Architectures and Design Automation for Photonic Networks On Chip

Gilbert R. Hendry

Submitted in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy
in the Graduate School of Arts and Sciences

Columbia University

2011

© 2011
Gilbert Hendry
All rights reserved.

Abstract

Architectures and Design Automation for Photonic Networks-on-Chip
Gilbert R. Hendry

Chip-scale photonics has emerged as an exciting field which can potentially solve many of the problems plaguing the high-performance computing industry, from large-scale to embedded. In theory, photonics is a superior communication medium because of its higher bandwidth density using wave-division multiplexing and bandwidth-power translucency to distance traveled. In practice, physical-layer design and engineering issues such as optical loss, crosstalk, and packaging have slowed its entry into widespread adoption at the chip and board scale. In this work, we present these issues and potential design improvements. The major contributions, however, are the tools and methods we have developed for the design of photonic interconnection networks, including a system-level simulator and CAD and modeling environment for layout, both of which are publicly available to the research community.

Contents

List of Figures

List of Tables

1

Introduction

THE past few decades have been an interesting time for computers. Incredible exponential growth in the complexity of both hardware and software of computers, as well as the amount of information available to us, has changed the way we do everything, every day. Most of this advancement comes from Gordon Moore's (possibly self-fulfilling) prophecy that the number of transistors double every 18 months. However, we could be quickly approaching the inflection point of progress.

As we start to see chips using 22nm technology and below, it is becoming apparent that transistors will not scale forever. More than a decade ago, in the late 1990's, engineers were playing new tricks to squeeze out more resolution from their 193nm photo-lithography fabs (2). Transistors will continue to get smaller from here, and many companies have at least 15nm, if not 11nm on their horizon. But where do we go from there? If trends continue, features will be a few molecules or atoms wide, incurring serious quantum effects.

One direction is up, or 3D stacking. Besides the architectural advantages of being able to place more compute power and memory in a single package, designers benefit from 3D stacking by being able to keep logically different or fabrication-sensitive parts separate while optimizing each layer. Key challenges are now being solved for 3D stacked chips, including power and signal delivery with TSVs (3), mechanical issues such as thermally-induced stress (4), and micro-fluidic cooling using vertical pipes (5). However, 3D stacking by bonding separate layers in post-processing is a relatively costly procedure. As we will see in later chapters, deposited silicon photonics is an excellent solution for this.

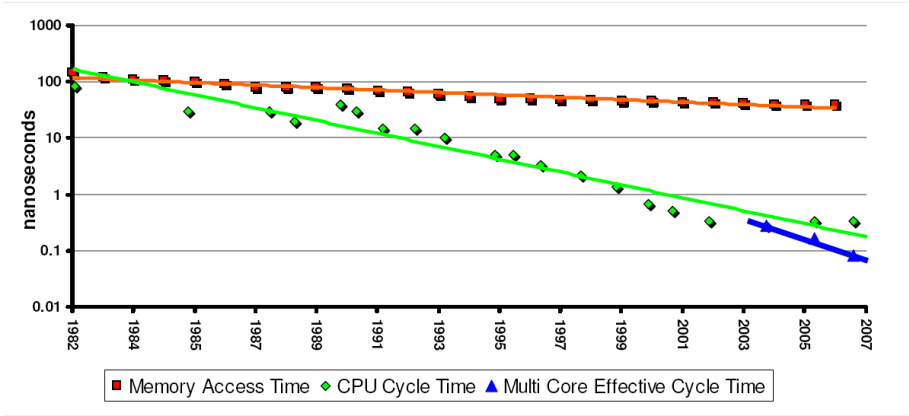


Figure 1.1: Divergence of CPU power from memory access time, from Exascale report (1).

1.1 The Memory Wall

New kinds of CMOS-type devices are also being developed. Multi-gate transistors have been in development for a few years (6) which will indeed provide more transistor density, and therefore compute and storage power on chip. However, we already know we can't make a single processing unit significantly better while meeting power constraints, so parallelism has become king in the architecture world. Provided that there is enough work to be done, or enough data to crunch through for the processor, the idea is that exploiting parallelism will be the only way to increase performance.

But parallelism necessarily implies communication. Neither 3D-stacking nor transistor redesign effectively address communication challenges on-chip. Some sources say as much as 50% of the power on chip is caused by the interconnect (7). However, what is more important is off-chip communication.

The first challenge in chip I/O is scaling frequency of the links. Traditionally, this has been difficult because of the careful design of the drivers, receivers, and matched PCB paths. This struggle has led to what is usually referred to as the memory wall (or gap), which can be seen in Figure 1.1. While dramatic increases in on-chip frequencies and transistor densities have yielded huge performance gains over the last decades, this has not necessarily been the case for DRAM and other IO.

The second challenge for electronics is that package IO pin count is already becoming difficult for ball-grid arrays, and is not expected to progress all that much (8). For example, the TILE64 (9) has a couple thousand total package pins, and approximately 1000 data IO pins. According to the ITRS, this is already at the max pin-count for the cost-performance category (commodity general-purpose

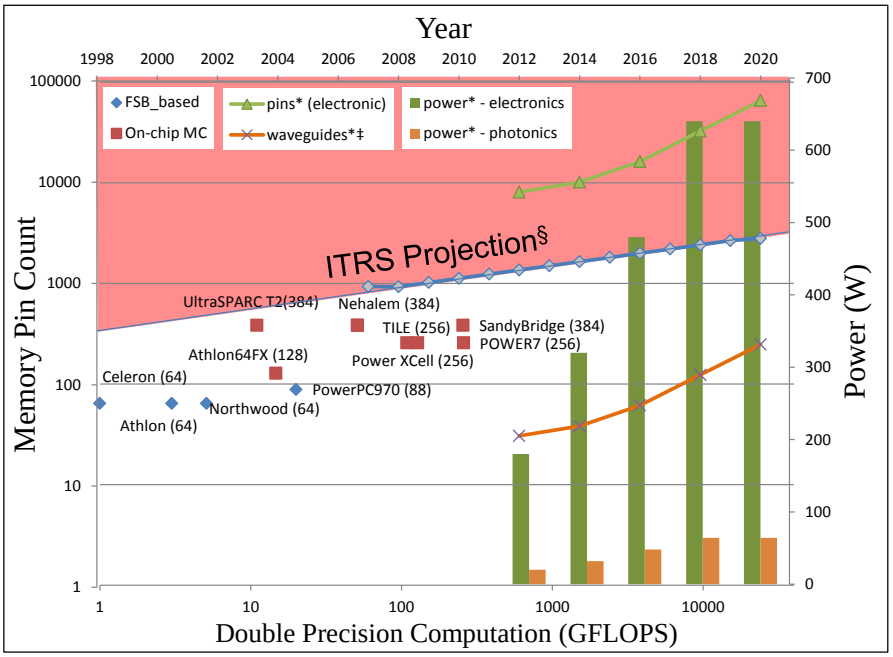


Figure 1.2: Pin counts and power for today's microprocessors and projected into the near future.

microprocessors) (8).

Figure 1.2 shows some interesting data from today's microprocessors. The red and blue data points show the number of package pins dedicated to DRAM versus the computation power and year of introduction. Clearly, this number has not increased very much over the last decade for both chips using front-side buses (FSBs) and on-chip memory controllers. This is one cause of the memory wall by limiting bandwidth and connectivity.

Figure 1.2 also shows some projections for power and number of pins for a 10 TFLOP processor with 1 B/FLOP of memory bandwidth. Clearly, the number of pins rises well into the infeasible range, and power becomes unmanageable. What this figure portrays overall, is that there is simply no more

*Hypothetical system, reaching 10 TFLOPS in 2016. Assumes doubled performance every 2 years, 1 B/FLOP required memory bandwidth, scaled bitrate from 5 (2012) to 10 Gb/s (2016). Power scales from 10 to 2 pJ/bit for electronics, 1.2 to 0.2 pJ/bit for photonics.

‡Assuming 128 wavelengths WDM

§This line is obtained from the bigger of the two values of the Package Pin Count Maximum projection for Cost-Performance microprocessors reported in the 2008 ITRS and dividing it by 3 (based on the assumption that 1/3 of the pins of the total package pin count is dedicated to communication with memory.)

room to put more electronic pins on a single package, so despite having more computation power on chip through transistor density, frequency, and number of parallel cores, the memory subsystem will still be limited by the same bandwidth of previous generations.

1.2 Exascale Computing

In 2008, a study was done with leading computing experts on how to achieve an Exascale computer (1). One of their main conclusions was that continuing on the path of today's programming models, execution models, computer architecture, and device technology was not sufficient due to power limitations and feasibility. A target power budget of 20 MW, more than which cannot be sustained financially (10), has been cited by the Department of Energy to realize an Exascale machine, which is 500 times faster than the 6 MW petascale machines of today. Clearly, this will require drastic changes in the way the machine is designed and implemented, and programs were spun off to try to seriously rethink the computer from all sides. Among all the issues, communication was identified as a key challenge.

The Exascale problem represents a unique challenge to the computing community, and therefore a unique opportunity for photonics. The problem poses a very real and constrained situation which will *force* engineers to consider new technologies such as photonics in a way that has not been necessary before. And, advances achieved in the area of high-performance and large-scale computing will also be applicable to other areas such as commercial or embedded systems.

From the application side, many people believe the user (programmer) *must* have some explicit control over data movement in a high-performance machine, or at the very least, make it so the programmer can't *not* be aware of data movement. This belief basically stems from the realization that implicit shared memory paradigms cannot scale efficiently. Though we may not necessarily see programs where chip-scale communication is *all* MPI-style explicit communication, programming/memory models such as partitioned global address space (PGAS) (11) will help to add data locality to programming and program execution.

1.3 The Role of Photonics

Optical communication has been around for some time in the internet backbone for long-haul transmission across the country. The idea is that optics is not as distance-dependent as electronics for both power and bandwidth because it can go much longer without have to repeat the signal. Latency benefits can also be argued, and higher bitrates are more feasible when they don't have to be repeated so often. In theory, optical transmission is very appealing for any scale of communication.

Steadily, optics has made its way down into smaller-scale and shorter-distance applications, the first being supercomputers and compute clusters. The lighter, longer, more compact form factor of optical fiber alone provides a benefit for machines with *many* connections to a compute rack. However, most of these solutions are active-cabling, which replaces electrical cables with plug-and-play fibers with transceivers at both ends. Though making use of optical technology in this way is itself interesting, many people in photonic networks research believe that we can go further and perform switching or routing optically as well. Electronic networks are typically built on a store-and-forward mechanism, where data packets are stored, routed, and arbitrated at each router in a network. By bypassing electronic switching and routing, optics as the potential to work outside the power, latency, and bandwidth restraints of the store-and-forward mechanism (see Figure 3.1 in Chapter 3). Methods using both MEMS-based circuit-switch (12) and optical packet switching (OPS) (13) have been proposed, though they have not made it into commercial systems yet.

There are many good reasons why photonics is also appealing for chip-scale communications. First, as we mentioned, photonics can provide immense IO bandwidth. For example, instead of the TILE64's (9) 1000 IO pins, photonics could use 100 waveguides (with the same pitch) using 10 wavelengths each, with higher signaling rate. As chip and IO area become more constrained, photonics is appealing because of either reduced packaging complexity or increased IO bandwidth.

Another reason why photonics is looking good is that chip-scale nanophotonics is maturing. The basic components of an on-chip link have been put together and demonstrated (14), and commercial silicon nanophotonics is becoming available today for board- and computer-scale communications (15). Though ring resonator based structures are not yet ready for production, the challenges for making them reliable are being solved right now.

The questions we are presented with now are ”how do we enable large scale integration of photonic devices?” and, once that is addressed, ”how do we make the best use of photonics?” This thesis presents design methodologies and tools necessary to investigate these questions. Specifically, a system-level simulator called PhoenixSim was developed for the exploration of both electronic and photonic networks on chip, which is freely available to the research community. A layout CAD tool called VANDAL was also developed targeted towards designing, modeling, and fabricating chip-scale optical links and networks. Finally, using these tools, improvements and design spaces were explored for different network architectures targeting multi-core CMPs to highly specific SoCs.

In Chapter 2, we provide a background of the photonic devices that are under investigation today, describing their functional and physical behavior and characteristics. In Chapter 3, we present network architecture designs and improvements, and considering arising issues when analyzing the collective effects of the physical layer both statically and at runtime. Finally, in Chapter 4, we present the simulation, optimization, and automation tools that are necessary to perform the design and analysis from Chapter 3. It is our ultimate aim to be able to merge the physical geometry and characteristics of photonic devices with architectural and system-level design practices, bringing the reality of chip-scale nanophotonics one step closer.

2

Silicon Photonics

In this chapter, we will present an overview of some of the nanophotonic devices and structures that are common in today's research, and which will be used later in architectural designs as well as layout and synthesis. The presentation of this material is not meant to imply that any of it is novel contribution to the field nor is it an exhaustive or in-depth analysis, rather it is background material for understanding later chapters for readers who are not necessarily familiar with photonics. We will present devices as functional components and describe their geometry and modeling, in some cases for both crystalline silicon and deposited materials.

2.1 WDM Photonic Link

First, we will briefly describe how photonic communication works in general. Figure 2.1 shows the general structure of a Wave Division Multiplexed (WDM) link. The term WDM refers to the fact that multiple frequencies of light can *each* carry information simultaneously in the same waveguide at the same time. This is advantageous because it dramatically increases the bandwidth density (bandwidth per unit area) of a link. Of course, to properly use WDM, we require devices which can perform functions on individual frequencies in a waveguide without significantly affecting the others, which will be introduced later. Following the numbered items in Figure 2.1:

1. Data arrives to the send side of the link, which must be ramped up to the modulation clock rate, known as serialization. This stage requires buffers and circuitry which can convert between two

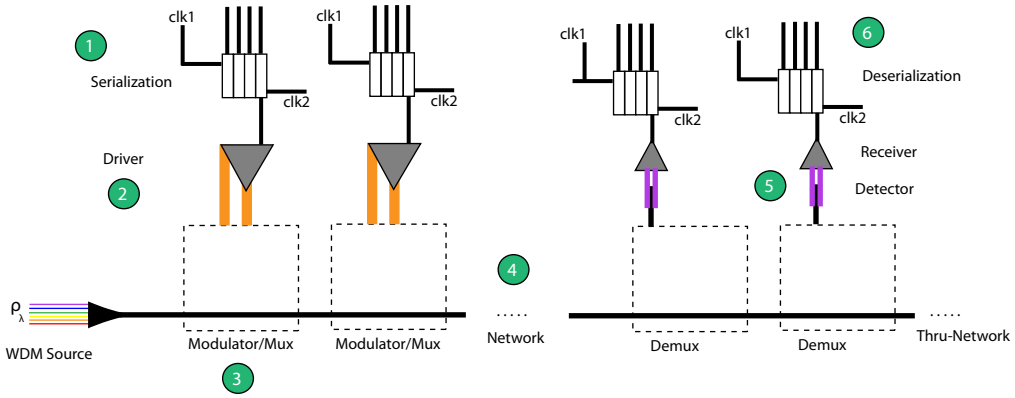


Figure 2.1: Basic form of a WDM link.

clock domains.

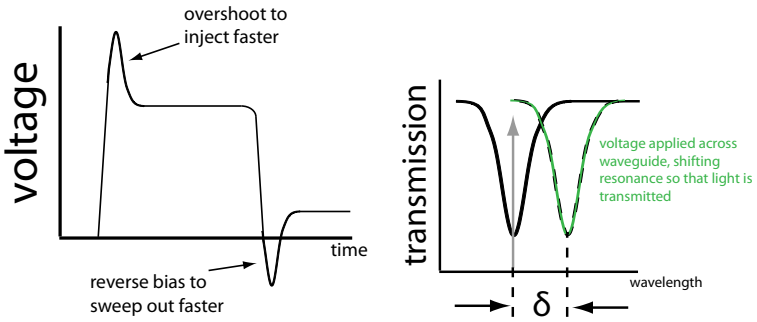
2. Analog circuitry drives the ones and zeros into the modulator, consisting of some amplifiers and wave-shaping circuits. One technique implemented in the driver circuitry, shown in Figure 2.2(a), overshoots the applied voltage to inject carriers quicker, and under-shoots the zero-voltage to sweep out carriers for faster modulation.

3. Modulators convert continuous-wave light of a specific frequency into light which carries the digital information. Many modulators arranged serially along a waveguide can operate each on their own wavelength in parallel. Usually, the continuous-wave source of light comes from lasers off-chip which are multiplexed together and launched into a waveguide. The simplest way to accomplish modulation is to use on-off keying (OOK), or blocking light for zeros, and letting light through for ones, which is shown in Figure 2.2(b).

4. A network transparently switches or routes the information using filters or active switches, either wavelength-dependent or broadband.

5. Each wavelength is filtered out, and arrives to a detector which can absorb light, producing a current. The receiver converts the current to a voltage and amplifies it up to a level which works with the digital circuitry. If some wavelengths are not filtered out and detected, they continue on to other parts of the network.

6. The data is ramped back down to the clock rate it started in, known as deserialization.



(a) Driver Enhancements (b) On-Off Keying

Figure 2.2: Modulation operation.
(a) to get a cleaner signal at high bitrates, we must apply wave-shaping circuits to the driver to sweep carriers in and out of the modulator. (b) On-Off Keying is sometimes achieved by shifting the frequency that the modulator operates on while keeping the laser source constant, which will block light for zeros, let light through for ones.

Of course, this is a simplified view of what happens. In reality many signaling issues come into play which affect communication performance such as noise, filter quality, and optical loss. But, keeping this basic structure and interoperation of components in mind, we can introduce some of the devices which have been developed to perform these functions.

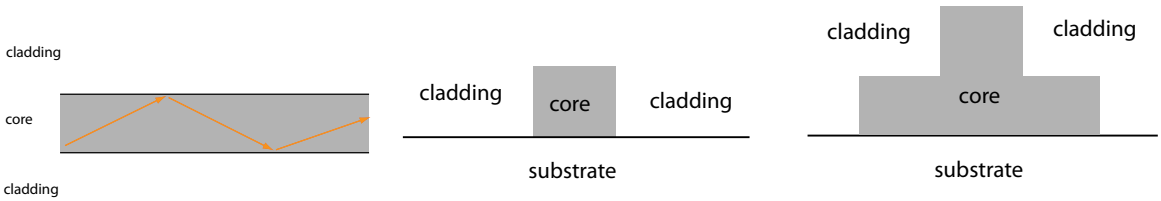
2.2 Simple Structures

This section describes some of the more simple integrated photonic structures and devices. Here, simple does not mean that they are easy to fabricate or model necessarily, but rather have fairly simple geometric structure, and may have fairly limited use for our purposes.

2.2.1 Waveguides

The fundamental unit of light transport on chip is the waveguide. Guiding light is fairly easy to understand: light is confined by a refractive index contrast in a narrow shaft. Figure 2.3 shows a basic diagram. In Figure 2.3(a), we see a top view of a waveguide with light reflecting off the walls. For all the light to be confined (known as *total internal reflection*), it must be incident at a small *critical* angle $\theta_c = arcsin(\frac{n_{cladding}}{n_{core}})$, which is greatly exaggerated in the figure.

Light can also be confined with different structures. Figure 2.3(b) shows the *rib* waveguide, the



(a) Waveguide Top (b) Rib Waveguide (c) Ridge Waveguide

Figure 2.3: Simple view of waveguides.

simplest form. Figure 2.3(c) shows a waveguide with a *ridge*, which is useful for creating a P-I-N diode across the waveguide (as we'll discuss later). Though there are many other ways of doing this, the rib and ridge waveguides are the main structures we will consider in this work to make things simple.

The main way to characterize waveguides is by finding its *effective index*, which describes how light propagates through the waveguide, taking into account both the materials of the core and cladding as well as the geometry of the waveguide. This allows us to use a *propagation constant* ($\beta = n_{eff}\frac{2\pi}{\lambda}$) when describing the propagation of a plane wave. To approximate the effective index, we use the effective index method for rectangular waveguides (16). Briefly, for a cross section of the waveguide, this involves calculating an effective 1-dimensional effective index in one direction first, then using it to find the final effective index in the other dimension. For the ridge waveguide, an average index of cladding and core can be used for the "shoulders" of the structure as an approximation.

Usually, the amount of optical power in a waveguide has a limit before nonlinear effects are produced, which can highly degrade a signal's integrity. However, sometimes nonlinear optical effects are sought after, such as using Four-Wave Mixing for frequency conversion (17), which can be used for wavelength-multicasting (18).

Recent work has shown waveguides which can support many wavelengths simultaneously (19) and can support up to 10 dBm of optical power without nonlinearities, and loss down to 0.3 dB/cm (20). Low-loss silicon nitride waveguides, which can be deposited rather than etched from crystalline silicon, have also been shown at 0.1 dB/m (21).

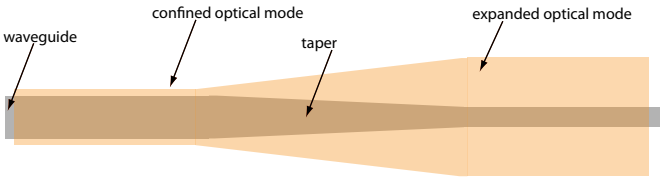


Figure 2.4: Waveguide taper for mode expansion.

2.2.2 Waveguide Taper

A waveguide taper is used to alter the mode profile of confined light in a waveguide. Usually, we design a waveguide such that it is *single-mode*, or most of the optical power is inside the waveguide. Sometimes, we need to have some of the optical power outside the waveguide, such as when we are coupling off-chip to a fiber where single-mode operation is larger, or enhance evanescent coupling between waveguides. Figure 2.4 shows a waveguide taper used for this purpose. As the waveguide gets smaller, the optical mode becomes less confined, and expands out. Expanded out like this, the waveguide will be higher loss, so for longer transmissions we usually want to keep it tightly-confined. This structure has been fabricated in silicon (22), and is commonly used for lateral fiber couplings.

2.2.3 Photo-Detector

One of the most important integrated photonic devices is the photo-detector, or photo-diode, which is used to absorb light, inducing an electrical current. Usually, we then convert electrical current into electrical voltage and amplify using a transimpedance amplifier (TIA). One way to make a photo-detector is to use a single crystal germanium film (which has good absorption in near-infrared) in a metal-semiconductor-metal (MSM) configuration (23), seen in from the top in Figure 2.5. The most important characteristic of a detector is its *responsivity*, or how much electrical current is produced for every unit of optical power introduced into it. This eventually dictates the receiver's *sensitivity*, or the minimum optical power necessary to reliably detect a logical "1". Also, it is important to keep capacitance low for low power operation (24). One characteristic of this detector is that it is broadband (will absorb any wavelength of light in the range of interest), so a wavelength filter must precede it in a WDM system. Section 2.3.4 shows a ring-based detector that can be used for both filtering and absorption.

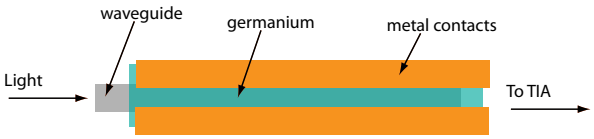


Figure 2.5: Germanium MSM photo-detector

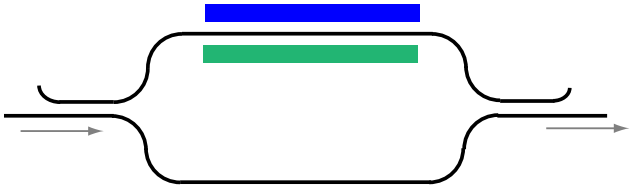


Figure 2.6: Mach-Zehnder interferometer

2.2.4 Mach-Zehnder Interferometer

Another device is the Mach-Zehnder interferometer, shown in Figure 2.6. The idea behind this structure is that the optical power is split at the first evanescently-coupled waveguides, and a phase change is induced in one of the arms. In silicon, this can be accomplished with a p-i-n region around the waveguide. Once the light couples back together at the output, the phase change induced will cause destructive interference (optimally, a shift of π radians), thus canceling the amplitude of the light wave. This device is mainly useful as a modulator (25), and has been shown in the first commercially-available integrated silicon photonics (15). However, the Mach-Zehnder structure is relatively large, requiring a long interaction length (the ”arms”) to induce the phase change. This makes the power consumption and insertion loss high, which makes it not as useful for potential on-chip network designs.

