize the interdependence of the performance to key parameters. Pairs of layers (physical + network, or network + application) can also be considered.

The graphical user interfaces (GUIs) constitute an important feature of PhoenixSim. A first user interface lets the user select one or more configurations to study. This GUI is totally generic and adapts automatically to new features added to the framework. In other terms, adding a component to PhoenixSim (typically, a network arbitration algorithm) does not signify adding a corresponding element to the interface. The graphical interface is also progressively constructed, as the user takes his first decisions about the use case to model/simulate. A second interface lets the user explore the results collected during the execution of the selected use case(s) (Fig. 3). This interface includes a chart generation engine which lets the user select which parameter should be assigned to the x-axis of the chart, as well as which performance metric to the y-axis. A system of filters allows the display of results corresponding to specific input, intermediate, or output values. This "result exploration interface" allows the user to observe very rapidly and from a variety of angles, the results corresponding to the configured use cases.

PhoenixSim is written in Java, which guarantees portability across computer platforms and operating systems, smooths the installation process and make the realization of ad-hoc visualization tools easier. The generic graphical interfaces are also widely built on top of Java's introspection capabilities.

It is worth noting that PhoenixSim, its current state, is not meant to specifically design circuits and networks – it does not produce hardware integrated circuit descriptions (GDS, LEF) nor VHDL or Verilog definitions. It is not either meant to carry detailed, cycle accurate simulations or to be connected to such simulators. As emphasized in the introduction, our main objective is to have the various aspects of computing systems "meeting in the middle" in order to validate their adequacy, prior to engage the huge design efforts required to finalize a design.

Note also that PhoenixSim is mainly developed to support the experiments conducted by our research group (e.g. [7, 17, 18]) and as such, is not made widely available. Access to the repository can be granted upon request by email, however.

## 3. PHYSICAL LAYER

The PhoenixSim physical layer takes as an input a set of optical paths described in terms of active and passive devices and transmission format. An optical path specifies a collection of devices and the position of each device in a topology. A transmission format simply specifies the signaling rate of each channel and the number of channels employed in a wavelength-division-multiplexed (WDM) link. The methodology implemented in the PhoenixSim physical layer can be summarized in several core operations: it starts by estimating the level of signal deterioration (in terms of power penalty) corresponding to each path and, if applicable, adapt the parameters of the devices to minimize the end-to-end power penalty. Once this end-to-end power penalty known, PhoenixSim identifies the optical power level required at the laser output to guarantee a specified quality level (reliability) of transmission (defined in terms of bit-error-rate of the system). Finally, it derives active and passive power consumptions of every device included in the layout.

### 3.1 Power penalty estimation and optimization

To estimate the power penalty imposed by each device (or group of devices) on the transmitted optical signals along the network, PhoenixSim relies on a collection of compact models [3, 20]. Each device model expresses a power penalty as a function of the device parameters and transmission format.

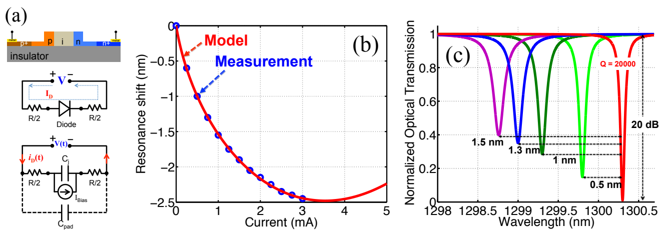


**Figure 2: Modeling of a carrier-injection ring resonator. The key parameters are the DC model and RF model (a), the resonance shift vs current (b), and how the loss (or extinction ratio) and Q of the ring change in accordance with the amount of the shift required (c).**