2.3 Ring-based Structures

In this section, we will discuss some of the latest research in micro-ring resonators. Ring-resonator structures have been under investigation long before nano-scale integration was possible. The ring structure creates a sort of feedback effect which ends up making the device resonant with certain frequencies. This resonant frequency also repeats, and the distance between resonances is called the free-spectral range (FSR). The smaller the ring, the longer the FSR. An FSR as large as possible is desired for modulation so as many wavelengths as possible can be used before the resonance repeats (see Section 2.3.1), and as small as possible for a broadband switch for the same reason (see Section 2.3.3). This resonant behavior

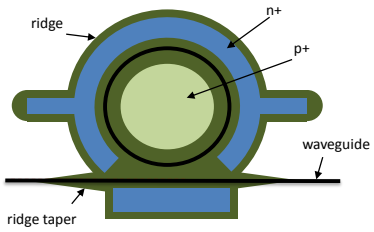


Figure 2.7: Geometry of a ring modulator.

becomes very useful for WDM systems to act on a single wavelength while ignoring any others present in the waveguide. Rings have been a hot topic of research in the past years because they are very small compared to other integrated solutions (as we will point out in the subsections below), which reduces the total area and power needed for a photonic link.

However, there are currently some unsolved challenges to ring-based devices. The first is the extreme temperature sensitivity of the devices. As much as a 1°C change in temperature will change the effective optical path length of the ring (by changing the index of the material), which starts to dramatically shift the resonant frequency. Ring structures are also highly sensitive to manufacturing variations for the same reason, and have very low yields. One way to combat these effects is to include ohmic heaters close to the ring, locally heating the area to stabilize the temperature, though an integrated solution with feedback control has not been demonstrated yet.

2.3.1 Ring Modulator

Modulation, or converting digital bits stored in buffers to information in traveling light, can be done with a ring modulator. The simplest way to accomplish this is to use on-off keying (OOK). Figure 2.7 shows the structure of a ring resonator which can accomplish this for a single wavelength. By applying voltage across the p+-n+ region, we can inject carriers into the waveguide, thus changing the loss (and optical path length) of the ring, which changes the resonant wavelength. Modulation has been shown up to 18Gb/s (26) for this type of modulator. Though we consider mostly chip-scale distances in this work, the basic modulator can even be used to modulate data across very large distances (27). Finally, multiple cascaded modulators for WDM modulation is discussed later in Section3.2.2.4.

One useful way to describe the behavior of the ring modulator is with equations based on the geom-

etry and materials of the device. Yariv did this, assuming a simplified model (28):

$$|b_1|^2 = \frac{\alpha^2 + t^2 - 2\alpha|t|cos(\theta + \phi_t)}{1 + \alpha^2|t|^2 - 2\alpha|t|cos(\theta + \phi_t)} \tag{2.1}$$

where α is the inner circulation factor (the ring's loss), ϕ_t is the phase change of the coupling region, θ is the phase change around the ring ($2\pi N_{eff}(2\pi R)/\lambda$), and t is the transmission coefficient. The optical power coupling coefficient between two waveguides, κ, can be expressed from waveguide and gap dimensions from the Appendix in (29), giving us:

$$\begin{aligned}
\kappa &= \frac{\omega\varepsilon_0 cos(k_{x2}\omega_2)}{2\sqrt{P_1 P_2}(k_{x1}^2 + \alpha_2^2)}(n_1^2 - n_0^2) \\
&\quad \times\sqrt{\frac{\pi R}{\alpha_2}} exp[\alpha(\omega_2 - 2s_0)] \\
&\quad \times[\alpha_2 cos(k_{x1}\omega_1) sinh(\alpha_2\omega_1) \\
&\quad\quad + k_{x1} sinh(k_{x1}\omega 1) cosh(\alpha_2\omega_1)]
\end{aligned}$$

which will give us t by conserving optical power using $t^2 + \kappa^2 = 1$.

We can also construct different types of modulators which have some advantages over the basic ring modulator in Figure 2.7. A second-order structure (using an additional ring) can make the modulator hitless (the resonance disappears instead of shifting), which would reduce crosstalk/loss in a WDM system (30). A different kind of doping region structure can be used to create a p-i-n-i-p instead of a single p-i-n diode for faster carrier injection and depletion (31). Also, a ring coupled with a Mach-Zhender interferometer can make the ring less sensitive to temperature variations (32). Finally, a modulator using deposited polysilicon can be made, which is also very fast (33).

One important thing about on-off keying using a ring modulator is the nonlinear threshold. When the modulator is on-resonance (transmitting no light, or a logical "0"), light is circulating in the resonator building up power. Eventually, it will dissipate due to radiation and scattering losses, but if we inject too much power into it, two-photon absorption (34) and free-carrier absorption (35) will start to happen. Again, this will change the loss in the ring and start to shift the resonance, inadvertently leaking light through. A common nonlinear threshold is about 0 dBm of optical power. One way to possibly combat

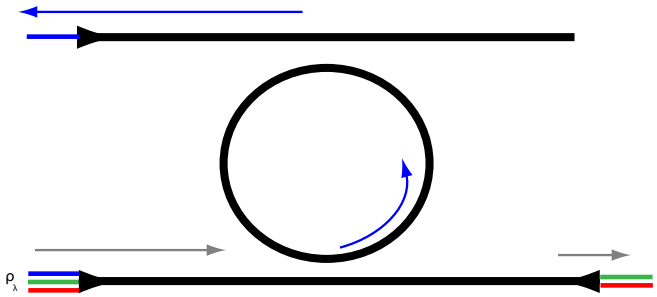


Figure 2.8: Single-wavelength filter

this effect is to either use a drop port to let the optical power out (which may slow it down a little), or to use differential phase-shift keying (DPSK) (36) instead of OOK.

2.3.2 Ring Filter

Another useful function of a ring resonator's wavelength-selective property is to filter out only one wavelength from a WDM waveguide. This is useful when we want to demultiplex wavelengths for detection, as discussed later in Section 3.2.2.4. Figure 2.8 shows this simple structure, filtering out just the blue wavelength. Filters, being one of the easier structures to fabricate because they are passive, have been demonstrated in silicon (37) and deposited silicon nitride (38).

Similarly with the modulator, we can describe the transmission equations for the drop and through port (39):

$$H_{drop} = \frac{\kappa_1^2 \kappa_2^2 \alpha}{1 - 2t_1 t_2 \alpha cos(\theta + \phi_t) + t_1^2 t_2^2 \alpha^2} \tag{2.2}$$

$$H_{thru} = \frac{t_1^2 + t_2^2 \alpha^2 - 2t_1 t_2 \alpha cos(\theta + \phi_t)}{1 - 2t_1 t_2 \alpha cos(\theta + \phi_t) + t_1^2 t_2^2 \alpha^2} \tag{2.3}$$

which becomes very useful when determining channel spacing, and therefore loss and crosstalk.

2.3.3 Ring Switch

If we make the diameter of the ring larger, the FSR startes to become smaller such that the resonant modes can line up with *every* wavelength in the WDM waveguide. We then would have a broadband filter, or, if we put p+ and n+ regions around the waveguide, we can shift the resonance to make a

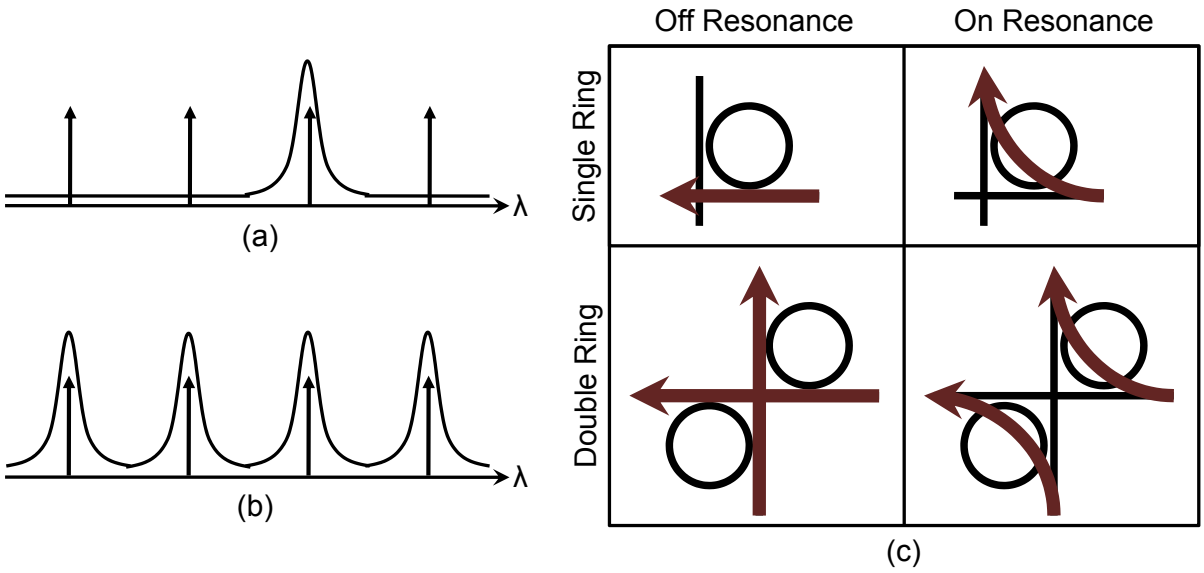


Figure 2.9: Optical ring resonators can be used as switching devices within an on-chip photonic network.

broadband *switch*. Figure 2.9a shows the filter configuration with smaller rings, and Figure 2.9b shows how a ring resonator's resonant profile can be designed to align with many optical wavelengths at once by making the ring larger. We can also implement double switching functionality by using two rings, as in Figure 2.9.

One important difference of the broadband switch from the filter is that because it is bigger, the drop loss will be higher, and it will take more power and take longer for it to switch its resonance. Also, it is probably a good idea to design the switch such that the default state is on-resonance, so that large drop losses are not incurred from passing through a waveguide where carriers are being injected. Broadband switches (or comb-switches) have been demonstrated in silicon with optical pumping (40) and using heaters to thermally switch the rings (41).

2.3.4 Ring Detector

A detector can also be implemented using a ring structure, which acts as both the filtering mechanism and the absorbing medium. One way to do this is by introducing defects into the silicon which enables absorption (42). Another way is to use deposited poly-crystalline for the ring (and waveguide for a good design) (43). In both situations, a p-i-n region is placed around the ring to collect current as shown in Figure 2.10. As of today, the ring-detector structure does not exhibit responsivities as high

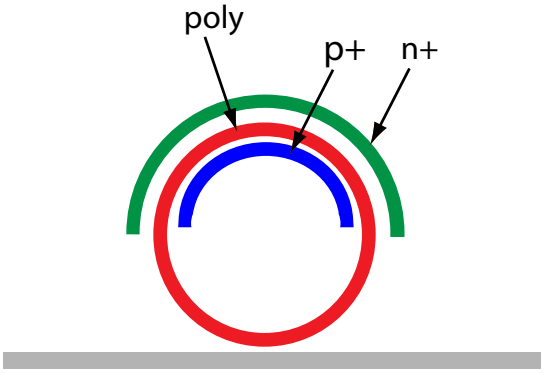


Figure 2.10: Poly ring photo-detector

as germanium-crystal based detectors, though the idea is that because the light circulates many times around the ring, the responsivity should be fairly good because it is subject to the absorbing region many times over.

3

Photonic Interconnection Network Architectures

Now that we have a brief understanding of the state of today's research in nanophotonic devices, we can begin to think about how we will use them in interconnection networks for chip-scale computing.

3.1 Network Architecture Overview

Constructing a photonic interconnection network begins by considering the usual machanisms of any network: arbitration, switching, and routing. First, let us consider how this is accomplished in typical electronic networks. In a classical electronic store-and-forward (SaF) network, data is broken up into *packets*, each of which have a *header* that describes its source, destination, identification, and other useful information. When traversing a network, arbitration, switching and routing can operate on individual packets on a per-router basis. This process is shown in Figure 3.1.

The first step, labeled ①, splits up the data into multiple packets, attaching the appropriate header information to each. When a packet arrives to a router, labeled ②, it is both stored in a buffer and a request is sent to an *arbiter*. The arbiter checks that there is enough available down-stream buffer space to hold the packet (flow control), checks that the input and output ports of the switch are not currently in use (switch allocation), and may optionally assign the packet to a new virtual channel (VC allocation).

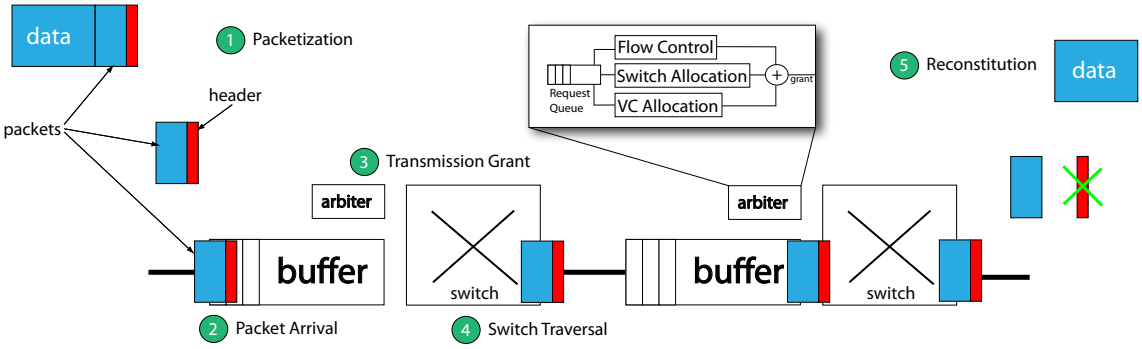


Figure 3.1: The process of store-and-forward packet networks.

After the arbiter performs these checks, it grants the packet access to the switch, and simultaneously informs the switch itself to set up, labeled ③. The packet then traverses through the switch to the next router, labeled ④. At the destination, headers are discarded, and packets are placed in order, labeled ⑤.

Figure 3.2 shows the structure of a typical *packet router*. Typically, a virtual channel multiplexer can route an incoming packet to the correct buffer space within its arriving clock cycle. Transmission requests and grants are sent on the buffer control bus, and packets traverse along the data path wires through the switch. This general router structure along with the SaF paradigm has led to innumerable network and NoC architecture improvements over the years, including wormhole routing, cuthrough routing, and dynamic routing. Techniques for managing buffer space in the network have been proposed for increased utilization, such as credit and token flow control.

However, the fundamental difference of photonics is that while it has benefits for strict communication of bits, it lacks the ability to store bits for an arbitrary amount of time while arbitration and routing can occur. This immediately excludes using the SaF system and general router structure implemented with pure photonics. As we will see in the next sections, there are different ways to either get around this limitation, or use photonics to implement the necessary functions without needing to store packets. These methods need to ensure *end-to-end arbitration*, or allocating a complete path from source to destination before actually transmitting the data. Alternate methods can also be employed which transmit data part-way through the network with photonics, convert to the electronic domain to temporarily store it, then complete the remaining transmission optically. This technique obviously is costly in terms of conversion hardware energy and area, but relaxes some of the arbitration constraints on performance.

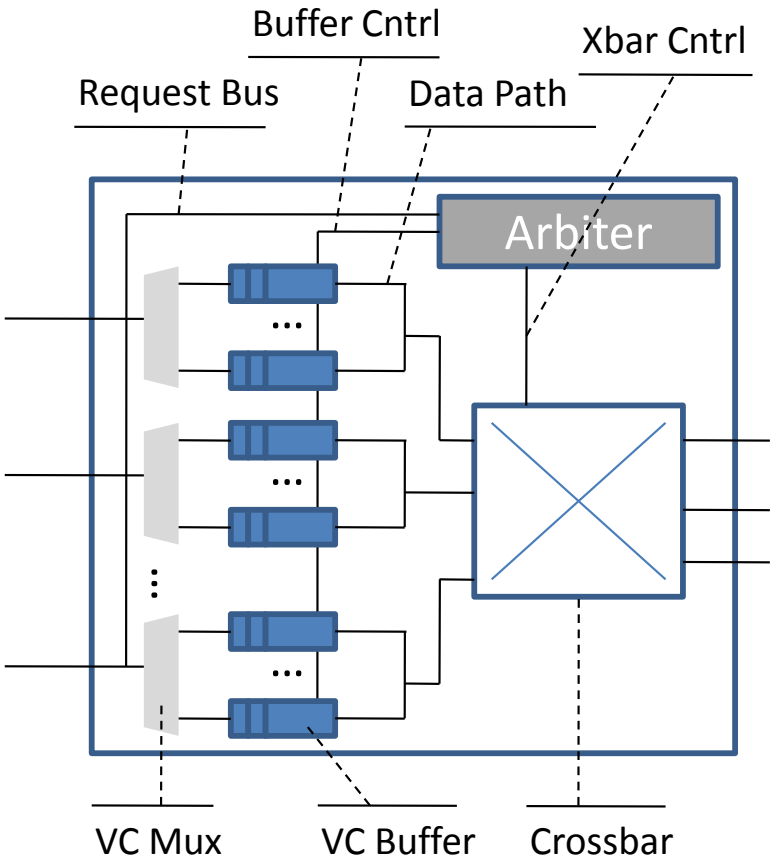


Figure 3.2: Anatomy of an electronic store-and-forward packet router

In any case, the two main classes of photonic interconnection architectures are ones that use circuit-switching, and ones that use the wavelength domain for arbitration. The next sections discuss these classes of architectures in detail, and provide examples of each.

3.2 Circuit-Switched Architectures

One solution to the problem of end-to-end arbitration is to use pure circuit-switching. Circuit-switching, or *spatially*-switched networks use broadband WDM signaling to produce extremely high-bandwidth connections between communicating nodes by multiplexing data onto many parallel wavelengths. Switching is enabled through the use of broadband switches that can actively be controlled to pass or switch all the wavelengths concurrently, discussed earlier in Section 2.3.3. Since this method requires some sort of logical control to activate or de-activate each ring along a path from source to destination, we use

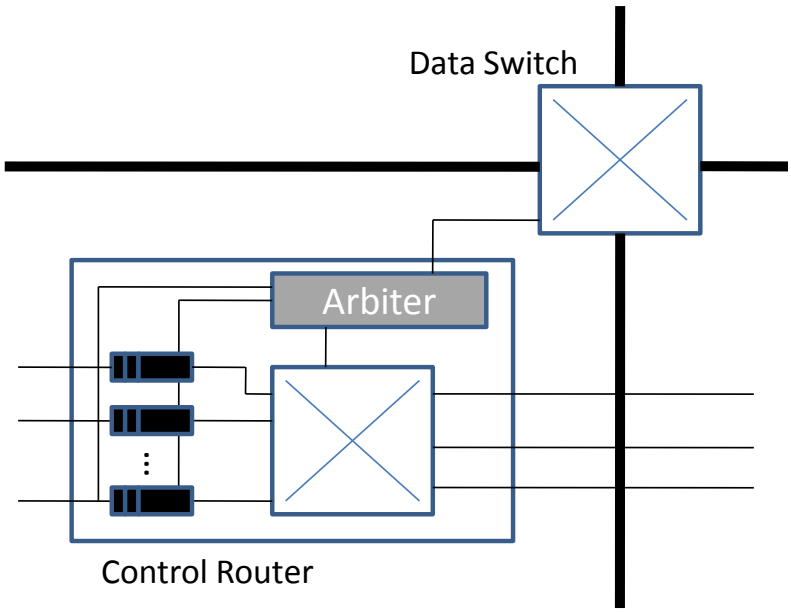


Figure 3.3: Anatomy of a circuit router

a lightweight packet-switched network to arbitrate access to a network of high speed data switches. A path-setup protocol, discussed in Section 3.2.1, is used to set up an end-to-end path through the network *ahead of time*, then transmitting the data when it is complete.

The overall structure of a *circuit router* can be found in Figure 3.3. The electronic *control router* has all the same basic parts of the packet router of Figure 3.2, and functions much the same way. One key difference is that the control router only has to handle 1- or 2-flit control messages, or basically just the header part of a normal data packet. Since the control router does not have to support large flows of data, it should be optimized for latency, and not bandwidth. We call this optimization ”lightweight”, because the channels are narrower, the buffers smaller, and virtual channels are not implemented, all to save power with little cost to performance. The data switch, however, is typically high-speed and high-bandwidth for increased performance. This switch could be implemented with high-speed electronics, though the focus of this book is its implementation with photonic devices.

3.2.1 Path-Setup Protocol

The path-setup protocol is the mechanism that the control routers use to ensure that the data switches in the network are always set correctly, and that two transmissions never conflict with each other. This is

accomplished with the use of four main types of control messages: *path-setup*, *path-ACK*, *path-blocked*, and *path-teardown*. Figure 3.4 shows an example of how the path-setup protocol works.

A network transmission is first initialized, labeled ①, at a network interface (NIF) who wishes to send data to another NIF in the network. After Router_1 receives the path-setup message, besides the usual routing and electronic-plane arbitration, the arbiter performs the additionanal step of checking that resources in its data switch are properly allocated by keeping track of the state of every possible input-outport port combination. After the routing (input-output port pair) is performed and Router_1 checks that the state is *PathFree*, and it is able to forward the path-setup message to Router_2, Router_1 then sets its state from the inport to the outport as *PathReserved*, which indicates that the path is in the process of being set up. The path-setup message continues on to Router_2 where it is blocked, labeled ②.

Blocking could occur if another transmission was either in the process of setting up a path (PathReserved), or already set up (PathSetup), and the input-output combination of the new transmission conflicts with the existing one. Depending on the switch design, this conflict could manifest itself in different ways. If a strictly non-blocking data switch is used, only desintation port conflicts are possible. If the data switch being used exhibits some blocking characteristics, additional blocking scenarios may occur.

Router_2 then turns the message around back to its source, turning it into a *path-blocked* message. This indicates to Router_1 that it should set its state back to PathFree, so that other transmissions may be allowed through. After the sending NIF receives the path-blocked message, it can then immediately retransmit, or implement a *backoff*. Much in the same way that Transmission Control Protocol (TCP) in internet networks uses backoff to manage network congestion when establishing a link, using backoff in the circuit path setup protocol gives the network time to free the contended resources before trying a new setup request to save power and further congestion. A good practice is to have a random chance of immediate retry, with some chance to backoff for a random or predictable amount of time. The setup then continues as normal until it is received by the destination NIF, indicating a successful setup.

The receiving NIF then sends a *path-ACK* message back through the network, which changes the states of each router to *PathSetup*. This is when the arbiter actually sends the appropriate signals to the data switch to implement the correct path, labeled ④. After the sending NIF receives the path-ACK message, it now knows that the end-to-end path is set up through the network, and can begin transmitting data on the data plane.