This power penalty estimation takes into account all forms of attenuations *and distortions* inflicted to the signals. For a Mach-Zehnder Interferometer (MZI) modulator, for example, the power penalty is calculated as a function of the bitrate, the number of wavelengths, but also of the MZI arm length, voltage required to obtain a $\pi$ shift ($V_\pi$), voltage applied to drive the arm (peak-to-peak drive voltage), waveguide propagation loss, and the loss of input and output junctions (typically MMI design). Compact models can be obtained from sets of laboratory measurements. For conditions falling close to the ones where a measurement has been made, power penalties can be predicted by means of interpolations or extrapolations. To cover exhaustively the parameter space, however, models based on physical principles, and validated by measurements, are required. A large portion of the efforts invested in the physical layer thus effectively consists of constructing such analytical models. In particular, we developed detailed models corresponding to ring resonator array-based modulators and demultiplexers, which have been the matter of a detailed publication [3]. Analytical models for crosstalk estimation, and advanced modulation formats (higher-order modulations) with Mach-Zehnder modulators are currently under active development. We stay also highly aware to potential new models based on the state-of-the-art devices that would appear in the literature. Recently, we incorporated into PhoenixSim the detailed modeling of the carrier-injection silicon ring resonators proposed by [20] (shown in Fig. 2), as well as the ring loss model proposed in [9].

Once compact models accepting various parameters are available, specific optimization procedures can be developed on top of them. The optimization of the Q-factor of a ring filter present in a demultiplexer array is chosen as an illustrative example here. The resonance profile of a ring, which eventually characterizes its frequency response, is determined by its length, the loss factor of its waveguide, and by its coupling coefficients. The two latter parameters are generally not subject to optimization, but so is the ring size. Thus, this lets us the possibility to target a specific Q value, which is crucial to optimally mitigate crosstalk effects appearing when Q tend to be too low, and signal truncation effects, resulting from too high Q values. Fig. 3 exemplifies this optimization.

### 3.2 Initial laser power requirements

Once power penalties associated to every device located alongside an optical path have been evaluated, the end-to-end power penalty is calculated. So far, individual power penalties are always considered as independent from each other, which lets us simply sum them up to obtain the end-to-end final value. In the future, more sophisticated power penalty combi-

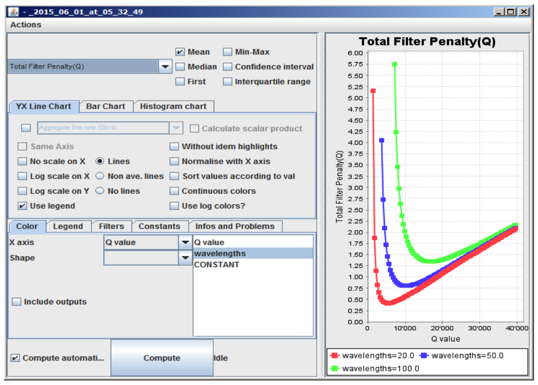


**Figure 3: PhoenixSim GUI showing the Q-factor optimization of ring demultiplexers by minimizing the total power penalty of the demux array.**

nation techniques, typically taking into account the sequence of devices along the link [15], might be developed.

The power penalty is one ingredient to the calculation of the required optical input power. The other is the sensitivity level of the receiver front-end (photodetector + TIA). Here, we also developed a compact model to express the receiver sensitivity as function of the bitrate (more details available in [3]). By summing these two values, the optical power required per wavelength is obtained. By multiplying this by the number of channels, the total laser power is obtained. At this point, the feasibility (in terms of optical signal quality) of the path layout under the considered transmission format is assessed. If the total required input power exceeds a threshold value, provided as an input parameter and representing the power level above which non-linear effects become noticeable, the layout is marked as invalid and the evaluation is halted.

## 3.3 Power consumption

Once the required input laser power level is known, the overall power consumption profile can be determined. Every device deployed is interrogated for three power consumption values corresponding to three possible states: inactive, active but no data, active with data. Pre-determined as well as optimized parameters are included in the power calculations: typically, the consumption of a modulator depends on the applied voltage, which can be subject to optimization.

Some power consumptions are based on measurements reported in the literature. For thermal stabilization of ring resonators, for instance, PhoenixSim considers a (default but configurable) constant consumption of 1mW per ring reported in [12], while for laser consumption, a constant wall-plug efficiency factor is assumed. Other power consumptions are obtained with compact models. For modulators, we consider a "classical" approximation $P \approx 1/4\,CV^2 \times r_b$ [11] where V is the voltage applied to the optical modulator, C is the equivalent RF capacitance and $r_b$ is the signaling rate. For receiver power consumption, we leverage the predictions proposed by [13].