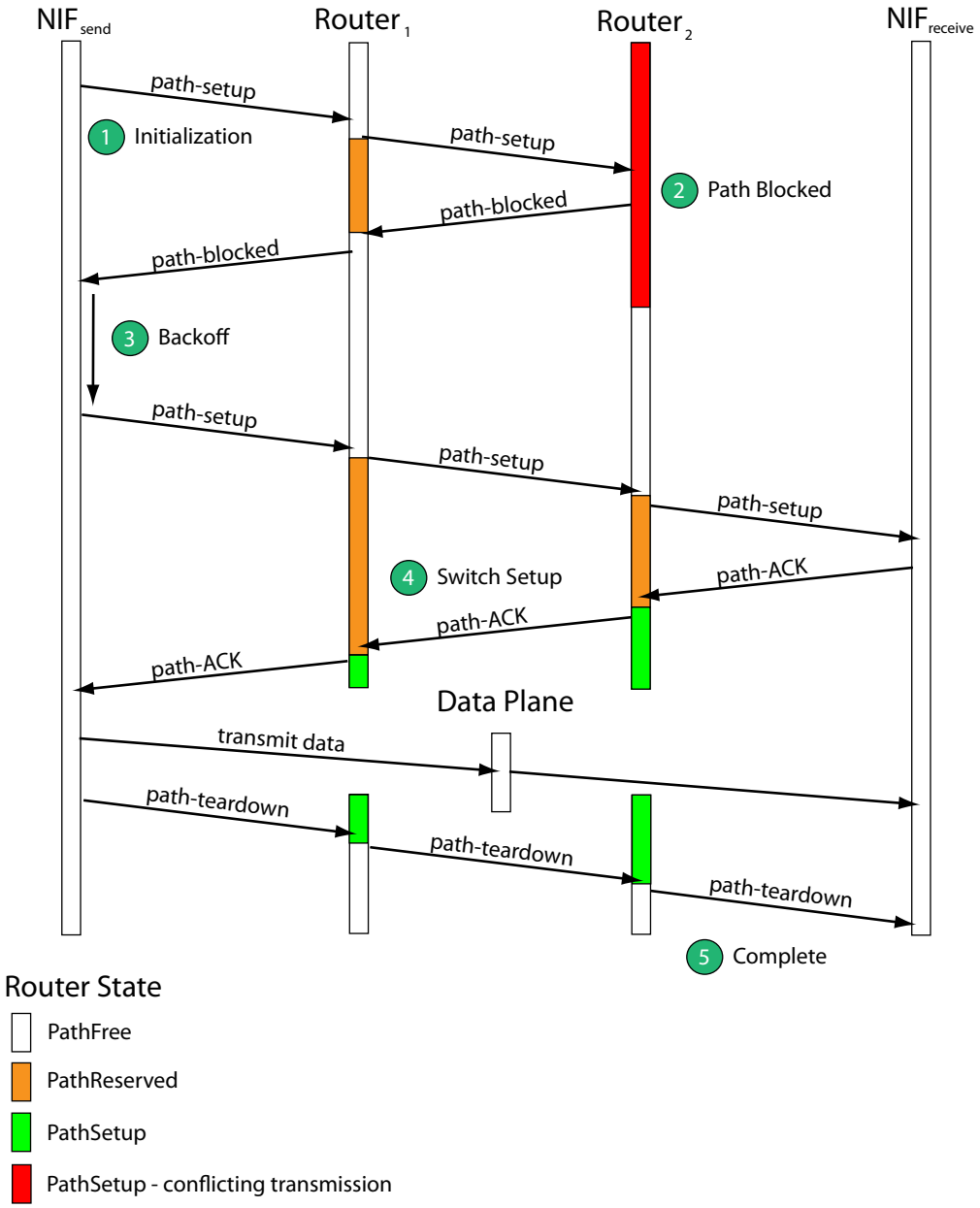


Figure 3.4: Path setup protocol

After the transmission is complete, the path reservation is released by sending *path-teardown* messages through the control network. Though it is shown initiated from the send side in Figure 3.4, path teardown can be initiated from either side. In fact, it may be preferable to have the destination initiate the path-teardown message in case retransmission is necessary due to bit errors.

3.2.2 Photonic Circuit-Switched Network Design

Much in the same way that network topology affects latency and congestion in packet-switched electronic networks, the design and placement of photonic switches in a photonic circuit-switched network will have consequences on performance. In this section, we will show how to design a simple photonic circuit-switched network based on a mesh topology.

3.2.2.1 Switch Design

The basic building block of a spatially-switched network is the spatial switch. As we discussed earlier, and shown in Figure 2.9, we can make spatial switches which are able to switch all the wavelengths of an optical signal at once from one port to another out of ring resonators. These 1×2 and 2×2 fundamental switching units can be used to realize higher-order switches. A natural design choice is trying to implement a 4×4 switch, one port for each of the cardinal directions.

Let us consider how this can be accomplished. Start by referring to Figure 3.5(a), which is simply four intersecting waveguides implementing a bidirectional port at each cardinal direction. The column labeled 3.5(a) in Table 3.1 summarizes the functionality of this simple configuration. Obviously, this is not much of a switch because it doesn't implement all possible paths, denoted by the Inport/Outport pairs in Table 3.1. We do not consider path turnarounds (for example, *from* the East port *to* the East port), because in circuit-switching, this is unlikely to be very useful.

Now we add in broadband ring resonators at waveguide junctions to implement paths in the switch, which is shown in Figure 3.5(b) and first appeared in a paper by Shacham (44). The column labeled 3.5(b) in Table 3.1 designates which ring is used to implement which path, completing all possible paths. Note that eight rings are needed to accomplish this, because the coupling regions between the ring and waveguides must be engineered specifically to support certain optical paths for the best switch extinction.

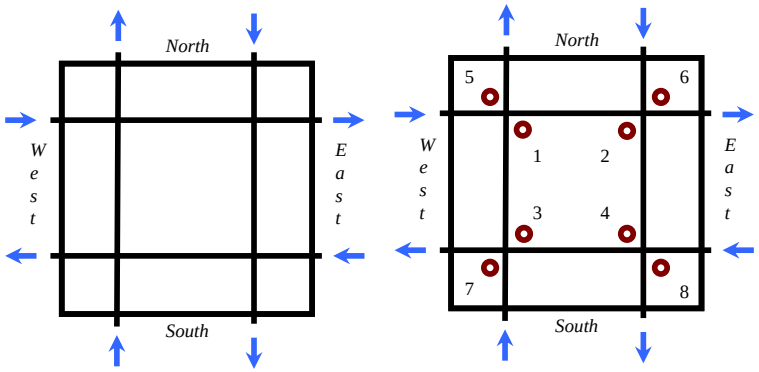


Figure 3.5: Designing a 4×4 switch. (a) just the crossing waveguides. (b) adding broadband rings to implement paths.

Table 3.1: Switch Design

Inport	**Outport**	**Ring/Path**				
		3.5(a)	3.5(b)	3.6(a)	3.6(c)	3.6(b)
W	N		5	–	–	3
W	E	–	–	4	7	–
W	S		2	8	6	8
N	E		6	2	–	2
N	S	–	–	6	5	–
N	W		4	–	1	5
E	N		3	1	3	1
E	S		8	–	–	6
E	W	–	–	5	2	–
S	N	–	–	3	4	–
S	E		1	–	8	4
S	W		7	7	–	7

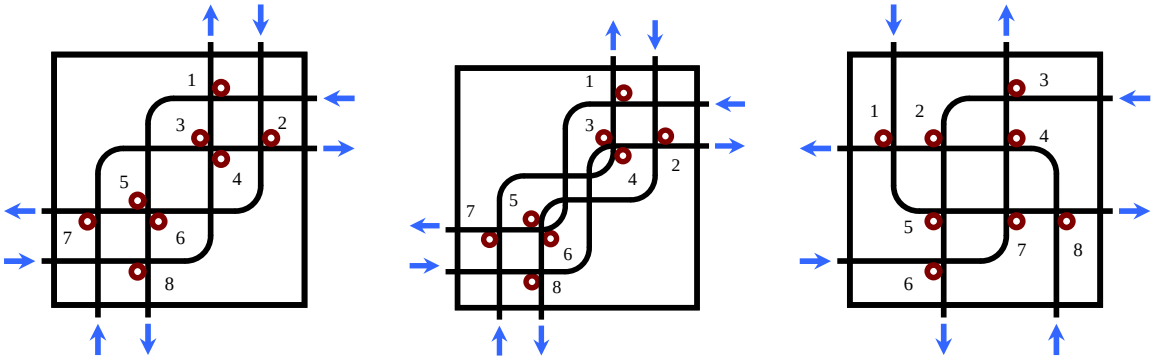
Though the switch in Figure 3.5(b) can implement all paths, as indicated in Table 3.1, it may not be as useful as it could be. If we want to use the switch in a network where we wish to have more than one path through the switch active at once, then the switch in Figure 3.5(b) will surely produce a large amount of circuit-path blocking. For example, the path from West to South cannot be active at the same time as the path from North to West, because those two paths rely on the same waveguide (too much optical power in a waveguide will produce unwanted nonlinearities), and more importantly the two paths rely on conflicting ring states (in rings 2 and 4).

This motivates us to come up with switch designs which can implement any number of paths, as long as the inport and outport don't conflict (because again, two signals in the same waveguide). In other words, we need a non-blocking switch design. The first of these designs was originally proposed by Wang (45), and can be seen in Figure 3.6(a). The next two switch designs, seen in Figures 3.6(c) and 3.6(b) were proposed by Chan (46) to reduce total insertion loss in a circuit-switched network, which as we will see later in Section 3.2.2.2 can be extremely important. Table 3.1 has columns which specify which rings implement which paths for these switch designs. The switch in Figure 3.6(b), labeled as StraightPath, was proposed to implement straight default paths (no switching through a ring), because switching through a ring incurs more loss than passing by one and straight paths through switches are more common in regular tiled network topologies. The switch in Figure 3.6(c), which we label as Symmetric, was proposed to reduce insertion loss by removing two waveguide crossings, which contribute to loss.

Finally, higher radix switch designs are also possible. Figure 3.7(a) shows a structure that is scalable to any number of ports, though all the inports are on the same side, as well as the outports. If turnarounds (inport going to same outport) are not required, some rings can be removed, as shown in Figure 3.7(b). Finally, implementing default paths for all inports, the switch in Figure 3.7(c) achieves the minimum number of 8 rings for a 4×4 switch. Though this design is higher loss than the switches of Figure 3.6, it is scalable to an arbitrary number of ports using the minimum number of rings.

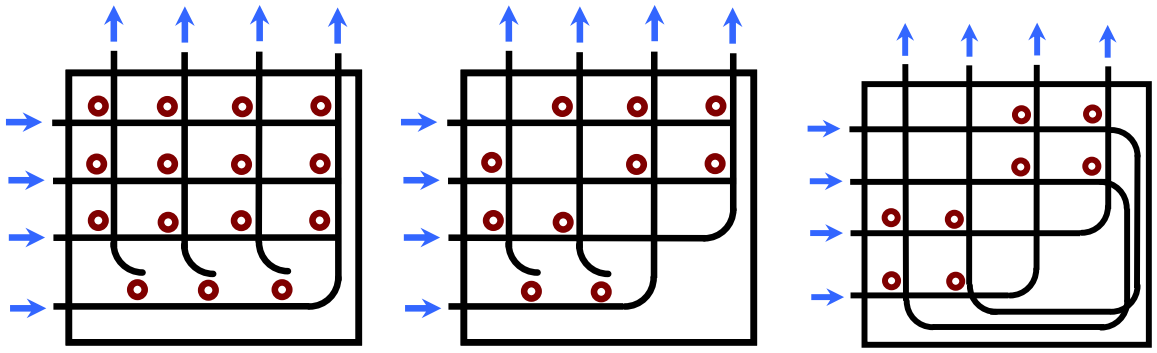
3.2.2.2 Optical Power Budget

One major design constraint of any photonic network is the optical power budget. This tells us the limits on injected optical power, and how this relates to the number of wavelengths we can use (and for



(a) Original (b) StraightPath (c) Symmetric

Figure 3.6: Various 4×4 non-blocking switch designs
(a) the original 4×4 non-blocking switch, (b) a 4×4 non-blocking switch that reduces the number of crossings, and (c) a 4×4 non-blocking switch that minimizes insertion loss for the straight path cases (North - South, and West - East).



(a) (b) (c)

Figure 3.7: Higher radix switch designs.
(a) basic structure of a switch with a scalable number of ports. (b) removing rings which implement turnarounds. (c) implementing a default path for each inport, to reduce the number of rings.

circuit-switched networks, the link bandwidth). The optical power budget is determined by two things: the nonlinear threshold of the devices, and the detector sensitivity.

The nonlinear threshold (Ψ) is the amount of optical power that will start to induce nonlinear effects. Two of these effects are known as two-photon absorption (TPA) and free-carrier absorption (FCA), both of which result in dramatically higher losses, and become relevant at high optical powers. For a microring resonator device, this could also affect the resonant behavior, thus changing its functionality. Figure 3.8 shows the three nonlinear threshold limitations we are concerned with in circuit-switched networks. The first, labeled ①, is the potential for nonlinear effects to be induced in a waveguide. Since optical power is highest right when it is injected on chip, this is most likely to occur right after the laser source delivery point. The waveguide nonlinear threshold says:

$$\sum_{\lambda}^{N} P_{\lambda} \leq \Psi_{wg} \tag{3.1}$$

or the sum of the optical power of all the wavelengths at a point in a waveguide must be less than the nonlinear threshold power, where N is the number of wavelengths. Since this critical point is directly after the laser source delivery point, the power for each wavelength is equal to the injected power, or $P_{\lambda} = \rho_{\lambda}$. Typical values for Ψ_{wg} can be in the 10's of mW, or around 10-20 dBm (19).

The second nonlinear critical point is in a modulator, labeled ②, which says:

$$P_{\lambda} \leq \Psi_{mod} \tag{3.2}$$

Since a modulator is resonant with only one wavelength, we are only concerned with the optical power of a single wavelength. While the real value of P_{λ} should be $\rho_{\lambda} - \zeta_{delivery}$, where $\zeta_{delivery}$ is the optical loss from the laser source delivery point to the first modulator, $\zeta_{delivery}$ is usually negligible, and therefore we only consider the injected power ρ_{λ}. However, assuming on/off keying, or amplitude modulation, a logical "0" means that optical power will stay circulating in the resonator until it dissipates (up to hundreds of times), while more power is constantly flowing in. Because optical power builds up in this way, the nonlinear threshold for a modulator (Ψ_{mod}) is usually much lower, around 0.6 mW, or -2 dBm.

Finally, for circuit-switched networks, the third potential source for nonlinear effects is in a broad-

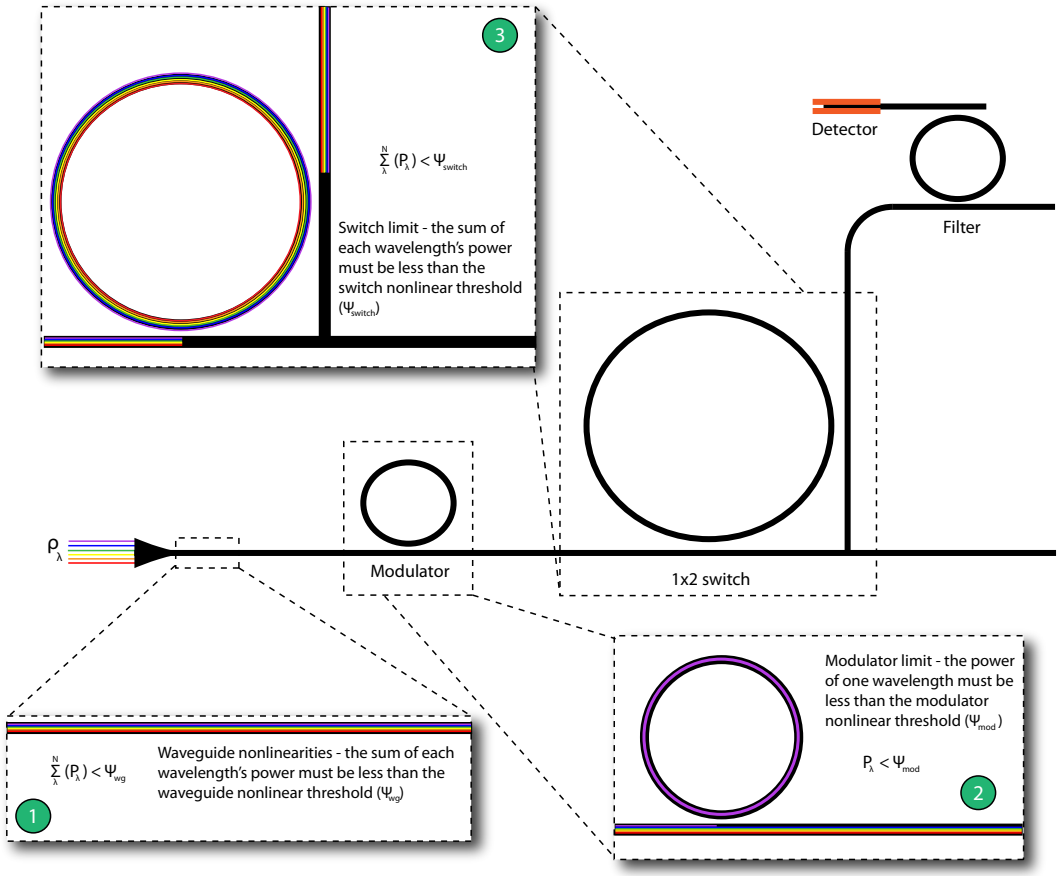


Figure 3.8: Points in a photonic link where nonlinear effects can be generated. (1) in a waveguide (2) in a modulator (3) in a switch.

band switch, labeled ③, which says:

$$\sum_{\lambda}^{N} P_{\lambda} \leq \Psi_{switch} \tag{3.3}$$

or the sum of the optical power of all the wavelengths going through the switch must be less than the nonlinear threshold power Ψ_{switch}. Since the switch that will experience the most amount of power is the one immediately following the modulator bank, the power of each wavelength will be $P_{\lambda} = \rho_{lambda} - \zeta_{delivery} - \zeta_{modulation}$, where $\zeta_{modulation}$ is the loss experienced passing through the modulator bank and being modulated (see Section 3.2.2.4). Also, much in the same way that a single wavelength will circulate many times in a modulator, all of the wavelengths will also circulate in the switch building up optical power, though not as many times (about 2 on average). This places a typical nonlinear switch threshold value of around 7-17 dBm, or half the amount of power in a single waveguide.

The other key factor in the optical power budget is the detector sensitivity. The photodetector sensitivity determines the optical power that must reach the receiver for bits to be reliably detected. For both cases, we assume the receiver circuit requires a peak current $\geq 20\mu A$, or an average power of $\geq 10\mu A$ for error free operation (BER $\leq 10^{12}$) (47). For a single-layer germanium detector, a responsivity of 1 A/W is not unreasonable which then requires an average optical power $\geq 10\mu W$, or -20 dBm.

These two limits, the nonlinear threshold and the detector sensitivity, form the optical power budget which will mandate how many wavelengths can be used and how far each optical signal can travel, which ultimately limits the scalability of a network. Figure 3.9 shows a summary of the optical power budget for the two summation-based nonlinear thresholds (waveguide and switch). It shows that each wavelength must inject enough power to overcome the worst-case insertion loss, where the total power injected is limited by the nonlinear thresholds. This means that a network with a lower amount of worst-case insertion loss will be able to reliably use more wavelengths, and thus have higher link bandwidth. The modulator nonlinear threshold places one final limitation on the system's worst-case insertion loss, in that if a single wavelength must inject more power than Ψ_{mod} to overcome loss, then it cannot be reliably modulated, and the network could therefore never be implemented.

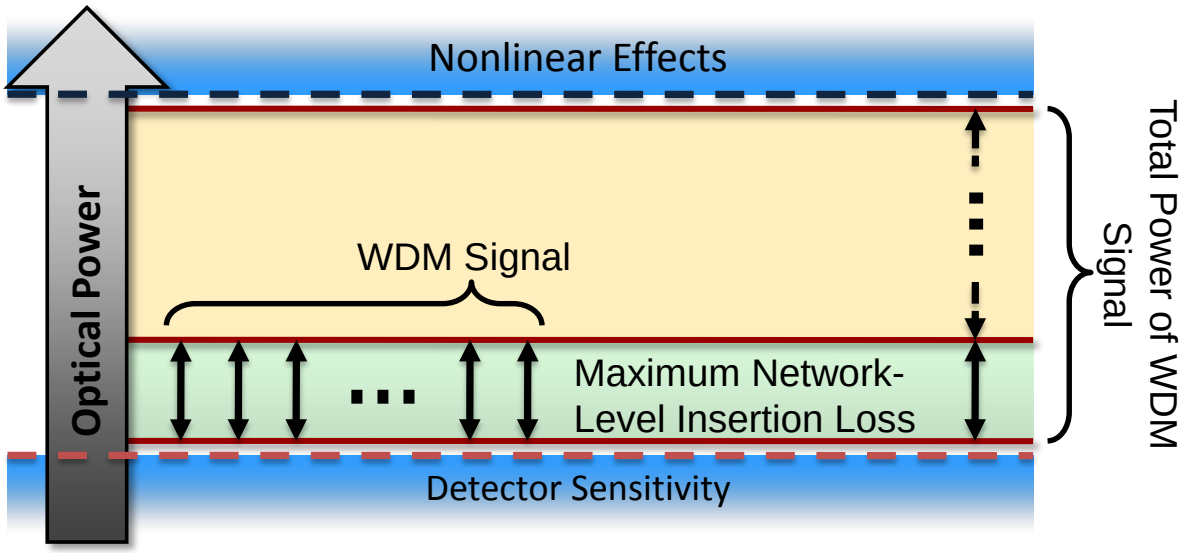


Figure 3.9: Insertion loss vs. bandwidth tradeoff.
The network-level insertion loss and the optical power budget are intertwined in influencing the scalability and performance of the network. They dictate how large and complex a network can become and also determine the number of wavelengths that can be used by the network.

3.2.2.3 Data Integrity and Crosstalk

A variety of interactions in a photonic interconnection network will work to degrade the integrity of transmitted data. Some of the main sources of noise that will accrue in an optical signal are intensity noise generated at the laser sources, inter-message crosstalk, intra-message crosstalk, and electrical noise generated by the optical receivers (Fig. 3.10). The standard figure of merit for measuring the quality of signal is the signal-to-noise ratio (SNR) which is defined as the ratio between signal power and noise power. From a system perspective, the SNR can be used to determine the statistical likelihood that each bit of data is transmitted erroneously (*e.g.* a transmitted 0 is detected as a 1), also called a *bit error rate* (BER). An understanding of the potential noise in any interconnection network is critical to determining the effective throughput of the system since error detection and correction will invariably cause performance overheads.