End-to-end power estimations delivered by PhoenixSim can be surprising, as shown in Figure 4a (bold line). We consider the consumption of a simple WDM point-to-point link with different number of channels at a rate of 10 Gbps/$\lambda$. A ring resonator array is used at the transmitter side, while an array of ring resonators demultiplex the WDM signal at the link end. We assume the channels to be equally spaced in the 1525-1575nm range (50 nm spectrum in the C-band). The power consumption per wavelength generally increases with

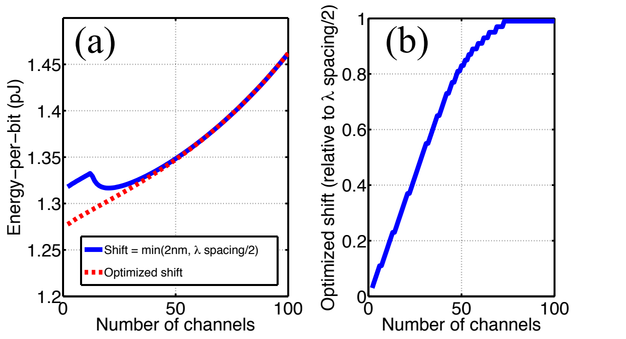


**Figure 4: Effect of ring modulator shift on energy at a rate of 10 Gbps/$\lambda$.**

bandwidth density, reflecting increasing amounts of crosstalk (whose inflicted power penalties must be compensated by extra laser power). However, one notes a "bump" for the lowest number of wavelengths. This bump is explained by the fact that modulator voltage is calculated to achieve wavelength shifts of min(2nm, 50nm / $\lambda$). The reason behind limiting the shift of resonance to a maximum of 2 nm is clear from Fig. 2(b). By injecting high currents (more than 3 mA), the red-shift due to generated heat will dominate over the desired blue-shift generated by the plasma-dispersion effect. Such shifts guarantee ideal optical properties but at the expense of an increased power consumption (itself due to the higher voltages at play [20]). The dotted line of Figure 4a, in contrast, shows the power consumption corresponding to optimized modulator shifts relative to the half of channel spacing (whose value is shown on Figure 4b). This example of device parameter selection for optimized overall power consumption exhibits the need for modeling and optimization tools able to realize system wide analysis.

## 4. NETWORK LAYER

The network layer is mainly meant to evaluate how messages sent over an interconnect are delayed, and how they influence the power consumption. The core of the network layer is therefore composed of an event driven network simulator. Besides the simulator core, PhoenixSim provides a collection of standard simulation components (buffers, routers, packetizers), but also simulation model "builders" which translate simulation input parameters in a collection of simulation objects able to exchange events. Some of these builders integrate physical layer calculations to determine reference data such as link bandwidths. Builders, when initializing the simulation model, create as many traffic sources and receivers as clients. A traffic source can be a random traffic generator, a trace traffic generator (reproducing packet sequences recorded in real systems or in cycle accurate simulators), or can connect to the application layer. Every source records the time at which packets enter the network. Similarly, receivers report packet arrival times. If the application layer is used on top of network layer, receivers also notify application threads waiting for message arrival, if any.

Three main types of networking paradigms have been implemented so far: packet oriented with random access buffers, circuit oriented with distributed arbitration and circuit oriented with centralized arbitration. The first paradigm mimics the behavior of standard electrical packet switches. It is mainly present to provide a point of comparison with traditional networking. Within the second paradigm (circuit oriented with distribution arbitration), clients send request messages to their entry switch, which may or may not forward