The first source of noise is from the laser sources which inherently causes random fluctuations in an optical signal, called intensity noise. This noise is quantified as relative intensity noise (RIN), which is the ratio of the power variance of the optical signal to the mean optical power squared. Quantum cascade lasers have been measured to have RIN on the order of -150 $\mathrm{dB\,Hz^{-1}}$ with an output of 10-dBm mean

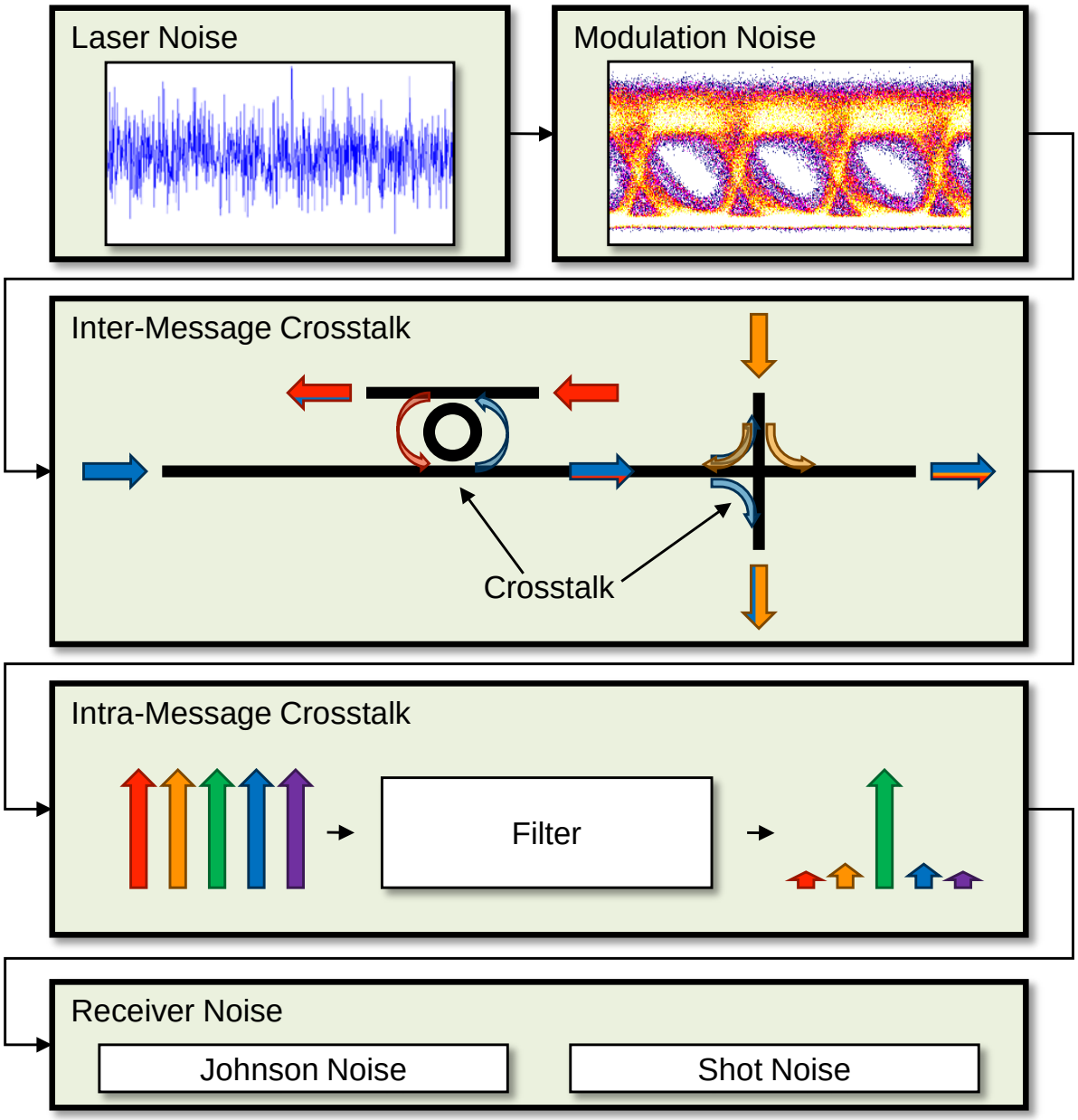


Figure 3.10: Sources of noise and crosstalk within a chip-scale photonic system.

optical power (48). To convert to a SNR, we use the theoretical relation (49):

$$SNR_{laser} = \frac{m^2}{2B \cdot RIN} \tag{3.4}$$

where B is the noise bandwidth, assumed equal to the modulation rate, and m is the modulation index, equal to $1 - E$, where E is the extinction ratio of the modulator.

A second source of noise is *inter-message crosstalk* which occurs when multiple photonic messages concurrently propagate through a photonic device. In a waveguide crossing for example, the ideal situation is for two orthogonally propagating messages to be completely isolated from each other with no interaction. In reality however, a small amount of optical power from each message will leak onto the other message. A similar situation occurs in ring-resonator filters and switches due to imperfect coupling of each wavelength channel.

For an N-port device, the crosstalk power that a message on a particular port receives is given by the sum of the power that is leaked by any existing messages on the other $N-1$ ports. If M is the set of all signals present in the device and the power of a signal k is given by the variable P_k, then the crosstalk power seen by signal s is given by

$$\sum_{k \in M, k \neq s} \frac{P_k}{IL(portin_k, portout_s)} \tag{3.5}$$

which aggregates the unwanted signal power that leaks into the output port being used by s. Function IL refers to the insertion loss characteristic between ports of the device. In Eq. (3.5), $portin_k$ denotes the input port of a message k, and $portout_s$ denotes the output port of s. This calculation is a first order approximation that only considers crosstalk for messages that coexist in a device and not from leaked power that propagates across multiple devices before interfering with a foreign signal.

A third source of noise called *intra-message crosstalk* occurs due to imperfect filtering. For example, in order for a WDM message to be received and converted into an electrical signal, each wavelength channel must be individually filtered and fed into a photo-detector. Due to imperfect extinction, power from the adjacent wavelength channels will leak through causing an additional source of noise. Intra-message crosstalk will also occur in any other location in a photonic network where filtering functionality is involved. The spectral response of ring resonators mimics a periodic Lorentzian function. Lastly,

noise is contributed at receivers from Johnson (thermal) noise and shot noise.

The combined effect of these multiple sources of noise can be used to compute an SNR for the final detected signal with the following equation:

$$SNR = \frac{P}{N_{laser} + N_{inter} + N_{intra} + N_{therm} + N_{shot}} \tag{3.6}$$

where P is the signal power and N corresponds to the noise power associated with noise or crosstalk source indicated by the subscript.

3.2.2.4 Modulator and Detector Banks

An important part of a circuit-switched network is the subsystem that converts from the electrical to optical domain and back again: modulator and detector banks. Besides the power and performance considerations associated with operating high-speed SerDes, drivers, and receivers, here we consider the optical physical layer characteristics of these modules. Figure 3.11 shows a modulator bank consisting of 4 ring resonators. Low speed data held in buffers (clocked at clk1) is serialized with shift registers (at clk2) to the drivers, which then control the active regions of the modulators.

The first case we should consider, labeled ①, is when modulators are off, essentially sending 0's. As illustrated in Figure 3.11, the important thing to consider here is the leakage due to imperfect extinction or resonance misalignment, which could contribute optical noise power, σ_m, to other valid signals in the network. This motivates us to design modulators such that they have high extinction, and maintain their resonances through design or with thermal tuning.

The second case, labeled ②, is shifting a modulator's resonance a spectral distance, δ, to send a 1. For a 1, our main concern is now insertion loss, because we need 1's to be able to overcome the insertion loss of the network in order to trigger clock edges for reception, assuming either coding or clock transmission. For the example labeled ②, the insertion loss of wavelength 3 is $\zeta_{modulator}(\delta)$, a value dependent on the distance the resonance is shifted. Note that shifting the resonance further will result in lower modulation loss, but requires more power, and is limited by the distance to the adjacent wavelength. There will also be some loss from the response tails of modulators R2 and R4, which we consider negligible because we make sure our resonances are aligned far enough apart, as we will see

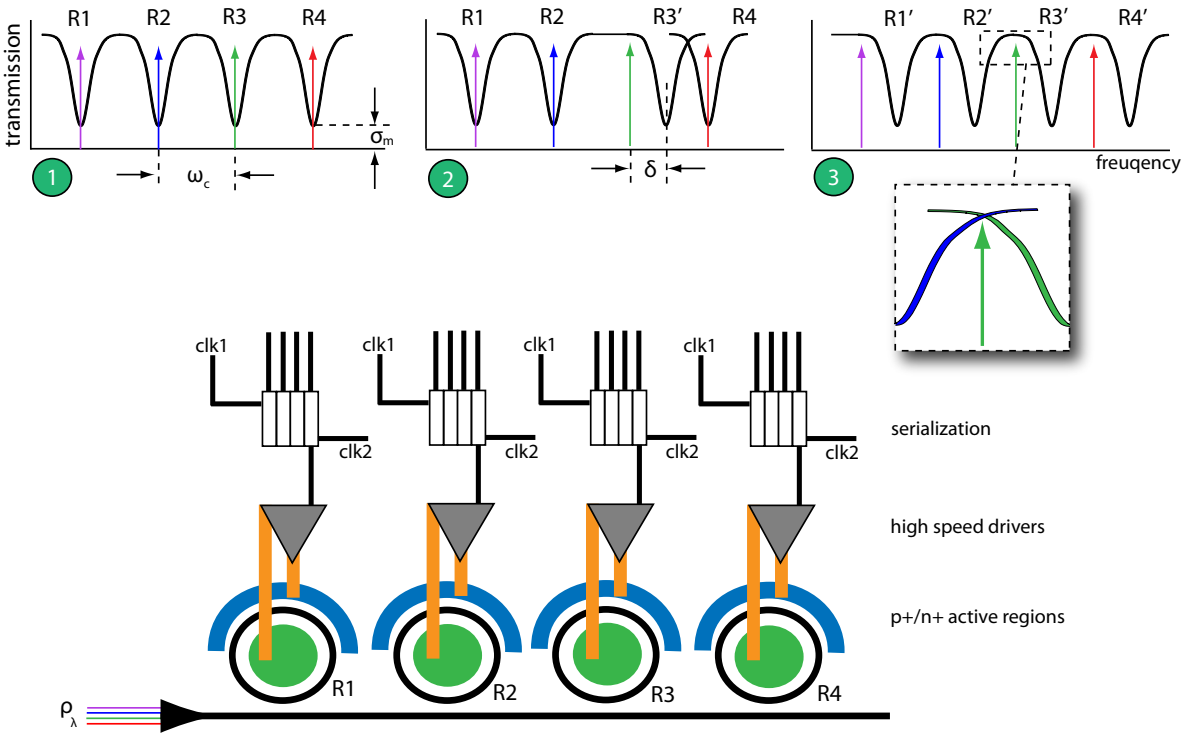


Figure 3.11: Modulator bank and response issues.
(1) all 0's, with extinction noise (2) a 1, shifted by δ with loss (3) all 1's, incurring $2\times$ loss.

later talking about detector banks.

Finally, the third case, labeled ③, is when there are more than one 1 adjacent to each other. Just like case ②, we are concerned with insertion loss, but here the loss will be $\zeta_{modulator}(\delta)+\zeta_{modulator}(\omega_c-\delta)$, where ω_c is the channel spacing. In addition to the loss incurred by the wavelength's own modulator, it now incurs the closest modulator's loss as well. If the modulators are designed such that $\omega_c = 2\delta$, or the resonance shifted falls exactly between channels, then the loss of one wavelength becomes $2\zeta_{modulator}(\delta)$, because the loss from both the wavelength's own modulator and the adjacent one will be equal.

Taking the worst case of all these effects which we must design for, we can expect the optical power per wavelength after modulation to be $P_\lambda = \rho_\lambda - 2\zeta_{modulator}(\delta) - \zeta_{propagation}$, and having σ_m noise power per wavelength.

The detector bank, shown in Figure 3.12, consists of passive ring resonator filters for demultiplexing individual wavelengths, coupled to germanium or other detectors. High speed receivers and deserializers then convert bits to the electrical domain. One important issue we will briefly mention is clock/data

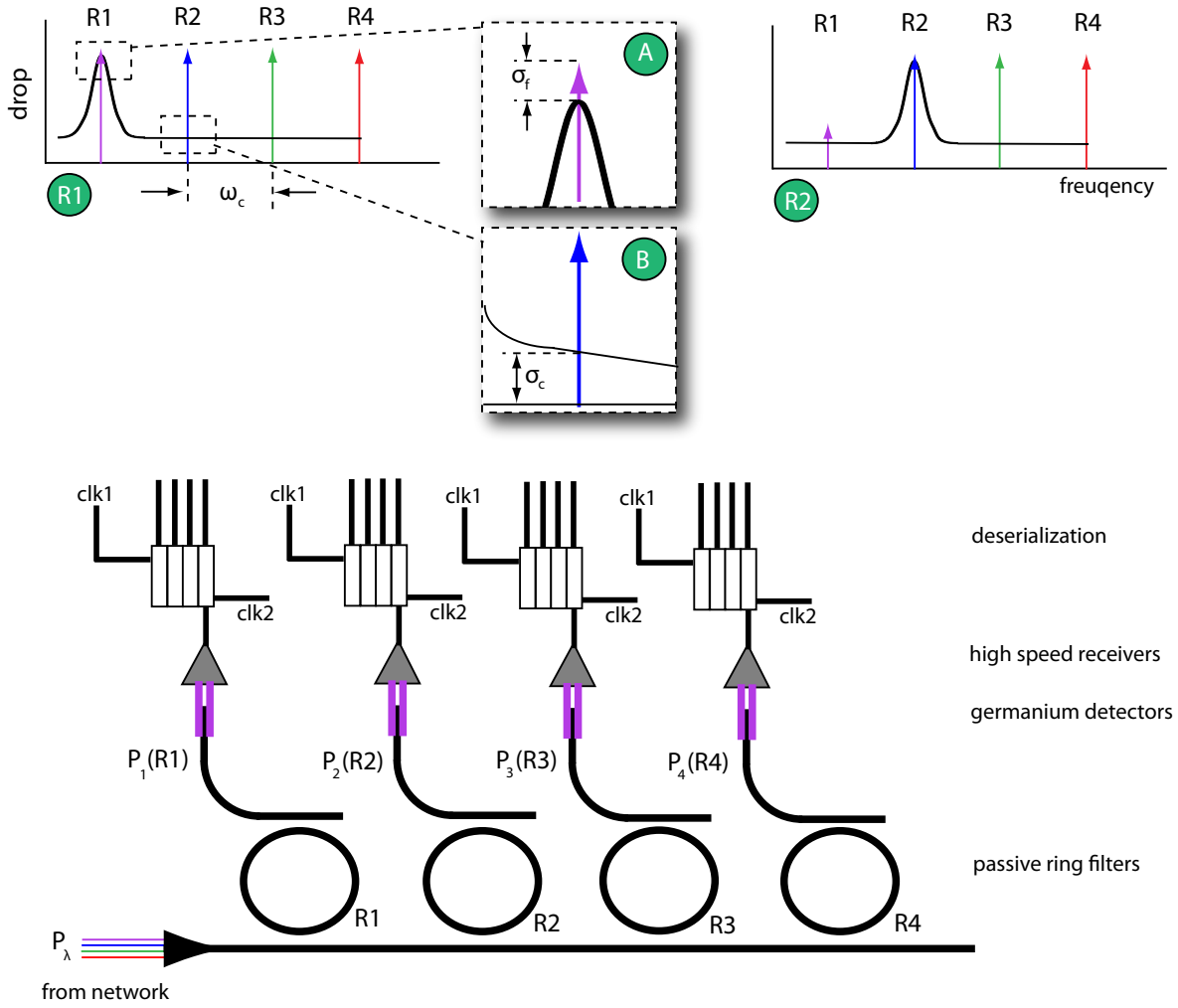


Figure 3.12: Detector bank, showing (R1) first filter response with (A) filter leakage and (B) crosstalk and (R2) the response of the next filter.

recovery (CDR). As of the writing of this book, there is no good solution to recovering a clock in chip computation time scales for high speed data rates. Burst-mode receivers today use phase-locked loops (PLLs) which fundamentally require microseconds of locking time, if not milliseconds in practice. One potential solution to this problem is running a configuration phase at startup, which characterizes the phase difference between data and clock (distributed globally, or asynchronously consistent) when receiving from every other network access point. Each network access point would then save this information, and use it to receive data from different cores at runtime. Circuit-switching uniquely benefits from this approach because a receiving access point knows its source before data transmission occurs by virtue of the path setup protocol (Section 3.2.1).

We will start by considering the optical characteristics of a detector bank by taking a look at the first filter encountered, labeled (R1) in Figure 3.12. There are two important characteristics, illustrated in the inserts labeled (A) and (B): filter leakage (σ_f) and wavelength crosstalk ($\sigma_c(\omega_c)$). Filter leakage is caused by an imperfect filter or drifting resonance, and results in some power being leaked to the next filter, in this case R2. Wavelength crosstalk, which is a function of channel spacing, is an adjacent wavelength leaking into the filter because of the Lorentzian lineshape. After these effects, the R1 detector can expect to approximately receive:

$$P_1(R1) = P_1(1-\sigma_{f1}) + P_2\sigma_c(\omega_c) \tag{3.7}$$

assuming that the units for σ are fractions of the total power. This says the detector will receive the power of wavelength 1 coming in minus the power leaked through, plus crosstalk from wavelength 2 into R1. Now lets take a look at the second filter, R2. Since some optical power leaked through from R1, it will get added the same way as crosstalk. Similarly, since some power from wavelength 2 leaked into R1, its power is diminished. The detector for R2 can expect to approximately receive:

$$P_2(R2) = P_2(1-\sigma_c(\omega_c)-\sigma_{f2}) + P_1(\sigma_{f1}\sigma_c(\omega_c)) + P_3\sigma_{c3}(\omega_c) \tag{3.8}$$

This says that the detector will receive the power of wavelength 2, diminished from R1 crosstalk and R2 leakage, plus crosstalk from the remaining power of wavelength 1 and the full power of wavelength 3. This motivates us to design the filters to have low leakage, narrow lineshape, and wide enough channel spacing to decrease crosstalk as much as possible, keeping BERs down. The remaining filters down the line act similarly, taking into account effects from the two adjacent filters. Higher order effects, taking into account the next 2 or 3 adjacent filters is possible, though must be made negligible by design for a reliable system, and thus can be ignored.

3.2.2.5 Mesh Topology

Now that we have a basic understanding of some of the issues in designing a circuit-switched photonic network, let us consider the design of a simple mesh topology. In general for large complex layouts, using a regular tiled pattern like a mesh can have many advantages in implementation, fabrication, packaging, test, verification, and fault tolerance.

Figure 3.13 shows the layout of a mesh circuit-switched network. For the photonic plane, we use a 4×4 switch as the base and add on the modulator and detector banks to the East and South port for injection and ejection to and from the network. This keeps the insertion loss low because we can use our optimized switches of Figure 3.6, but it does introduce blocking conditions into the switch. For instance, if the local access point is injecting into the network, the incoming south port of the whole switch is blocked. Likewise for the east port and ejecting from the network to the detector bank.

Simple interface logic in the NIF, shown in Figure 3.14 implements the necessary control to send and receive optical data by mediating between the local access point and the electronic control network. The basic NIF can only serve one send request and only receive one transmission at a time by virtue of the network architecture. We will investigate changes to this NIF later, including selective transmission in Section 3.2.3.2 and a complete overhaul in Section 3.2.5. For now, this will serve as the basic network interface for circuit-switched architectures.

We will now describe a full analysis of the photonic mesh network, which we will start calling P-Mesh for short. Lets begin with an insertion loss analysis. Appendix A.1.1 shows how to run this simulation in PhoenixSim. While a simulation is not strictly necessary for just an insertion loss analysis, as it is not time-dependent, it is a useful tool for quickly analyzing complex networks once the model is set up. Figure 3.15 shows the worst-case insertion loss for various network sizes, using each of the three different switches in Figure 3.6. We see that using the Symmetric switch produces the least amount of loss. To find out why, we can refer to Figure 3.16, which shows the breakdown of where insertion loss is gathered for the worst case path for the 8×8 network size. We see that the StraightPath switch does save on insertion loss due to dropping through ring switches as it was intended, but with adding more crossing losses. The Symmetric is able to save even more loss by eliminating some of the crossings from the Original.

Figure 3.15 also shows the modulator power budget, assuming a -20 dBm detector sensitivity and 0 dBm modulator nonlinear threshold. It shows that any network size above 8×8 is infeasible due to loss, only the Symmetric switch will work for the 8×8, and all smaller networks will work. Let's focus on the 8×8 P-Mesh using the Symmetric switch for now. If we know the worst-case network insertion loss ($\zeta_{network}$) is 19.1 dB, we can calculate the total number of wavelengths that can fit into the optical power budget, which we've already constrained for the modulator. For Ψ, we assume a waveguide nonlinear

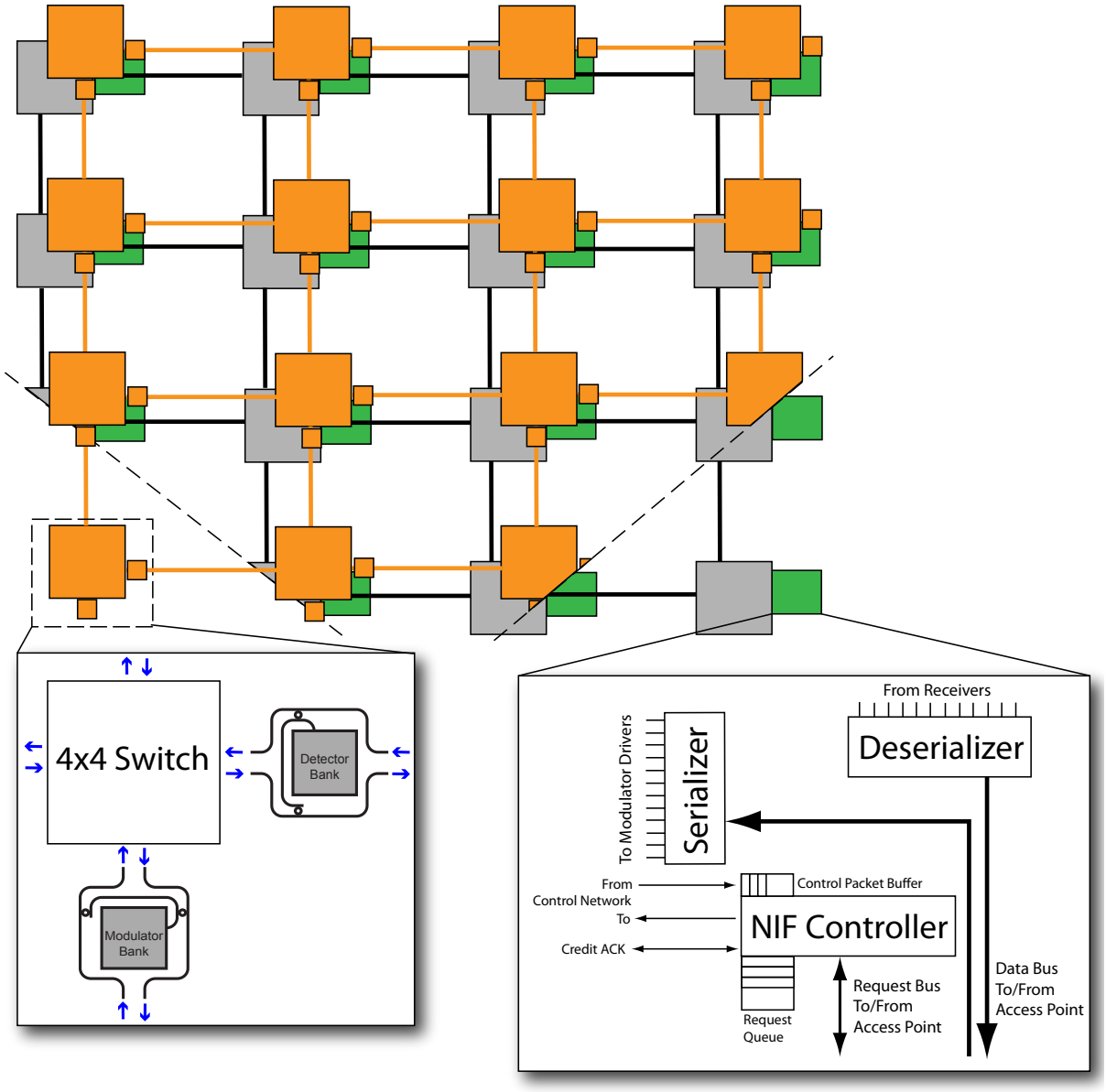
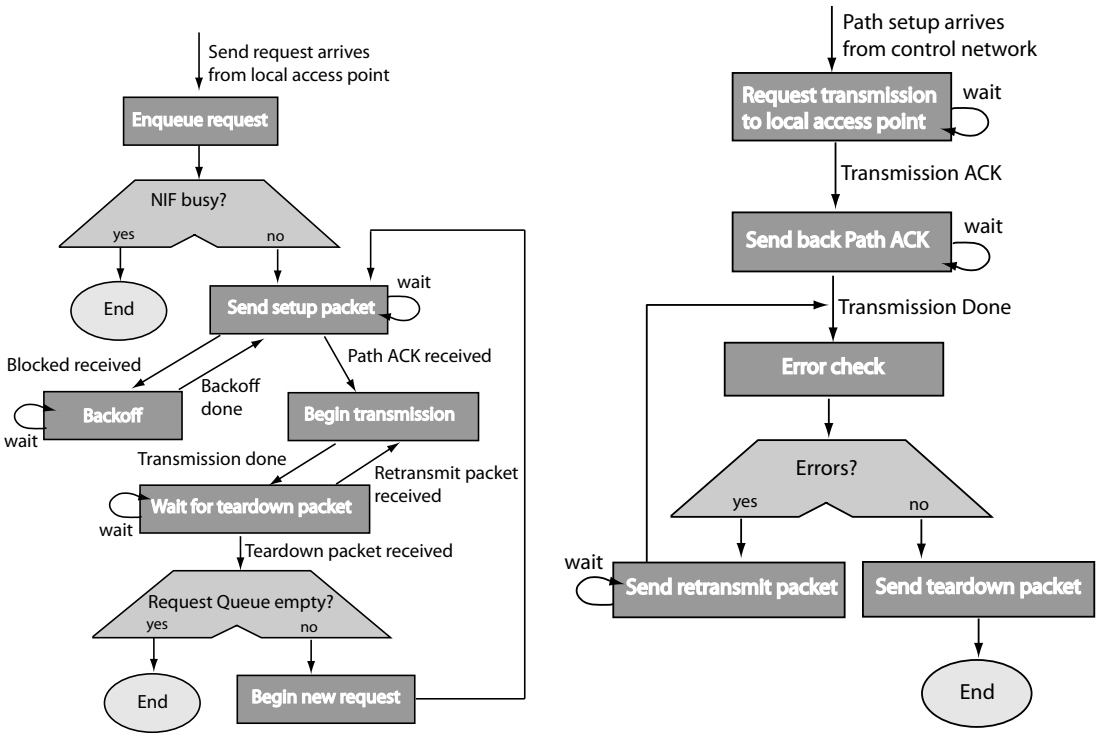


Figure 3.13: 4x4 Mesh topology, showing 5-port photonic switch and NIF block diagram. Key - photonics (orange), electronics (gray/black), NIF (green).