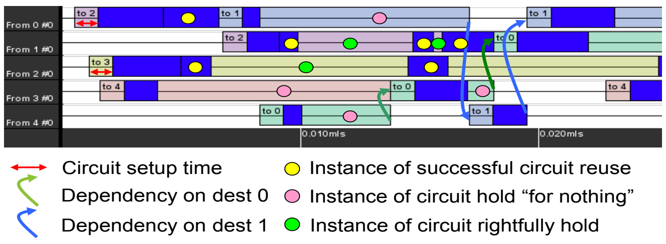


**Figure 5: Example of circuit maintenance. Each line represents the activity of one emitter. Shaded boxes indicate the presence of a circuit (each destination being associated with a tone), filled boxes the presence of a message over the circuit.**

it to the next switch (or to the final destination) depending on path availability. The circuit route is determined in the process. In case of path unavailability, request forwarding (and routing) is interrupted and a negative acknowledgment (NACK) is sent back. Upon NACK reception, the client may immediately resend a request, wait some time, or send a request to another destination [18]. In the third paradigm, the existence of a central arbiter is assumed. Here, the emphasis is put on the management of the circuits over time. Circuits are requested to the arbiter upon reception of a message in the input queue, circuit setup starts as soon as granted by the arbiter, and circuits are utilized as soon as setup is done. However, circuits may be maintained once all messages present in the corresponding queue have been transmitted. In this way, the circuit request and setup process can be potentially saved for future messages destined to the same client [17]. Fig. 5 exemplifies this circuit maintenance concept. Circuits may also be prefetched in advance if the arrival of a message geared to an unconnected yet client is predicted [19]. Predictions can be based on history analysis, or on "hints" provided by the network clients. Note that until now, the switch topology carrying the circuits is assumed strictly non-blocking. In other terms, a circuit can always be routed between two clients as long as these clients have available input and output ports. Extending support to arbitrary topologies counts among our future plans.

## 5. APPLICATION LAYER

In contrast to the traffic present in telecommunication links, which tend to be vastly averaged by successive layers of aggregation, interconnect traffic in (distributed) computing systems is highly application specific. Furthermore, there is clear interdependence between the network and the executed distributed application: delayed packet may slow the application down, and thus the emission of future packets. There is therefore a fundamental need for capturing at least the main aspects of the application behavior when conducting performance analyses of distributed system interconnects.

To address this requirement, the application layer lets the user create simple parallel programs (skeletons). These skeletons are executed concurrently to the network simulator, but above all interact with it via calls such as send, *blockingReceive*, or *nonBlockingReceive*.

The code listed in Fig. 6 is taken as an example within which each of the four ranks (i.e. parallel processes) i) executes a serial task, ii) sends a message to the next rank, iii) waits until the message from the previous rank is received, iv) executes a second task and v) terminates. The task durations and message sizes are made different for each rank. Time representations of the execution of this example skeleton are depicted in Fig. 7a and 7b. A central arbitration, circuit-oriented network is assumed. Network link bandwidth is simply assumed to be 10Gbps, while circuit setup time of 100ns (7a) and 1$\mu$s (7b) are considered. The difference in setup time induces different start times for the second tasks, and thus different skeleton total execution times (also called time-to-solution).

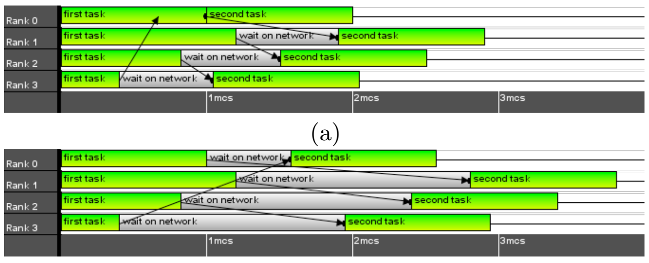
```
@Override
public void runImpl(ActionManager c, int rank, Time timeRef)
                throws InterruptedException {

    int[] jobDurations = {1000, 1200, 820, 400};
    int[] packetSizes = {8000, 2000, 1200, 1700};

    c.doSomeJob(timeRef, jobDurations[rank], "first task");
    c.send(timeRef, packetSizes[rank], (rank + 1)% 4);
    c.blockingReadFromAny(timeRef);
    c.doSomeJob(timeRef, 1000, "second task");

    this.executionEnd(rank, timeRef);
}
```

**Figure 6: Example of application skeleton.**



(a)

(b)

**Figure 7: Timeline representation of skeleton listed in Fig. 6 over 10 Gbps circuits with 100ns (a) and 1 $\mu$s (b) setup time.**

The application layer provides a collection of other commands permitting to:

- wait for *one* message from a specific origin, a subset of origins, all origins but one, or any origin.
- wait for *a set of* messages from a set of origins, all origins but one (generally itself) or all origins – reduce operations.
- tentatively receive message in a non-blocking fashion (i.e. the thread continues even if no message has arrived)
- perform broadcast operations
- define the thread state as *idle* for a determined amount of time

In general, the application layer thus mimics a message-passing based programming environment using interfaces similar to those of MPI.