(a) Send (b) Receive

Figure 3.14: Control flowchart for a circuit-switched NIF.
(a) Data-send control (b) Data-receive control.

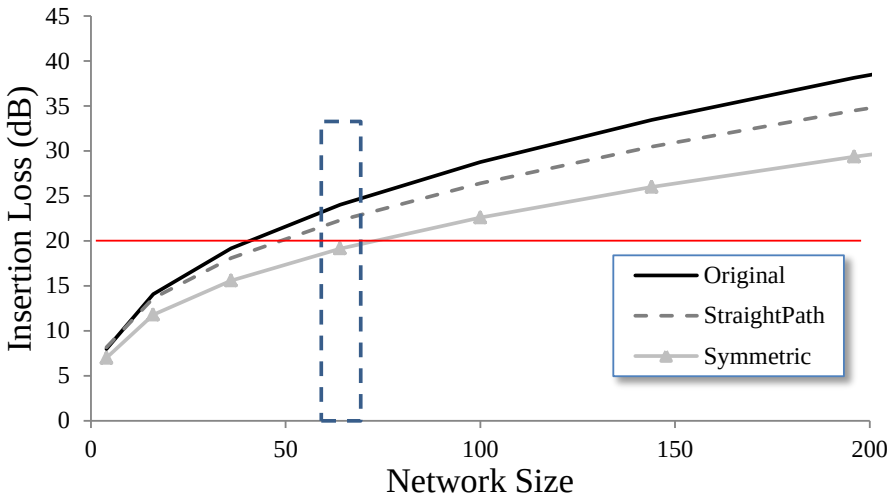


Figure 3.15: Worst case network insertion loss measured in simulation using PhoenixSim for a photonic mesh, using the three different kinds of 4×4 switches from Figure 3.6.
Breakdown of sources of loss for highlighted region is shown in Figure 3.16. Red line indicates modulator power budget.

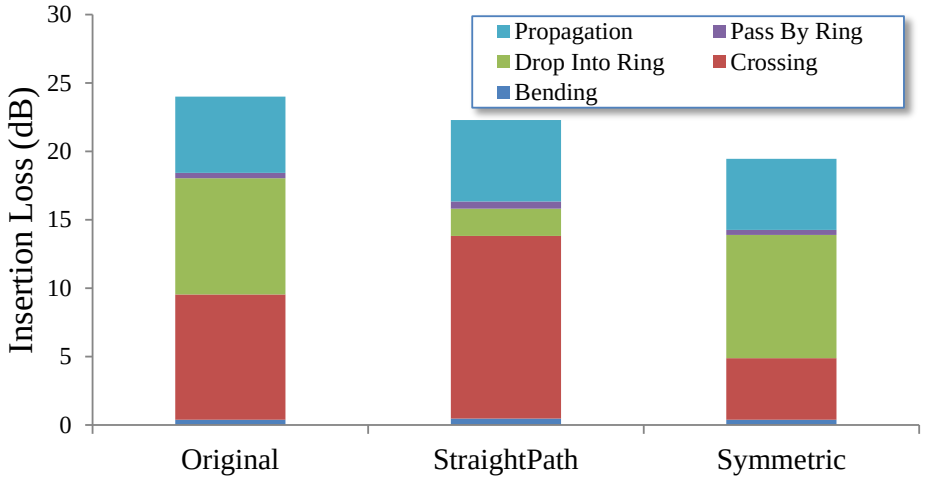


Figure 3.16: Insertion loss breakdown of 8×8 P-mesh for different switch designs.

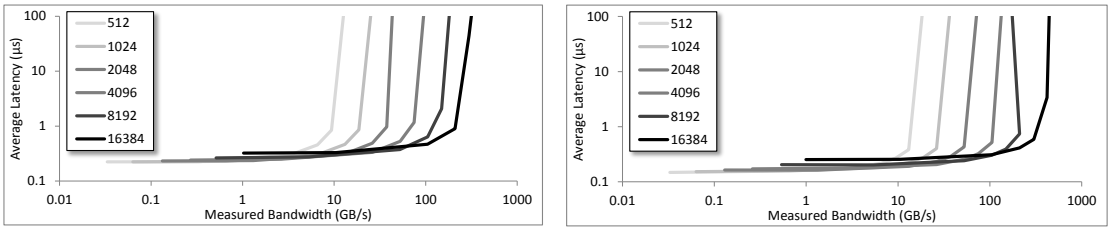
threshold (Ψ_{wg}) of 20 dBm, a switch nonlinear threshold (Ψ_{switch}) of 17 dBm, and recalling from Section 3.2.2.4 accounting for the loss in the modulator bank, we take the more constrained (smaller) value of Ψ_{wg} and $\Psi_{switch} + 2\zeta_{modulator} + \zeta_{prop}$. If $\zeta_{modulator}$ is 0.5 dB and ζ_{prop} is negligible, this means that $\Psi = \Psi_{switch} + 2\zeta_{modulator} = 18dBm$, the amount of optical power that can be injected without incurring nonlinear effects in either the waveguide or the switch. We also assume a detector sensitivity (S_{det}) of -20 dBm, so:

$$N_\lambda = 10^{\frac{(\Psi - S_{det} - \zeta_{network})}{10}} = 10^{\frac{(18-(-20)-19.1)}{10}} = 76 \tag{3.9}$$

the number of wavelengths we can use (N_λ) is 76. We will limit this number to 64 for channel spacing and modulator FSR reasons, which we will go into more detail on in Section 4.2.1 when we optimize the physical layer of broadband networks.

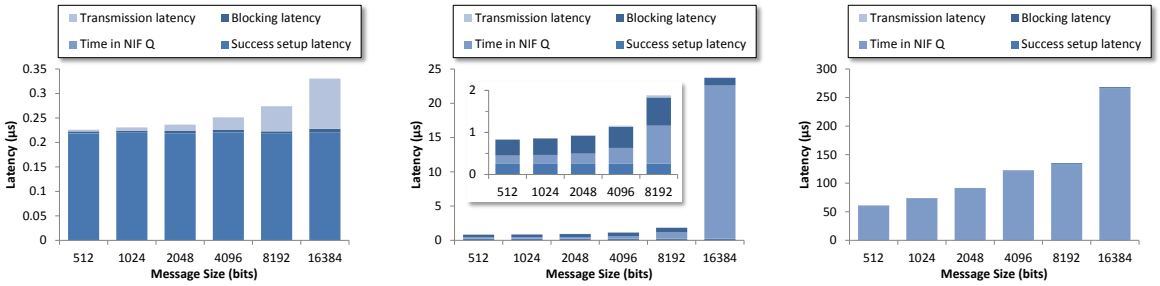
Given the number of wavelengths that are feasible in this network, we can now investigate the performance characteristics through simulation. We will start with latency and bandwidth characteristics, and Appendix A.2.1.1 shows how to run this in PhoenixSim. For now, we will investigate 5 simple synthetic benchmarks: Bitreverse, Random, Hotspot, Neighbor, and Tornado.

Lets take a look at the first two, Bitreverse and Random. Bitreverse is a communication pattern where each node takes the one's complement of its ID, resulting in many long communications (for example, access point 0 will communicate with access point 63 for a 64-node network). For Random, destinations are randomly picked uniformly each time a new communication occurs. Message requests



(a) Bitreverse (b) Random

Figure 3.17: Average latency versus measured bandwidth for P-Mesh running (a) Bitreverse and (b) Random synthetic traffic patterns for different message sizes, in bits.



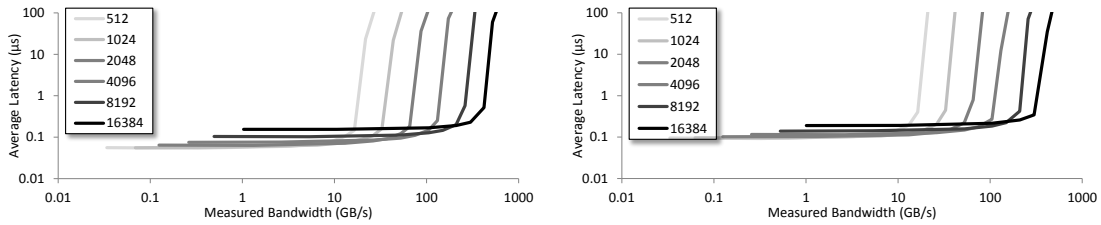
(a) Zero-load (b) Just Before (c) Saturation

Figure 3.18: Latency breakdown for Bitreverse pattern at (a) zero load, (b) just before saturation, and (c) saturation.

are generated as a Poisson arrival process, specified by a mean inter-arrival time.

Figure 3.17 shows the performance of the P-Mesh for Bitreverse and Random for different message sizes. Performance between the two applications are similar, showing a few tens of nanoseconds zero-load latency, saturating in the tens or hundreds of Gb/s of total network bandwidth. Note that it is possible to exhibit *less* bandwidth at *higher* injection rates, shown by the 8192 bits message size, which is due to higher injection rates causing more blocking, thus reducing the overall performance. This has nothing to do with the specific size of 8192 bits, but is rather an artifact of the randomization of message destinations. Across multiple simulation runs this effect would average out, though we presented it here for discussion.

One interesting feature of circuit-switched networks is the zero-load latency insensitivity to message size. Because of the high bandwidth available once the link is established, the transmission time is a relatively small part of the total latency. Figure 3.18(a) shows the latency breakdown for Bitreverse at zero-load. Clearly, nearly all zero-load latency is contributed by the round-trip of the successful path



(a) Neighbor (b) Tornado

Figure 3.19: Average latency versus measured bandwidth for P-Mesh running (a) Neighbor and (b) Tornado synthetic traffic patterns for different message sizes, in bits.

setup. Transmission latency only starts to become relevant at message sizes of 8kb and above.

Figure 3.18(b) shows the latency breakdown at just before network saturation. Here, the time spent in the NIF queue starts to add up as messages back up, and the time spent retrying and backing off while blocked starts to become significant. Finally, in saturation in Figure 3.18(c), latency keeps adding up as the incoming message rate is higher than the network can sustain, backing up messages in the NIF queue. Here, simulation is open-loop, or access points will continue to request communication despite the NIF queue being full. While this does not closely model the behavior of real systems, it enables us to characterize the network's saturation points, shown in Figure 3.17 where latency asymptotes.

The next synthetic benchmarks we will consider are Neighbor and Tornado, two similar traffic patterns where communication is limited to 1-hop and 2-hops away, respectively, in the same row and column. While these patterns may seem trivial, they are important to consider as they are exhibited by many scientific applications which model continuous effects by assigning a processor in a grid to a partition of space, data, and/or computation, sharing boundary data with its neighbors. Figure 3.19 show the latency versus bandwidth characteristics for Neighbor and Tornado. Although they look similar to Bitreverse and Random, the P-Mesh actually exhibits twice as much saturation bandwidth and about one third the zero-load latency, which is expected from having such short distance transmissions.

Finally, the Hotspot benchmark selects an access point at random which all nodes in the network send to for the duration of the simulation. Clearly, this is not the desired behavior of an application on any network, but it is often necessary broadcast/gather type of communication operations, such as implementing a barrier. Figure 3.20 shows the latency/bandwidth results, which look similar to the others, but have some irregular or crooked lines, caused by the high variability in performance depending

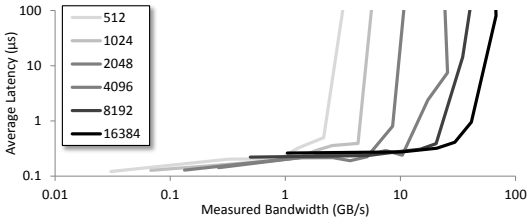
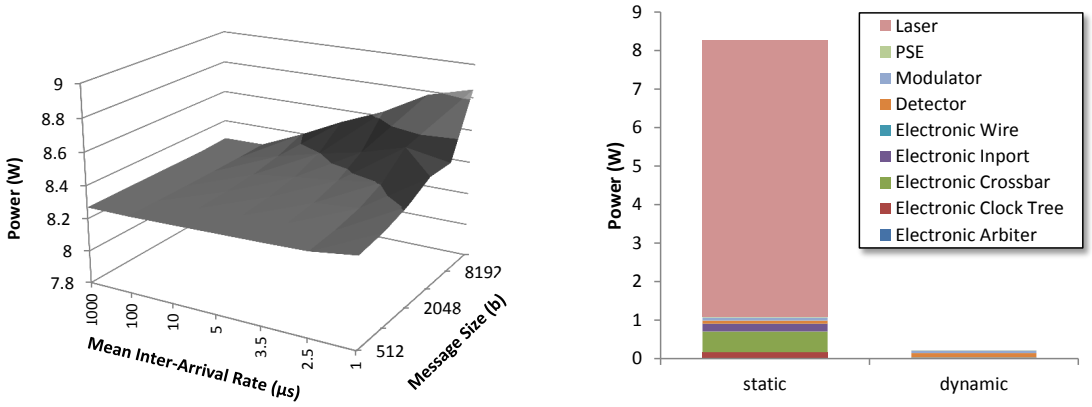


Figure 3.20: Average latency versus measured bandwidth for P-Mesh running Hotspot.



(a) Total (b) Breakdown

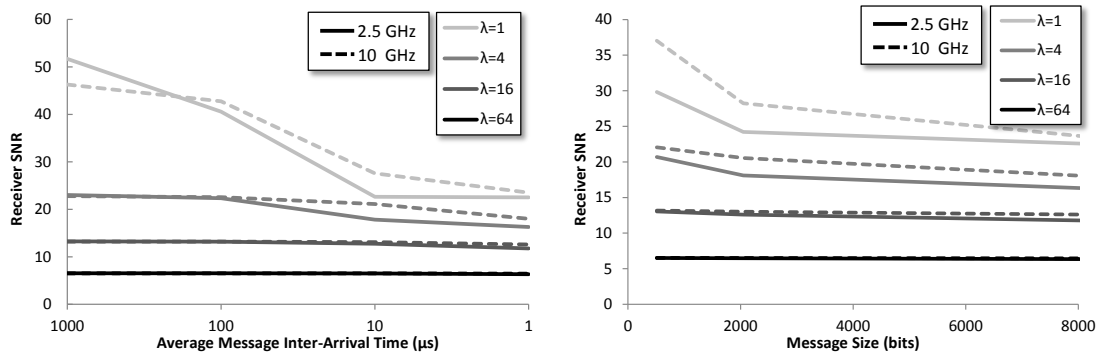
Figure 3.21: Power dissipation in P-Mesh running Random traffic showing (a) total power for various loads and (b) power breakdown at saturation.

on the location of the randomly chosen hot spot. Notice also the decrease in bandwidth (almost an order of magnitude) compared to the other benchmarks, which reflects the ability of a single access points to accept and receive transmissions as quickly as possible.

For on-chip communication, power is likely to be one of the main design considerations. Figure 3.21(a) shows the power dissipation of the network components for Random traffic, though this is essentially characteristic of any traffic. The first thing we see is that the power profile is extremely flat, ranging from 18.3W at zero-load to 8.9W at full saturation. This is because most of the power in the network comes from static power, as shown in Figure 3.21(b), and in particular, laser power. Laser power is found according to:

$$P_{laser}(mW) = \frac{N_\lambda N_{gw}}{\eta_L} 10^{\frac{\zeta_{network} - S_{det}}{10}} \tag{3.10}$$

where η_L is the laser power conversion efficiency (50% assumed for Figure 3.21), $\zeta_{network}$ is the



(a) Arrival (b) Message Size

Figure 3.22: Measured SNR (a) against message arrival rate for 8192b messages and (b) against message size for $1\mu s$ arrival.

worst case network insertion loss, N_λ is the number of wavelengths used, N_{gw} is the number of network gateways where the laser has to be delivered, and S_{det} is the detector sensitivity. If not for the laser power, most of the power would be consumed by the electronic crossbars and buffers of the electronic control network. One advantage of a simple circuit-switched network is that no electronic switching occurs for any of the data bits, which means that the only load-dependent power consumption comes from modulation and detection of optical signals. Clearly though, every extra wavelength costs relatively significant power. Later, in Section 4.2.1, we will take a look at optimization for power and performance where laser power will be important.

Finally, let's consider optical crosstalk of the network. We would expect that as load increases, the SNR of the received signals would decrease due to increasing interference from signals passing each other in switches and waveguide crossings. To find out, we run a simulation for various loads in the P-Mesh while varying the number of wavlenghts and bitrate used. Appendix A.2.1.2 provides details on running this with PhoenixSim. Figure 3.22 shows the results for Random traffic.

Figure 3.22(a) shows the electrical SNR that the receiver sees from the photodetector for different message arrival rates for 8kb messages. Although network load does have an impact for lower numbers of wavelengths due to message interaction in the network, the most dominant factor in SNR is the number of wavelengths itself due to inter-wavelength crosstalk when demultiplexing just before the detectors. The larger the number of wavelengths, the more leaks in to off-resonance filters. The same goes for Figure 3.22(b) which shows SNR against message size for $1\mu s$ average message arrival, though 10 Gb/s

signaling improves SNR somewhat due to physically shorter messages in the waveguides, reducing the likelihood that they intersect.

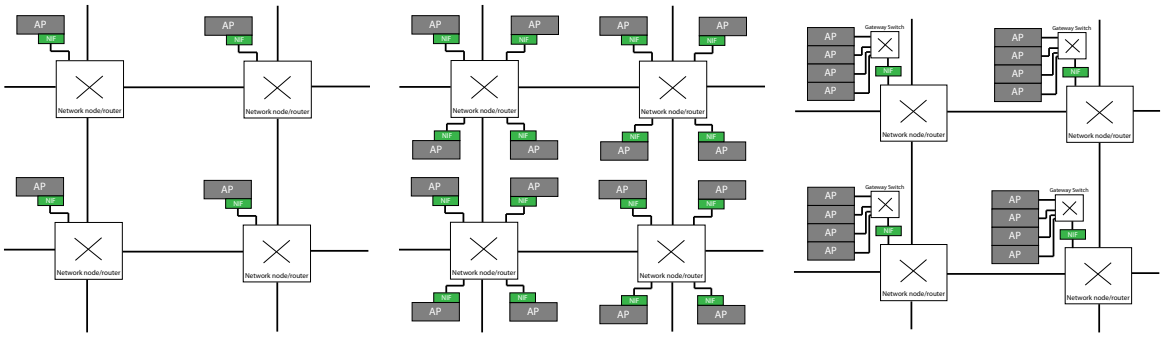
3.2.3 Design Considerations and Improvements

Now that we have explored the effect of the spatial arrangement of circuit-routers, or topology, on physical-layer network characteristics, lets take a look at some minor implementation improvements to a circuit-switched network.

3.2.3.1 Gateway Concentration

Gateway concentration refers to associating more than one network *access point* with a single network *gateway*. Here, the terminology is defined as follows: an *access point* is an entity which requests and receives network communications, such as a core in a CMP. A network *gateway* is an entry point into the network switching and routing functions. For direct networks such as a mesh, every network router or node has a gateway associated with it. Gateway concentration is employed to essentially increase the usefulness of the network by making sure its bandwidth is being used at every network node. Ignoring the contention characteristics of the network for a moment, if there are more access points sharing the same network gateway, that gateway is less likely to be underutilized. When implementation is highly constrained, such as chip-scale NoC's, every transistor and resource which takes up area and power must be used to it's maximum potential. Besides improving the effectiveness of the network resources, concentration can also improve the scalability in terms of network access points for networks that suffer in some way as they get larger.

Kumar *et al.* (50) investigated the two different kinds of concentration for electronic NoC's: *integrated* and *external*, illustrated in Figure 3.23 for a portion of a mesh topology. For integrated concentration, in Figure 3.23(b), each access point is given its own NIF, increasing the radix of the network node/gateway. External concentration, in Figure 3.23(c) uses a separate gateway switch to arbitrate access to the main network NIF. External concentration is usually preferred when increasing the radix of the network node is especially complex or costly, the expected global communication requirements of individual access points is sufficiently low such that sharing a single network gateway will not significantly affect performance, or it is desirable to use different network technologies or architectures for



(a) Normal (b) Integrated (c) External

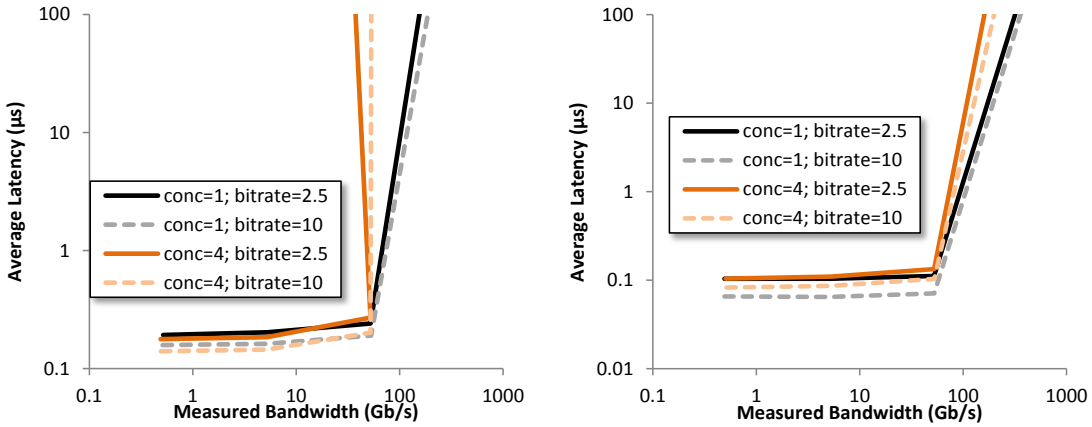
Figure 3.23: Network access point (AP) concentration on a mesh, (a) normal, unconcentrated (b) integrated 4-way concentration, and (c) external 4-way concentration.

local and global communications.

For a circuit-switched network, in most cases it is relatively difficult to increase the radix of the photonic switch while maintaining a manageable number of waveguide crossings and rings. For this reason, we generally have to use *external* concentration for circuit-switching. This has fairly significant implications on power and performance for a few reasons.

First of all, it is generally believed that electronics are still more efficient for traveling short distances on a chip because of the maturity of CMOS technology and design, which makes it a natural design choice to implement the gateway switch with electronics. However, to avoid having to provide sufficient buffering at the NIF to hold entire (potentially large) photonic transmissions, we extend the path setup protocol into the gateway switch domain so that paths are reserved *through* the gateway switch both from the sending and receiving side. The important implication of this is that the gateway switch must be able to bandwidth-match the photonics, which can require a relatively large high power or high-speed electronic crossbar. If concentration must be used to increase the scalability of the network, it may be therefore beneficial to use lower modulation rates to avoid both SerDes and large gateway crossbars.

To illustrate these tradeoffs, we will demonstrate different versions of a normal and concentrated P-Mesh. Details on running this simulation can be found in Appendix A.2.2. We consider two versions of a P-Mesh: one with no concentration, and one with 4-way external concentration. Both are normalized to the same size, 64 access points. This means that the unconcentrated photonic network will be 8×8, whereas the concentrated will be 4×4 (with 4 access points at each node). We also consider both 2.5 Gb/s



(a) Random (b) Neighbor

Figure 3.24: Performance for (a) Random and (b) Neighbor for normal and 4-way external concentration, with 2.5 and 10 Gb/s modulation.

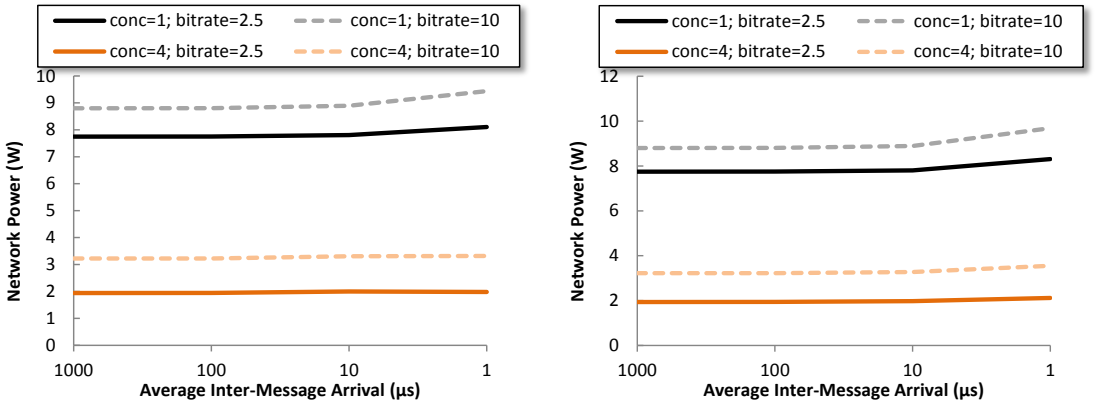
and 10 Gb/s modulation, as it will affect the power considerably. Both networks use 64 wavelengths, limited by either modulator power limit or modulator FSR. For convenience, we choose an 8kb message size for all experiments.

Figure 3.24 shows the latency/bandwidth characteristics of each version of the network. Though the differences are small, the full unconcentrated network is able to achieve more bandwidth at saturation due to blocking from concentrated cores trying to gain access to the NIF. As expected, this is more profound for Random traffic, where it is much more likely that cores will want to transmit outside their immediate concentration group.

Figure 3.25 shows the power dissipation for the different networks. Notice that the concentrated versions both use much less power. To understand why, Figure 3.26 shows the breakdown of power consumption in the networks. Because the concentrated networks are smaller, there is much less insertion loss, thus requiring less injected laser power. Ignoring laser power for a moment, we also see that the networks using 10G modulation either have significant SerDes power (for unconcentrated), or significant crossbar power (for high-speed gateway switch).

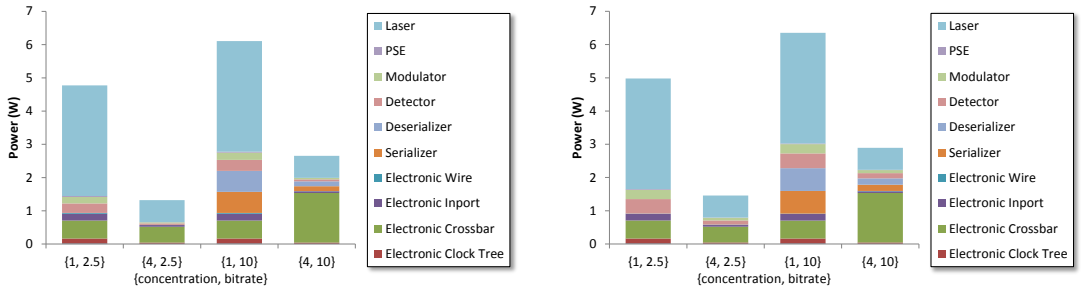
3.2.3.2 Selective Transmission

Intuitively, an electronic packet-switched network would be much more efficient at transporting small packets over small distances, whereas circuit-switching would be more useful for larger messages, larger



(a) Random (b) Neighbor

Figure 3.25: Power for (a) Random and (b) Neighbor for normal and 4-way external concentration, with 2.5 and 10 Gb/s modulation.



(a) Random (b) Neighbor

Figure 3.26: Power breakdown for (a) Random and (b) Neighbor for normal and 4-way external concentration, with 2.5 and 10 Gb/s modulation.

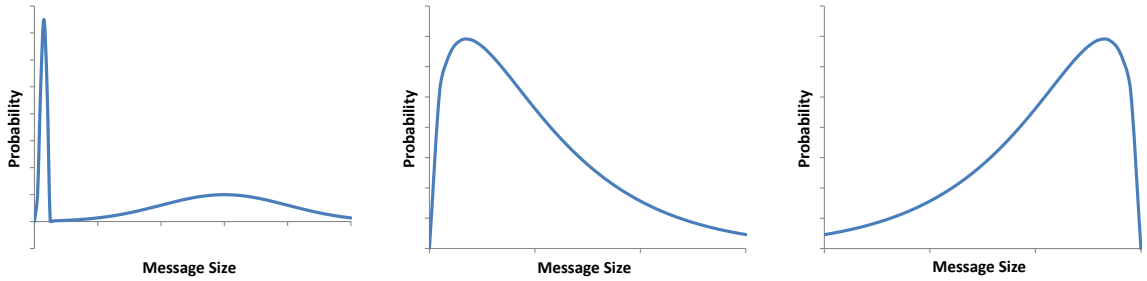
distances, or relatively static communication patterns. For a circuit-switched network, since there is already a light-weight packet-switched electronic network for control and arbitration, we can implement a policy in the NIF to use the control network instead of the standard photonic path setup for smaller messages or for shorter distances. We call this policy *selective transmission*, and was first used in photonic circuit-switched networks by Hendry *et al.* (51).

The first questions that are raised is what size message or distance do we decide to send on the control network as opposed to the photonic circuit-switched network, how does this affect power and performance, and what support do we need from the electronic control network. The answer to this might depend on a number of factors, such as how large are we willing to make the electronic control network (buffer size, link width, etc), the topology, and the expected application behavior. In this section, we will briefly investigate how we might go about implementing selective transmission and the factors involved.

Let's consider the characteristics of applications and how they would use a NoC qualitatively. One common communication pattern is shown in Figure 3.27(a), where many small messages are used to signal control information, such as coherence or barriers, and the remaining larger data messages are distributed out in some way. One natural choice of selective transmission policy would be to set a size threshold such that all small control messages are sent on the electronic network, while all larger data messages are sent optically. The question then becomes how large do the electronic buffer and channel widths need to be in order to provide good performance at low power. The same question applies to the more general case, or for broader message size distributions such as the examples in Figures 3.27(b) and 3.27(c). However, the question of message size threshold is not necessarily clear.

Another possible selection policy would be to send messages which are destined for nearby access points on the control network, and leave long-distance communication up to the photonic network. Finally, a third policy we will consider is the combination of the two, where messages are selected based on the product of their size and distance to be traveled.

We will demonstrate an example of implementing selective transmission by simulating some scenarios. Instructions for running this simulation can be found in Appendix A.2.3. We will consider three policies: message size selection, distance selection, and both size-distance selection, all with different threshold values. We will run one of the previous synthetic benchmarks, Random, for determining source-destination pairs (or distance), and use the distributions in Figure 3.27 to dictate message sizes.



(a) Control (b) More Small (c) More Large

Figure 3.27: Potential probability distribution of message sizes for an application. (a) application containing many small control messages (b) application with more small messages than large (c) application with more large messages than small.

Finally, we will vary buffer size and channel width in the electronic control network to see the power-performance tradeoffs.

First, let's take a look at power consumption. As we saw before in Figure 3.21(b), power consumption for circuit-switched networks is consumed mostly in static power from the lasers and electronics, so we can just look at a single data point irregardless of network activity, selection policy, and selection threshold for a moment. Figure 3.28 shows the network power consumption for different buffer sizes and channel widths, ranging from 10.4W to 19.3W. The design choice as to which buffer size and channel width to use may be specifically constrained by power limitations, but is more likely dependent on how much performance can we get out of increasing these parameters (and thus power). In any case, to reduce the number of variables we are looking at in simulated results, let us pretend for a moment that we are limited to 15W of network power. These only leaves us with 5 buffer-width choices. We will choose a buffer size of 2176, and a channel width of 136, as it seems like a good improvement over buffer=1088 (double the size), and we don't pay too much more power for double the channel width (only about 1W).

For performance, lets first take a look at the Control-type message size distribution from Figure 3.27(a). Figure 3.29(a) shows the latency-bandwidth characteristics for distance, size, and size-distance based selection policies for uniform-random source destination pairs. We can see that distance-based selection actually *degrades* performance for higher message arrival rates, probably due to the possibility of large messages being transmitted on the electronic network, though not going very far end up blocking

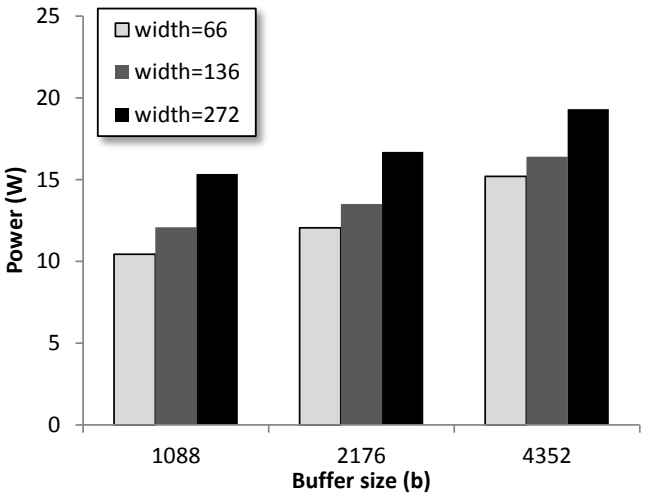
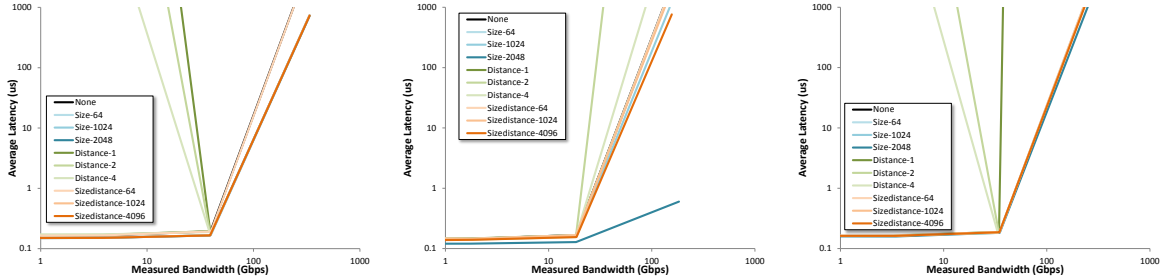


Figure 3.28: Power for different buffer sizes and channel widths.



(a) Control (b) More-Small (c) More-Large

Figure 3.29: Confusing graphs depicting latency and bandwidth characteristics for control-type message sizes with random destinations.

circuit-path setup messages. This is also true for the More-Large size distribution, found in Figure 3.29(c). While we could perform an extensive search of the design space including policy, threshold, buffer size, and channel width, this is best left to a designer with particular application traffic loads in mind, as the right design will mostly depend on that.

3.2.4 Off-chip Memory Access

One of the most important subsystems in a computer is the off-chip memory, and how data is moved to, from, and around the chip. Photonics has some unique potentials for IO communication because of the nature of the transport medium. Because waveguides are bit-rate transparent through the chip edge, no additional driver and receiver circuits are needed for off-chip communication. Furthermore,

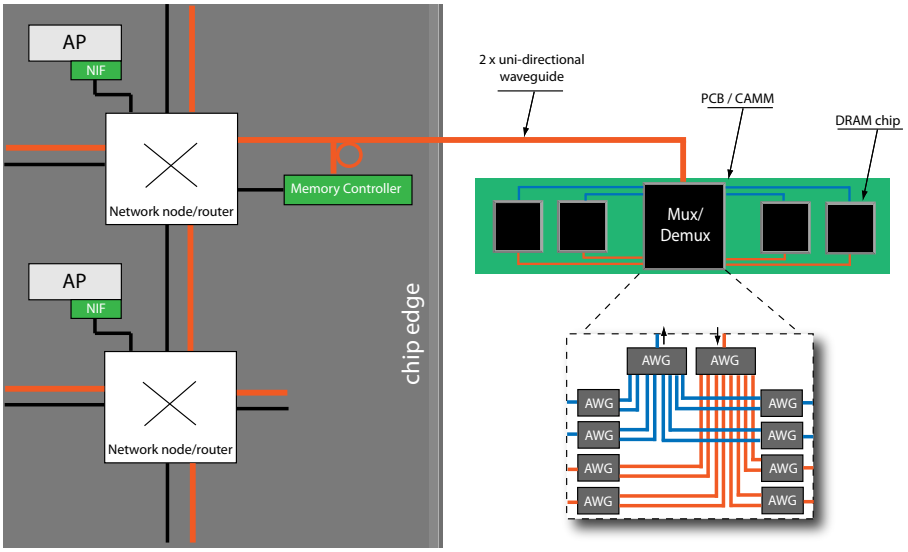


Figure 3.30: Anatomy of a circuit-switched memory link.

distance is much less of a factor in terms of bit rate and power when considering distances at the board-scale. Finally, WDM offers extremely hequirements. Memory subsystem technology, organization, performance, and protocols are all subjects of other books, but here we would like to briefly deigh bandwidth densities compared to electrical links which addresses many pin-count packaging challenges as well as increasing bandwidth rscribe how circuit-switched networks can accomplish core-memory communication. Hendry *et al.* (52) first proposed this by extending the same circuit-switched network resources and path-setup protocol out to the chip edge for DRAM reads and writes.

Figure 3.30 shows one possible implementation of circuit-switched memory access using lateral coupling to attach memory to a chip. A memory controller interacts with the electronic control network to establish end-to-end optical circuits between access points in the network and individual circuit-accessed memory modules (CAMMs). The memory controller issues row/column access commands to the DRAM chips using it's own modulators on dedicated wavelengths. CAMMs multiplex wavelengths into separate DRAM chips, allowing either one chip to be accessed at a time on a set of wavelengths, or all to be accessed in parallel.

Figure 3.31 shows the control flowchart for the memory controller, and how it interfaces the electronic path setup protocol with DRAM access commands. Only one transaction can take place at a time by virtue of the path-setup network protocol, a memory access paradigm significantly different from today's memory subsystems. Like circuit-switching itself, this design would be better suited for appli-

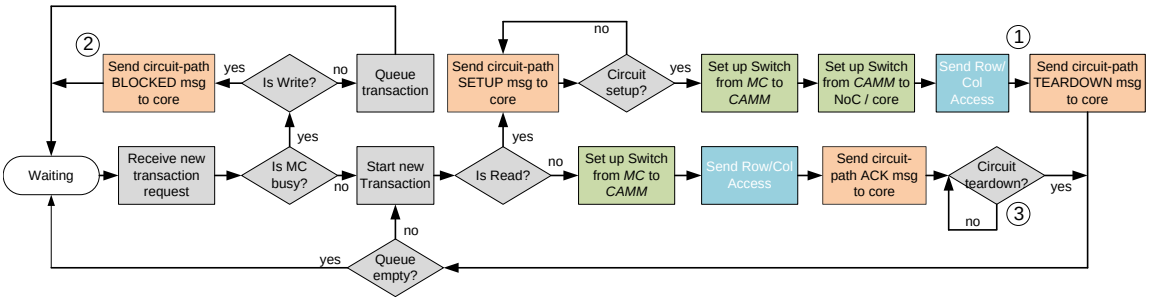
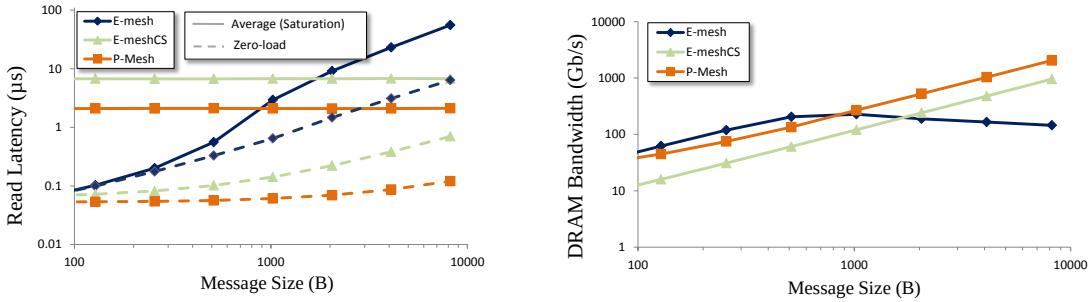


Figure 3.31: Control flowchart of circuit-switched memory controller, interacting with electronic control network.



(a) Read Latency

(b) DRAM Bandwidth

Figure 3.32: Characteristics of DRAM subsystem for packet-switched E-Mesh, circuit-switched E-Mesh, and photonic circuit-switched P-Mesh.

cations with large chunks of data, streaming, or otherwise static communication patterns requiring high memory bandwidth.

Though a full system application-specific analysis is beyond the scope of this book, we can provide some characterizations through simulation of the circuit-accessed memory subsystem design. Details on running this simulation can be found in Appendix A.2.4. Our goal is to approximately assess the effectiveness of both circuit-switched memory access and photonic technology itself. To do this, we compare an electronic packet-switched mesh to an electronic circuit-switched mesh to our regular photonic P-Mesh running random DRAM accesses with equal probability of read and write. Each network is 8×8, with a memory access point at each peripheral network node. The important metrics here are read latency for reducing CPU stall time, and total DRAM bandwidth.

Figure 3.32(a) shows the average read latency for both zero-load and saturation. Assuming messages in continuous parts of memory, which may be reasonable for some systems (*e.g.* embedded systems reading sensor data, stream-processing systems, or large cache/buffer lines) , the circuit-switched networks

are able to achieve significantly lower zero-load read latency, which gets more profound with larger messages. This is because the memory controller is not trying to optimize DRAM utilization by scheduling small accesses, but reads whole DRAM array rows at a time. Under contention, the packet-switched E-Mesh has much lower latency for smaller messages because of the path-setup overhead, though this is amortized in the circuit-switched networks for larger messages. The electronic circuit-switched network, though with similar characteristics to the P-Mesh, has significantly higher latency as message size increases because its bandwidth is pin-count limited. For total DRAM bandwidth, in Figure 3.32(b), the circuit-switched networks continue to scale with message size as expected, where the packet-switched version drops off as larger messages are just packetized and saturate the DRAM subsystem.

3.2.5 Time Division Multiplexed Arbitration

It may have been clear that, up to now, circuit-switching may not be the best choice for arbitrating optical links in a NoC. Though the links become very efficient at optically transporting data across a chip when they are established, the path-setup protocol does not have any notion of fairness, and thus can lead to degraded performance under high loads due to contention. To address this, we can consider circuit-switched networks which are arbitrated by other means. One way to do this is to implement a TDM-arbitrated and distributed control of photonic switches, first shown by Hendry (53). The basic concept behind this is as follows: during a specified amount of time, or time slot, switches in the network are configured to allow communication between one or more pairs of access points. Each time slot is of length

$$t_{slot} = t_{setup} + t_{transmission} + t_{propagation} \tag{3.11}$$

where t_{setup} is the time it takes to change the state of a ring resonator, $t_{transmission}$ is the time each node is allowed to transmit data per time slot, and $t_{propagation}$ is the worst-case propagation latency between any two valid communicating pairs. If each switch is able to keep track of the current time slot using a global clock, it can be made aware of its correct configuration using control registers for any given time slot. This allows the control of the switches to be completely distributed.

This concept should be distinguished from TDM mechanisms in other networks. Typically, requests

to use network resources are arbitrated by sources or individual network nodes to dynamically allocate a temporal schedule for access to virtual channels, physical links, switches, or virtual circuits, thus providing fairness guarantees to latency and bandwidth (54, 55, 56, 57, 58). Our method aims at providing the same fairness, but because there is no practical equivalent to a buffer implementation in silicon photonic integrated technology, we must apply TDM arbitration through the entire network by creating end-to-end optical circuit paths. Here, the scheduling of nodes' access to network resources is done statically, at design-time. If there are N_{slot} time slots, each of duration t_{slot}, then the total TDM period, T_{TDM} is

$$T_{TDM} = N_{slot} \times t_{slot} \tag{3.12}$$

3.2.5.1 Fully-Connected

Our first implementation of TDM stipulates that full network communication *coverage* must be implemented, or that every network node is able to send messages to every other node within T_{TDM}.

In this arbitration scheme, the network repeatedly cycles through every time slot. If a network node has data to send to another node, it waits for the correct time slot. If a node has multiple messages to different destinations queued up, it can send them out of order. Also, by statically selecting different values for $t_{transmission}$, we can vary the granularity of the arbitration. If, for instance, the system architecture specifies that only fixed-length messages may be sent on the network (*i.e.* cache lines), then we can adjust $t_{transmission}$ to exactly match that size.

The naive way to accomplish the resource scheduling is to assign a single time slot to every communicating pair in the network. Thus, we would require

$$N_{slot} = N \times (N-1) \tag{3.13}$$

time slots to implement full coverage, where N is the number of nodes in the network. A 64-node network would therefore require 4032 time slots. This naive scheduling of one transmission per time slot in the network achieves the worst-case network utilization, and is only described here as an example of a possible control schedule.