## 6. CROSS-LAYER MODELING EXAMPLE

PhoenixSim finally allows us to investigate how these layers interact and to achieve multi-layer simulation data. To that aim, we consider, as an example, the design and analysis of a wavelength routed optical network interconnecting N compute nodes. At the physical layer, each node is equipped with a tunable laser associated with an MZI modulator whose output is coupled to an optical fiber. These N input fibers connect to an optical multiplexer, directly followed by a wavelength demultiplexer, whose outputs connect to the receivers through N output fibers. Each node is thus connected to a specific output of the wavelength demultiplexer, and associated to a specific wavelength. To send a message to a given destination, a transmitter tunes its laser to the corresponding wavelength. The bandwidth made available to each node (both for

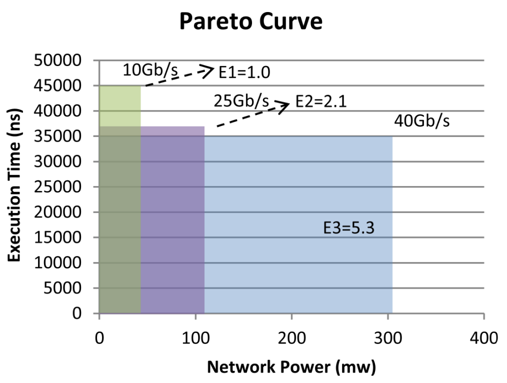


**Figure 8: Time-power Pareto curve of HPCG on the example tunable laser based system with various modulation rates. E1 to E3 in the right figure show the normalized network energy to solution under different modulation rates.**

transmission and reception) is thus equal to the wavelength modulation rate.

We assume this architecture to be circuit arbitrated. That is, as the network is non-blocking, the above-presented circuit maintained paradigm applies here. Finally, the connected compute nodes are assumed to execute in parallel an application skeleton of the High Performance Conjugate Gradient (HPCG) benchmark [1,5]. HPCG is a benchmark that generates and solves a synthetic 3D sparse linear system using a local symmetric Gauss-Seidel preconditioned conjugate gradient method. Although in practice compute nodes may integrate multiple processors, we assume here for example purpose that each compute node integrates a single processor capable of 100 GigaFLOPS.

We aim to find the answer to the following question in this example experiment: what speed should the modulators operate at. Fundamentally, this speed should be maximized to deliver the highest performance. However, one easily notes that the modulation speed is closely related to the power consumption. Thus, the goal becomes to analyze which trade-off should be achieved between these two dimensions. Fig. 8 shows the execution time and network power dissipation of HPCG (nx, ny, nz = 8) in PhoenixSim on a four-node system corresponding to various modulation speeds (10, 25 and 40 Gbps). The simulation assumes a fast tunable laser with 5ns wavelength switching latency [16] and 5% laser wall-plug efficiency. As expected, the execution time decreases with the modulation speed. However, as the modulation speed increases, the power consumption of the network increases as well. Overall, results of this kind show that optimizing for pure performance or rather for pure energy consumption leads to very different designs. Fig. 9 shows partial timeline representation of HPCG in the case of 10 Gbps modulation speed.

## 7. CONCLUSIONS

We presented PhoenixSim, a platform devoted to conduct cross-layer analyses and designs of optical interconnects for distributed computing systems. PhoenixSim aims at realizing early design space explorations spanning over physical layer devices, network layer architectures and application-layer designs. It is meant to establish an iterative closed-loop between all the aspects of photonic architecture development, from optical link optimization to photonic enabled application designs. This holistic approach can potentially save design cycles spent at, for example, over-optimizing a physical-layer

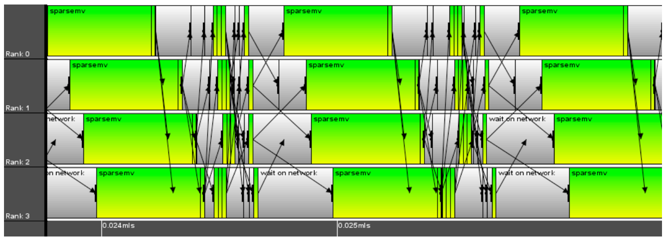


**Figure 9: Partial timeline representation of HPCG running on the example tunable laser based system with 10 Gbps modulation rate.**

parameter that provides limited benefits to the application layer, or studying in many details a network architecture that is unrealistic from a physical layer point of view.

## Acknowledgment

This work is supported by DARPA Microsystems Technology Office (MTO) under the COEDM (Computing with Optically Enabled Data Movement) project, and by the Department of Energy (DoE) ASCR project ”Data Movement Dominates” through contract PO1426332 with Sandia National Laboratories.

## 8. REFERENCES

[1] https://software.sandia.gov/hpcg/html/index.html.
[2] S. Assefa, et al. A 90nm CMOS integrated nano-photonics technology for 25Gbps WDM optical communications applications. In *IEEE International Electron Devices Meeting (IEDM)*, (33), 2012.
[3] M. Bahadori, et al. Comprehensive design space exploration of silicon photonic interconnects. In *IEEE Journal of Lightwave Technology*, in press, 2016.
[4] J. Chan, et al. PhoenixSim: A simulator for physical-layer analysis of chip-scale photonic interconnection networks. *DATE*, 2010.
[5] J. Dongarra, et al. High-performance conjugate-gradient benchmark: A new metric for ranking high-performance computing systems. *International Journal of High Performance Computing Applications*, 2015.
[6] M. Glick, et al. Modeling and simulation environment for photonic interconnection networks in high performance computing. *ICTON*, 2013.
[7] R. Hendry, et al. Physical layer analysis and modeling of silicon photonic WDM bus architectures. *SiPhotonics workshop @ HiPEAC*, 2014.
[8] M. Hochberg and T. Baehr-Jones. Towards fabless silicon photonics. *Nature Photonics*, 4(8):492–494, 2010.
[9] H. Jayatilleka, et al. Crosstalk in SOI microring resonator-based filters. *Optical Interconnects.*, 2015.
[10] Y. Liu, et al. Ultra-compact 320 Gb/s and 160 Gb/s WDM transmitters based on silicon microrings. *OFC*, 2014.
[11] D. A. Miller. Energy consumption in optical modulators for interconnects. *Optics express*, 20(102):A293–A308, 2012.
[12] K. Padmaraju, et al. Integrated thermal stabilization of a microring modulator. *Optics express*, 21(12):14342–14350, 2013.
[13] R. Polster. Architecture of silicon photonic links. *PhD Thesis, Stits doctoral school*, 2015.
[14] R. Polster, et al. Tia optimization for optical network receivers for multi-core systems-in-package. *PRIME conference*, 2014.
[15] S. Rumley, et al. Silicon photonics for exascale systems. *Journal of Lightwave Technology*, 33(3):547–562, 2015.
[16] J. E. Simsarian, et al. Less than 5-ns wavelength switching with an SG-DBR laser. *IEEE photonics technology letters*, 18(1-4):565–567, 2006.
[17] K. Wen, et al. Reuse distance based circuit replacement in silicon photonic interconnection networks for HPC. *HOTI*, 2014.
[18] K. Wen, et al. Reducing energy per delivered bit in silicon photonic interconnection networks. *Optical Interconnects conf.*, 2014.
[19] K. Wen, et al. Latency-avoiding dynamic optical circuit prefetching using application-specific predictors. *Exacomm*, 2015.
[20] R. Wu, et al. Compact modeling and system implications of microring modulators in nanophotonic interconnects. *ACM/IEEE SLIP*, 2015.