We can improve on the naive implementation by scheduling more than one transmission per time

slot, thus reducing the total number of time slots, and the worst-case latency of a message waiting for its slot. In order to maintain correct operation we must adhere to the following constraints during a single time slot:

1. **Source contention** - A node can only send to one destination, assuming a single set of modulators at an access point.
2. **Destination contention** - A node can only receive from one source, assuming a single set of detectors at an access point.
3. **Topology contention** - Transmission cannot overlap in the same waveguide.

Thus we are presented with the problem of scheduling in both time and space at least one transmission from every node to every other node in the network. We will not go into the details of how to do this here, but it has been shown that this can be accomplished by searching the solution space using a *genetic algorithm* (53). A 4×4 mesh, for example, requires only 18 time slots. Because a smaller network leads to both lower power (from less loss, thus less laser power) as well as better performance because less time slots are needed, it is wise to use concentration, discussed in Section 3.2.3.1 for a TDM network.

TDM Switch Some implementation changes from pure circuit-switching are necessary to consider for a TDM-arbitrated network. Figure 3.33 shows the layout for the photonic switch in the TDM network. It consists of waveguide paths and electro-optically controlled 200-μm ring resonator-based PSEs, which spatially switch a broadband signal. Ports are labeled as North, South, East, West, and Gateway (GW).

Because we optimized our arbitration assuming X-then-Y routing, the switch does not need to implement full connectivity between the ports. Table 3.2 shows the port combinations, and the PSE number that implements the path, referring to Figure 3.33. For example, we can see in Figure 3.33 that the PSE labeled as 1 can switch a signal from the gateway (modulator bank) to the north port. Note that the signal must pass through a ring only when coming from a gateway, entering a gateway, and turning from an X- to Y-dimension, saving on insertion loss when traveling in straight lines.

Switch Controller In the proposed network architecture, each switch is controlled by a local controller which is aware of the current TDM slot by tracking ticks of a global TDM clock, and is therefore aware

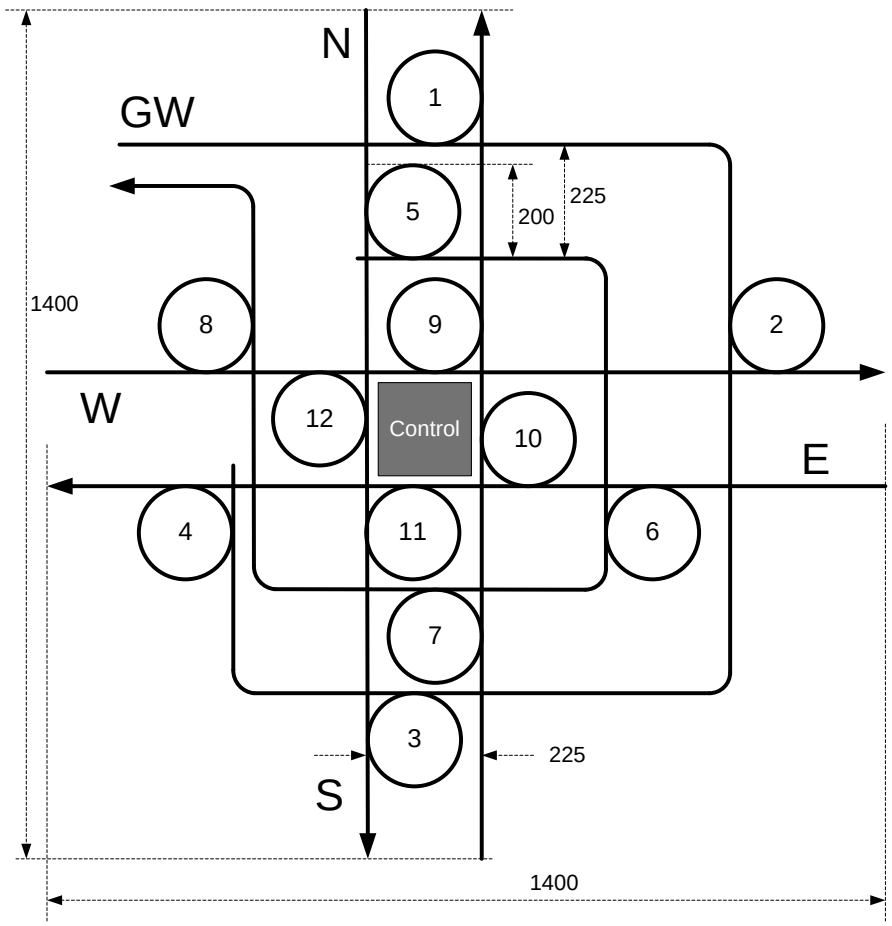


Figure 3.33: Layout of TDM photonic switch, showing waveguides and ring resonators. Units in microns.

Table 3.2: Switch Functionality

Inport	Outport	PSE
Mod	N	1
Mod	E	2
Mod	S	3
Mod	W	4
N	Det	5
E	Det	6
S	Det	7
W	Det	8
W	N	9
E	N	10
E	S	11
W	S	12
E/W	W/E	-
N/S	S/N	-

of how the switch should be set. A global, synchronous TDM clock can be implemented with waterfall clock distribution, synchronous latency-insensitive design (59), or optical clock distribution (60). The period of this clock must be the TDM period, t_{slot}. Clearly, t_{slot} should be set to an expected average message transmission time, so that time slots are just big enough to allow end-to-end transmission. Taking into account time of flight overhead, this value could be tens of nanoseconds equating to less than 250 MHz TDM clock frequency (depending on t_{slot}), a very feasible implementation by today's standards.

The output logic can be implemented as a single lookup table (LUT) which takes the switch ID register as an input, allowing identical ROM instantiation among network tiles. In practice, only the fraction of the table that is necessary to run the local switch would be instantiated to save area and power.

The size of the output logic is proportional to the number of TDM slots, which is dictated by the number of network nodes. Specifically, there is one bit per PSE per TDM slot, indicating whether the PSE is on or off. Since there are 12 PSEs per switch, this means the ROM of each switch controller contains $1.5 \times N_{slot}$ bytes of information.

TDM Network Gateway Figure 3.34 shows the microarchitecture of a network gateway, providing network and memory access to four cores. This is done by a main *TDM controller*, which arbitrates access to the network. The gateway operation consists of five main steps, numbered in Figure 3.34:

1. Communication requests are made to the TDM controller, which controls an electronic crossbar that connects the various gateway components.

2. When the network is in the correct TDM slot, the TDM controller sets the crossbar from the requesting core to the serializer, which ramps the data up to 10 Gb/s modulation. This bitrate clock is also transmitted on a seperate wavelength for data recovery at the receivers.

3. When a signal is received, it is first deserialized, clocked by the received transmission clock.

4. If the data has reached its destination, it sits in a temporary buffer, waiting for access to the electronic crossbar. Access will be immediately available unless cores in the same gateway are communicating locally through the crossbar.

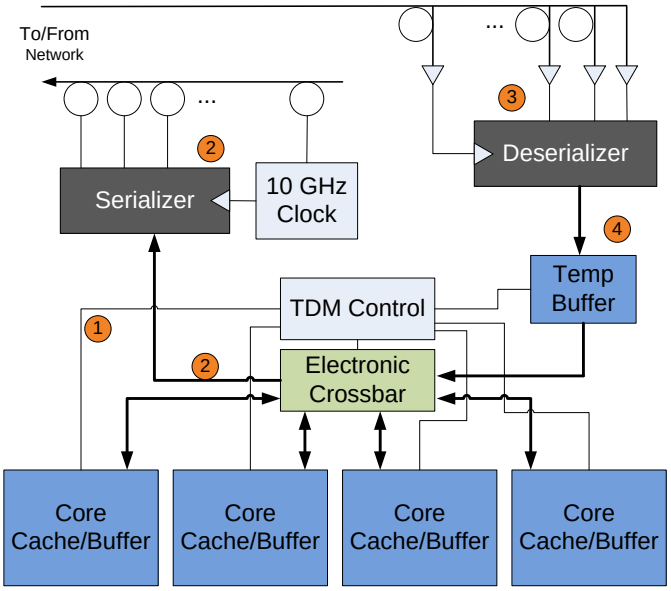


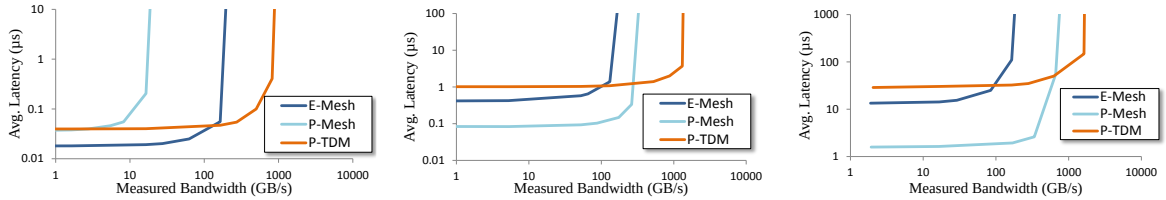
Figure 3.34: TDM network gateway microarchitecture.

The temporary buffer is only used to store received transmissions that are destined for the cores in the gateway. The TDM controller gives priority to the temporary buffer over local core-core communication, therefore it needs to hold a maximum of 3 transmissions: one for receiving incoming transmissions, one for sending the last received transmission on to the correct core, and one buffer in case of destination contention.

One important function of the gateway in our TDM architecture is to allow *out-of-order* access to the network from the cores, a property inherent in the TDM architecture. In other words, if a request from one core can be sent during a time slot, it does not have to wait for other requests from other cores that need other time slots to be serviced even if their requests arrive to the controller first. This property motivates the microarchitecture design decision to use concentration to increase the network utilization.

Evaluation We investigate a 4×4 TDM network with 4-way concentration running Random traffic, and compare it to an E-Mesh and P-Mesh. Details on running this simulation can be found in Appendix A.2.5.

Figure 3.35 shows the latency/bandwidth characteristics for small (256B), medium (8kB) and large (256kB) messages. In all cases, the TDM achieves higher bandwidth because it accomplishes its goal of providing some level of fairness, and is not dependent on message size like the P-Mesh is. However, it



(a) Size=256B (b) Size=8kB (c) Size=256kB

Figure 3.35: Latency and bandwidth characteristics for a full-coverage 4×4 TDM network concentrated to 64 access points for (a) 256B messages (b) 8kB messages (c) 256kB messages.

does suffer from higher zero-load latencies somewhat due to messages having to wait for their time slot to begin transmission.

3.2.5.2 Enhanced TDM

We aim to improve on the full-coverage implementation by decreasing the number of time slots required, described by Hendry (61). Instead of searching the solution space, we will simplify the problem and describe a method we can apply manually to a mesh topology for scheduling which is scalable and results in significantly fewer time slots.

To simplify the problem, let us first concede that photonic transmission will no longer be entirely end-to-end for every node pair. Rather, the mesh X-dimension transmission is first completed, converted to the electronic domain, and stored in a buffer until the Y-dimension transmission can be completed. This means that we will pay optical to electrical conversion energy costs twice. This simplification reduces the energy per bit benefits that end-to-end photonic transmission technology provides, which we will investigate momentarily.

We can first observe that two transmissions can always take place in a row during a time slot, for any size row, where the two sending nodes are on opposite sides of the row. This is illustrated in Figure 3.36 for one row of four nodes, assuming bidirectional links connecting neighboring nodes consisting of two uni-directional waveguides. The red nodes are the sending nodes, and exhaust all possible combinations of destinations (green) in the row. The process repeats for all other nodes being designated as the sending nodes. Note that communications are shown symmetric across the midpoint of the row in Figure 3.36, though this is not required.

We now make it our goal to schedule communications similar to Figure 3.36 such that two transmis-

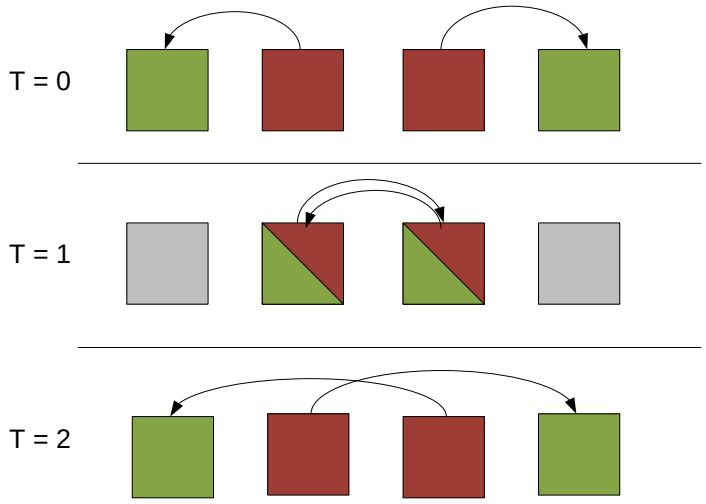


Figure 3.36: Row communication TDM slots example for four nodes.

sions occur in every row *and* every column in each time slot. Since each node in a row must communicate with every other in its row (R-1 of them), and two nodes are communicating at once per row, we would require

$$N_{slot} = (R-1) \times (\frac{R}{2}) \tag{3.14}$$

time slots, where R is the number of nodes in a row (and column, assuming a square network), R is even, and $R \geq 4$. For an 8×8 64-node network, this is merely 28 time slots, a significant improvement over the previous end-to-end implementation with 142 time slots (53).

Figure 3.37 illustrates an example of how to schedule a 4×4 TDM network, which requires 6 time slots. We represent the transmission possibilities as a 16×16 *control matrix*. Each entry in the matrix is color-coded to indicate which sender-receiver pair is enabled during a time slot. Note that a node may only send and receive once per time slot, which translates into the rule that a color may only appear once in a row and column in the control matrix. Also note that not all node combinations are necessary because we conceded that optical circuit paths only travel in one mesh dimension during a slot, which is why many control matrix entries are blank (white).

Some visual and numerical patterns are useful when specifying the control matrix for any size network. For instance, the 4×4 squares lying on the black diagonal indicate row communications. Other

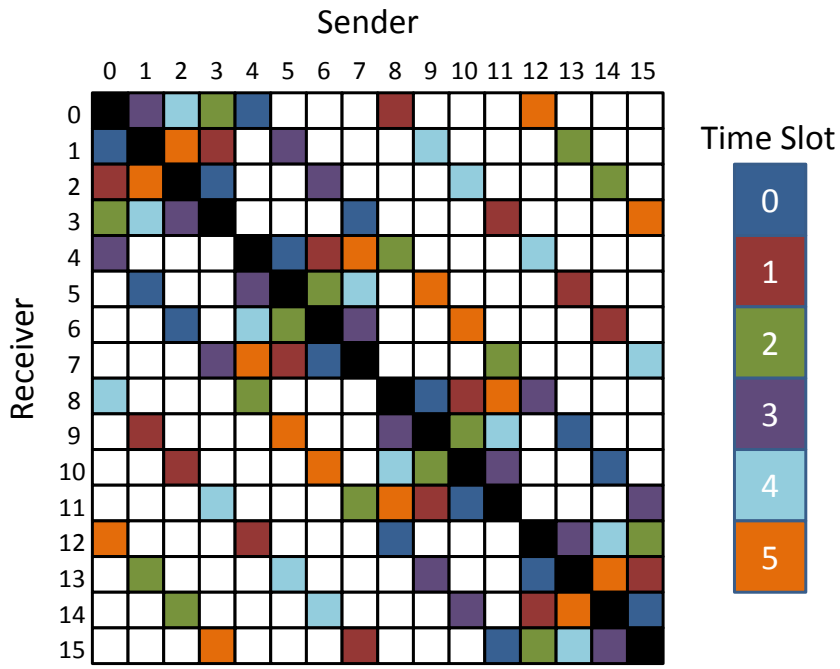


Figure 3.37: E-TDM control matrix for a 4×4 network.

diagonal stripes represent column communication. First, all row communications are added in, each block (row) utilizing every time slot exactly twice, per Figure 3.36. The block pattern shifts slightly to accommodate column communications, and is mirrored across the network bisection line (row R/2).

ETDM Switch For Enhanced-TDM, some slight changes are required over the original TDM design. Figure 3.38 shows the layout for the photonic switches in the network.

Because we optimized our arbitration for fewer TDM slots at the cost of paying O-E-O energy by doing X-then-Y routing, the switch does not need to implement as many paths as the original TDM switch. A table similar to Table 3.2 could be constructed, though it should be clear how the switch operates. Note that the signal must pass through a ring only when coming from a gateway and entering a gateway, which saves on insertion loss when traveling in straight lines.

Network Gateway The ETDM gateway microarchitecture is also slightly different, and we also add in the ability to access memory from each gateway. Figure 3.39 shows the microarchitecture, providing network and memory access to four cores. This is accomplished through the use of a main *TDM controller*, which arbitrates network and memory resources and acts as a memory controller by keeping a

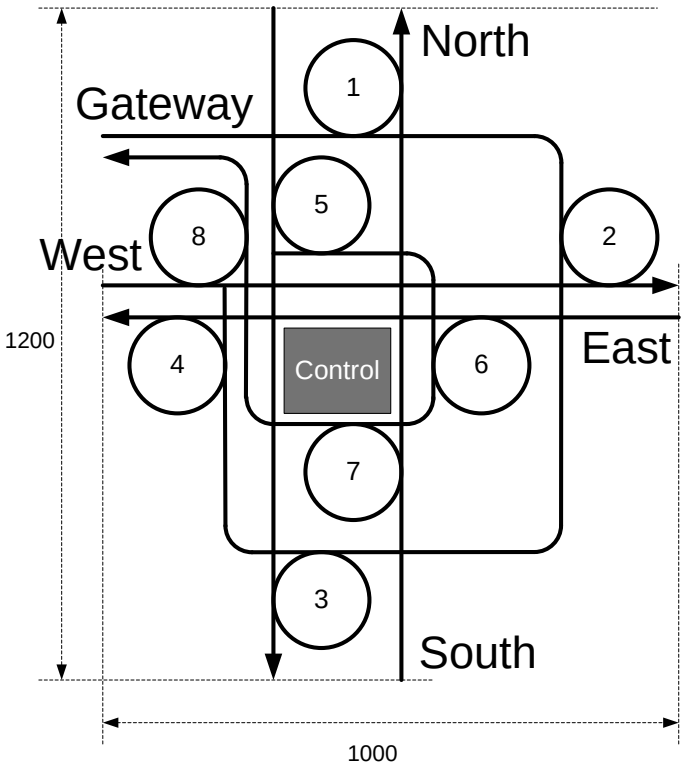


Figure 3.38: Layout of ETDM photonic switch, showing waveguides and ring resonators. Units in microns.

master schedule of events that occur during each time slot.

Each gateway has two vertically-coupled (62) connections to a memory bank. Local reads and writes are serviced by scheduling row and column accesses during free slots in the master schedule. Remote memory accesses are sent to the destination gateway, where they are then scheduled in a similar fashion. Remote reads are read directly from memory into the network to save on buffering power.

The following describes an example of the gateway operation, numbered in Figure 3.39:

1. Communication requests are made to the TDM controller, which controls an electronic crossbar that connects the various gateway components.

2. When the network is in the correct TDM slot, depending on the type of communication (memory read, memory write, MPI-send, etc), the TDM controller sets the broadband rings that control access to and from the modulators and detectors. This can also be done ahead of time when the time slot switches, if the transaction has been queued up.

3. The TDM controller also sets the crossbar from the requesting core to the serializer, which ramps the data up to 10 Gb/s modulation. The transmission clock is also transmitted on a separate wavelength.

4. When a signal is received, it is first deserialized, clocked by the received transmission clock.

5. If the data has reached its destination, it sits in a temporary buffer, waiting for access to the electronic crossbar. Access will be immediately available unless cores in the same gateway are communicating locally through the crossbar.

6. If the data is using the gateway as an intermediate point while switching dimensions, it sits in the X-Y buffer and notifies the TDM controller. It can then transmit during the correct TDM slot.

The sizes of the buffers can be exactly specified based on the size of the network. The X-Y buffer is used to hold transmissions that have arrived at this gateway to continue through the network in a different direction, and are waiting for their time slot. Therefore, they must hold a maximum of $2 \times (R-1)$ transmissions, which is the number of time slots in one TDM frame in which a message could be

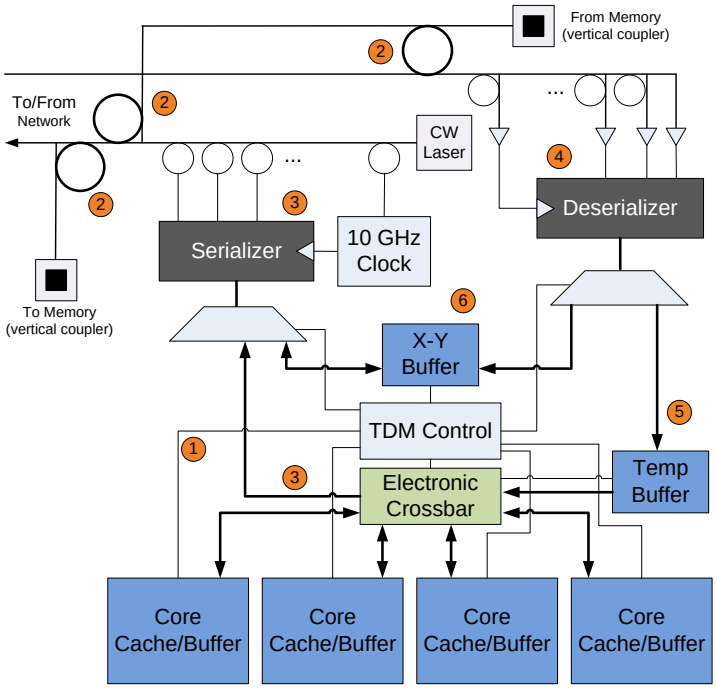


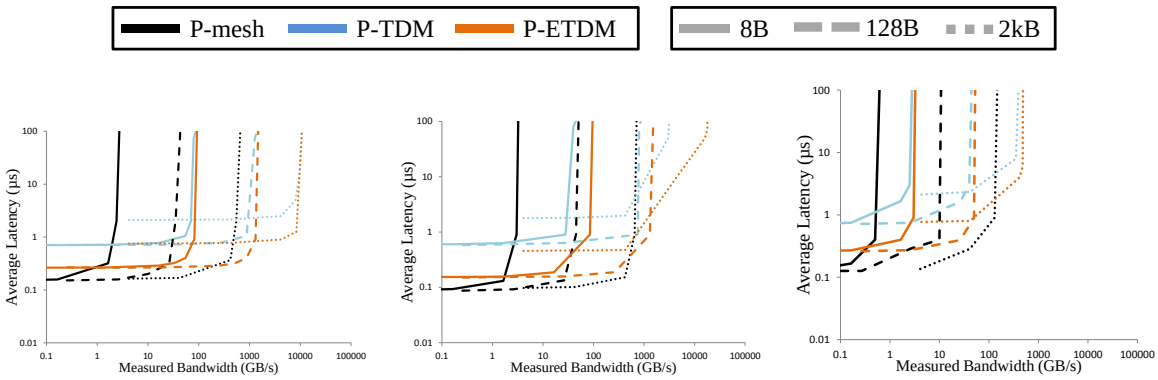
Figure 3.39: ETDM network gateway microarchitecture.

received. A 64-node network will therefore require a buffer of size $14\times$ $S_{transmission}$, where $S_{transmission}$ is the maximum message size that can be transmitted in one time slot.

The temporary buffer is only used to store received transmissions that are destined for the cores in the gateway. The TDM controller gives priority to the temporary buffer over local core-core communication, therefore it needs to hold a maximum of 2 transmissions: one for receiving incoming transmissions, and one for sending the last received transmission on to the correct core.

Evaluation To test the ETDM network characteristics, we use PhoenixSim to run Random, Tornado, and Hotspot traffic in the network for 5 ms with 8, 128, and 2k-byte messages, representing control, cache-line, and application-level message sizes, respectively. We set t_{slot} at 10 ns, requiring 1 ns each for t_{setup} and $t_{propagation}$, making $S_{transmission}$ equal to 10240 bits, or about 1.2 kB.

Figure 3.40 shows the average read latency vs. total bandwidth in the network. The two TDM networks show higher zero load latency than the P-mesh, as expected from the overhead of waiting for the correct slot. However, the enhanced TDM network shows significant zero-load latency improvement over the original TDM design. Both TDM networks also show higher throughput compared to the



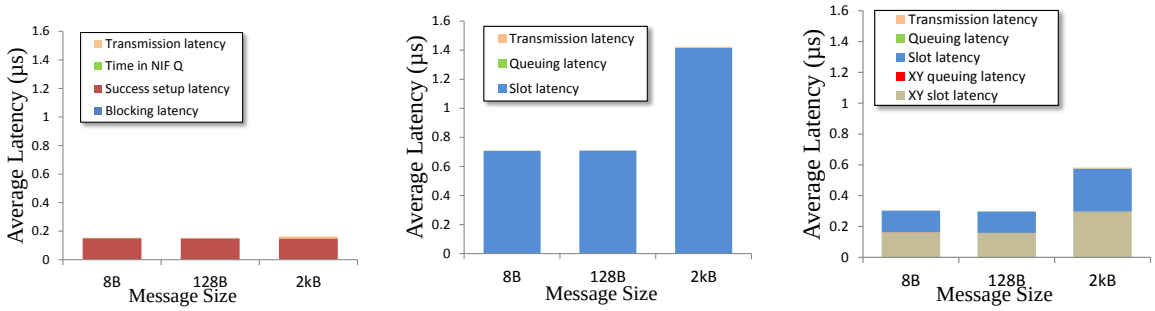
(a) Uniform (b) Tornado (c) Hotspot

Figure 3.40: Latency / bandwidth characteristics of ETDM under synthetic traffic for (a) Random (b) Tornado (c) Hotspot.

P-mesh for all message sizes, due mostly to their ability to service message requests that arrive to the gateway's controller out of order, thus increasing network utilization. Bandwidth gains are most profound in the traffic patterns with more chances of circuit-path blocking in the P-mesh, either from long communication (Random) or predictably conflicting resources (Tornado).

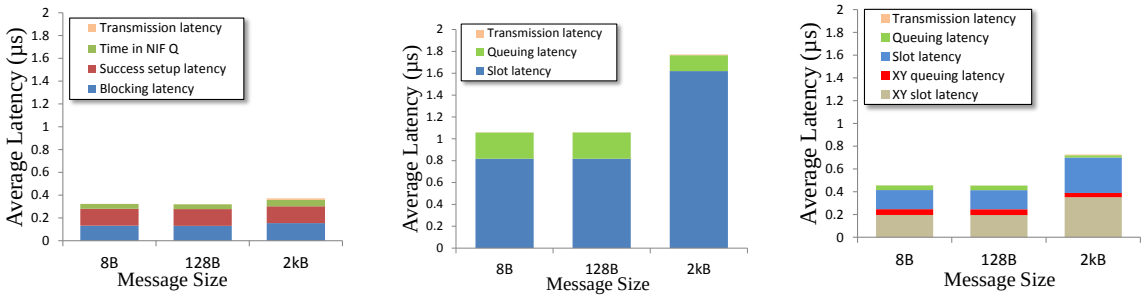
Figure 3.41 shows the sources of zero-load latency under Uniform traffic for each network as message size increases. The Pmesh is superior in this respect, as it is entirely dependent on the electronic router hop latency. the original TDM design's latency comes entirely from the slot latency, or when a message is next in line for a time slot, but is waiting for that slot. Again, our design improves the zero-load latency over the original TDM design by descreasing the time slot count, despite additional delay when changing dimensions (XY-buffer queuing and slot latency). The TDM networks also show a significant increase in latency for the larger 2kB messages because the message must be sent in multiple slots. Though the slot period could have been changed to match the message size for the different simulations, we chose to keep a single slot period to illustrate the effects of its relationship to expected message size.

To illustrate the effects of contention on network latency, Figure 3.42 shows the sources of latency while loaded at half capacity. For the P-mesh, blocking latency enters the picture, forcing queuing at the network gateways. The original TDM design is still dominated by slot latency, where queuing latency is dictated by the traffic pattern. The ETDM method has a similar relationship, though must less



(a) P-mesh (b) P-TDM (c) P-ETDM

Figure 3.41: Zero-load latency breakdown under Uniform traffic.



(a) P-mesh (b) P-TDM (c) P-ETDM

Figure 3.42: Half-load latency breakdown under Uniform traffic.

severe because of the reduced slot count. An extra traffic-dependent queuing latency is introduced at the XY-buffer, though it is small compared to the total.

Figure 3.43 shows a coarse network power breakdown under Uniform traffic near saturation, assuming around 12% integrated laser efficiency (63). Electronic power is still a large part of all the networks, mainly in the electronic crossbar necessary to implement external concentration, which must match the bandwidth of the photonic links using many parallel wires. The TDM control circuitry contributes minimal power overhead to the two TDM networks. An advantage of the E-TDM network is that is has less insertion loss, and therefore requires less laser power. Instead of laser power, the P-ETDM consumes power in the XY-buffer (~2W) which is necessary to implement dimension-only transmission. Regardless, the P-ETDM consumes the lowest total power.

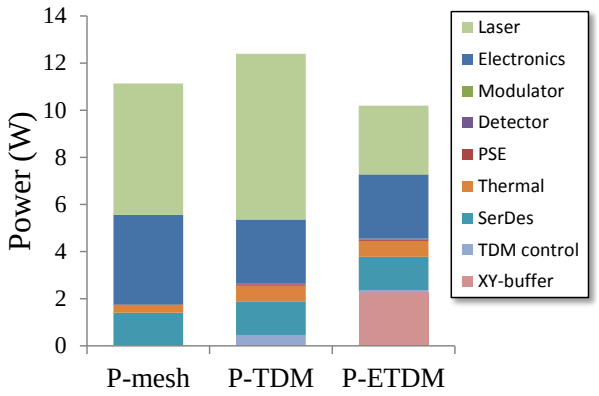


Figure 3.43: Power breakdown.

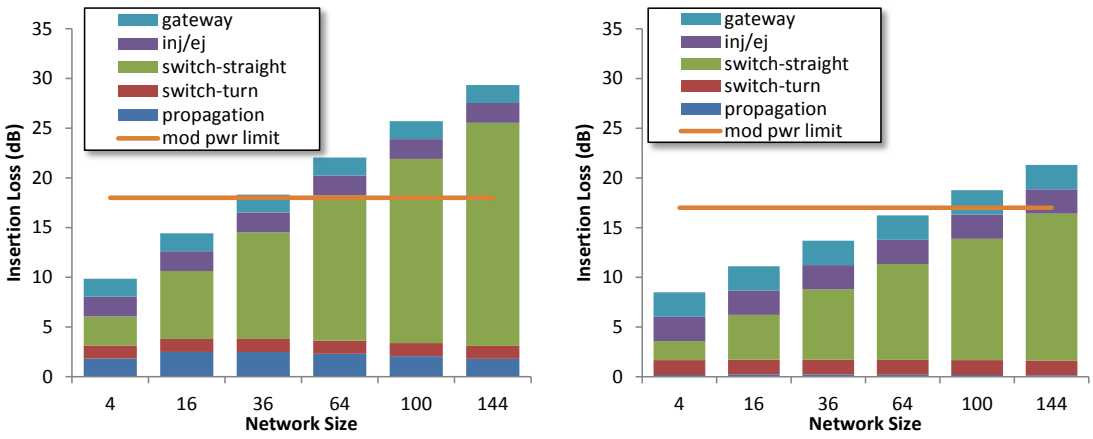
3.2.6 Impact of Deposited Multi-layer Devices

Multi-layer devices using deposited materials are important to consider for photonic NoC design for their potential in reducing insertion loss. Though not as versatile as pure crystalline silicon, materials such as silicon nitride (Si_3N_4) and polycrystalline silicon can be combined to not only implement the same functions, but add extra dimensions of design freedom in photonic layout. For circuit-switched networks, as we will see, the benefits of multi-layer devices manifest themselves mainly in eliminating waveguide crossings which, if you recall from Figure 3.15 can be quite significant. This kind of work was first investigated in Biberman *et al.* (64).

3.2.6.1 Multi-layer Mesh

Lets take a look at the insertion loss of a photonic mesh using multi-layer deposited materials. This analysis is actually a part of an optimization example discussed later in Section 4.2.1.1 (where we show an approximate expression for loss), though we present a simplified part of it here to show the architectural implications of multi-layer devices.

Figure 3.44 shows the insertion loss for various size networks with a constant channel spacing of 1.3 nm. Networks larger than 144 nodes do not continue to fit on a $2cm \times 2cm$ chip, so they are not shown. The multi-layer networks save the most insertion loss in propagation through low-loss silicon nitride and reduced crossings in traversing through switches, even though the drop loss of a poly-Si switch is larger. Clearly, multi-layer devices are key in either reducing laser power or increasing the scalability of



(a) Single-layer (b) Multi-layer

Figure 3.44: Insertion loss of single- and multi-layer photonic mesh.

circuit-switched or otherwise spatial networks, in this case enabling an 8×8 network.

3.2.6.2 Matrix-Crossbar

Now that we have lower insertion loss due to the elimination of crossings, we can also start to think about topologies that were previously not possible due to a large amount of loss. A final topology that we can explore is a full crossbar made from matrix-style switches, which can be seen in Figure 3.45. This topology utilizes fully non-blocking switches which, when coupled with a one-hop electrical setup network such as a flattened butterfly, can be used for both low setup latency and high-bandwidth optical circuits. One drawback is that it utilizes a large number of broadband rings, O(N^2), so we must make sure that they can all fit on a chip as follows:

$$S_{chip} \geq N \cdot M \cdot P_{wg} + R_{bend} + N \cdot M \cdot (P_{ring} + D_{ring}) + L_{taper} \tag{3.15}$$

Here, S_{chip} is the size of the chip dimension (2 cm), P_{wg} is the waveguide pitch (20 μm), R_{bend} is the bending radius for the end $\subset$-waveguide (20 μm), P_{ring} is the microring resonator pitch (30 μm), D_{ring} is the ring diameter, and L_{taper} is the length of the input taper, or optical terminator. Rearranging, we get a maximum microring resonator size of:

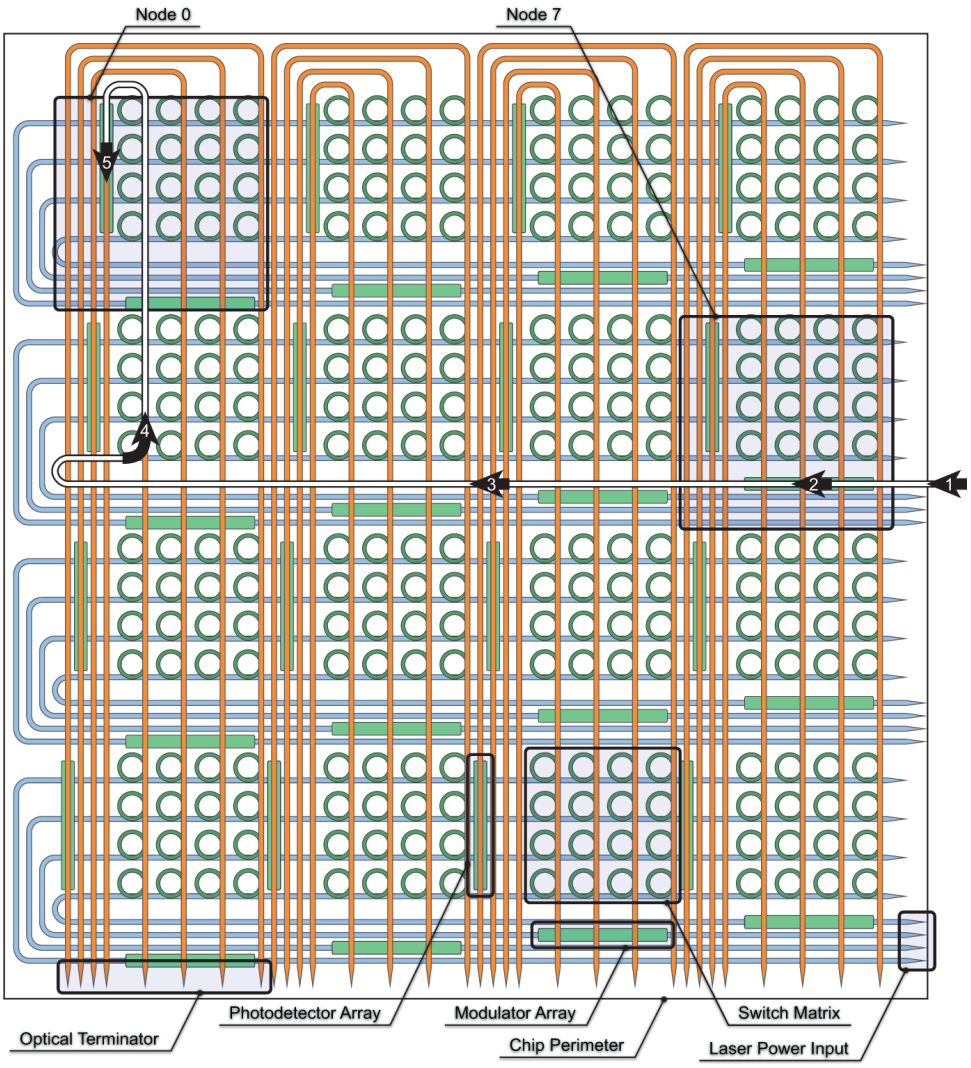
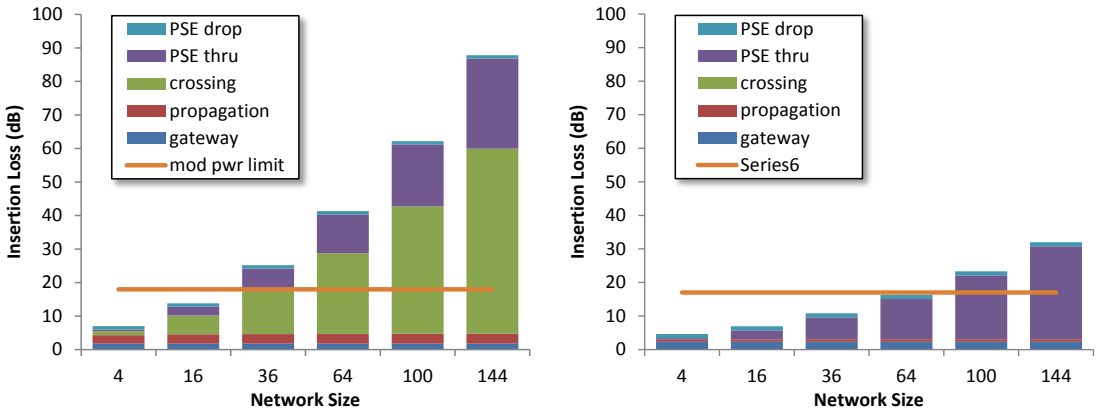


Figure 3.45: A 4×4 example of the matrix-crossbar network topology (not drawn to scale).



(a) Single-layer (b) Multi-layer

Figure 3.46: Circuit-switched matrix crossbar insertion loss analysis versus network size.

$$D_{ring} \leq \frac{S_{chip} - R_{bend} - L_{taper}}{N \cdot M} - P_{waveguide} - P_{ring} \tag{3.16}$$

For example, for an 8×8 network, we would have a maximum microring resonator diameter of 261 μm. Larger microring resonator sizes can be achieved with Archimedean spiral rings to significantly compact the size (65).

We perform the same insertion loss analysis on this new topology using a channel spacing of 1.8 nm, resulting in Figure 3.46. Similar to the non-blocking torus, this new topology is only made feasible using deposited materials by reducing the insertion loss due to the elimination of waveguide crossings, despite higher switch drop loss and poly-nitride coupling loss.

3.3 Wavelength-Arbited Architectues

Wavelength-arbited, or wavelength-routed architectures use the fact that the photonic medium, mainly the waveguide, can support multiple wavelengths (WDM) and can selectively filter out some or all wavelengths to different destinations. This can be accomplished in a number of different ways, as we will see in this section.

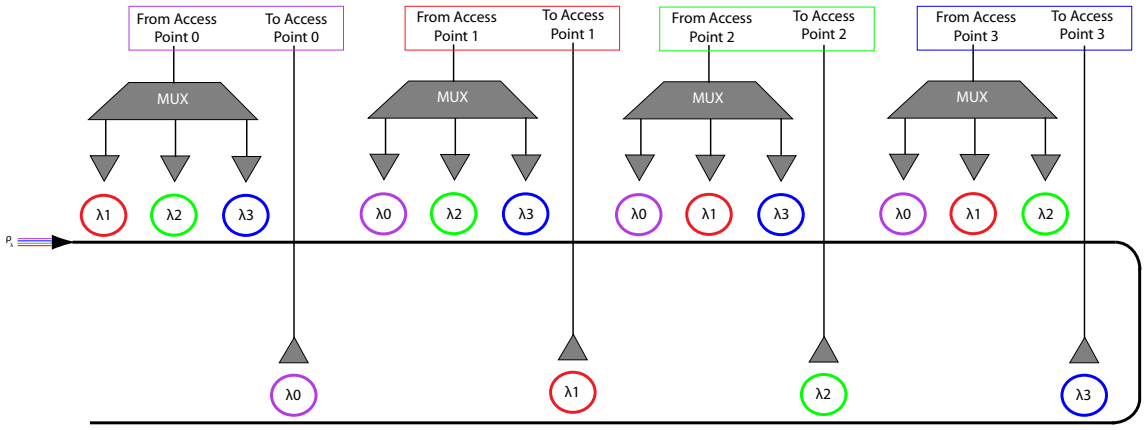


Figure 3.47: Source-routed bus.

3.3.1 Wavelength Bus Structures

The wavelength bus is the main unit of data transportation, usually consisting of waveguides which pass along all access points in a network. Various building blocks are typically used in wavelength-arbited designs, including destination-routed buses, source-routed buses, Multi-Write Single-Read (MWSR) buses, Single-Write Multi-Read (SRMW) buses, and a token arbitration ring. These structures are described in the following subsections.

3.3.1.1 Source-Routed Bus

The source-routed bus, in its pure form, is a configuration where each access point reads from a single channel associated with it. Any other access point can write to this channel and conversely, each access point can write to any access point's channel. Figure 3.47 shows how this would be implemented with microring modulators and filter/detectors, using N^2 rings, where N is the number of access points, and N wavelength channels. More bandwidth can be achieved by using more wavelengths for each access point's channel, though this is limited by the scale of the network, the channel spacing, and the modulator FSR.

It may be clear, however, that two access points can't write to the same destination at the same time, and must therefore arbitrate for access first. This could be done using a separate arbitration network, such as the token ring in Section 3.3.1.6. Once arbitrated, all other access points must turn their modulators OFF-resonance to allow previous access points' signals to pass through. For example, referring to Figure

3.47, if AP 0 was writing to AP 3, once arbitration has occurred and AP 0 has been granted access to AP 3's (green) wavelength channel, AP 1 and 2 must make sure their modulators associated with the green wavelength are OFF-resonance, else they would interfere with the signal before it reached AP 3's filter/detector.

This leads to the conclusion that the modulators are best designed to be OFF-resonance by default, which also implies that optical power is constantly reaching the detectors under no activity. Coding or other data-valid signaling is assumed to properly detect data. Also, the detector/receiver circuits must be designed for low leakage power when amplifying a logical 1 to avoid burning unnecessary power when the network is inactive. Also, the design of the modulators is a key factor for loss and power. The default off-resonant wavelength of the modulator must be spaced far enough from the laser wavelength channel so that significant loss is not experience when passing by all the other, but doing so both require more modulator power and larger channel spacing, which limits the scalability.

3.3.1.2 Destination-Routed Bus

Another way to connect access points on a wavelength bus is to use a destination-routed bus, which is in a way the opposite of source-routing in both function and implementation. Figure 3.48 shows the implementation using ring resonator modulators and filter/detectors. In this design, an access point modulates a single wavelength channel, and its intended destination selectively reads it. Like the source-routed implementation, this method uses N^2 rings and N wavelength channels. In order to determine the intended destination, arbitration could occur *a priori*, as with the source-routed bus using the token ring in Section 3.52, where each access point's filters are OFF-resonance by default, and are tuned ON-resonance once they are notified as such. Again, as with the modulators in source-routed buses, there is a tradeoff between the filter through loss, channel spacing, and filter tuning power.

Another way to arbitrate the destination-routed bus is to use filters which are designed such that they tap off a slight portion of the power so that all access points receive the same transmission in a broadcast-and-select fashion. A destination ID tag is compared to each access point's destination, and the message is thrown away if it was not intended for it. Clearly, this requires much more power per wavelength than the other methods, because enough power must make it into each detector for this method to work. Also, in practice, this method may be much more difficult because the ring filters will likely be designed

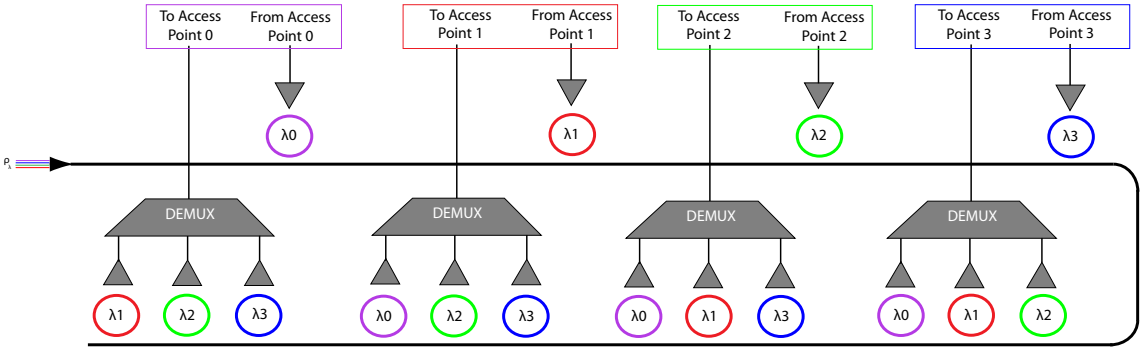


Figure 3.48: Destination-routed bus.

such that they are just slightly off resonance with the wavelength, and thus tapping only a small part of the optical power. Small drifts in resonance of any of the filters could therefore highly disrupt the whole bus.

3.3.1.3 Multi-Write Single-Read

A third way of connecting access points together is using a multi-write, single-read bus. Though not strictly wavelength-routed, networks designed to use this structure often have a similar serpentine or other layout which passes by every access point. The configuration can be seen in Figure 3.49. Each destination has its own waveguide, with a single set of filters where every other access point can transmit on the same wavelengths. Note that Figure 3.49 only shows the waveguides for AP3 (top line) and AP2 (bottom loop). AP0 and AP1 would have separate waveguides, with each other access points having sets of modulators to send to them. Clearly, MWSR must also be arbitrated *a priori*, and must have default OFF-resonant modulators. In the MWSR scheme, however, bandwidth is more easily achieved through WDM because the wavelength domain is not used for routing, such as in Figure 3.49 where 3 wavelengths are used for increased bandwidth. The MWSR scheme uses M wavelengths (where M is the number of wavelengths used for bandwidth) and N waveguides each with $N \times M$ rings, for a total of MN^2 rings.

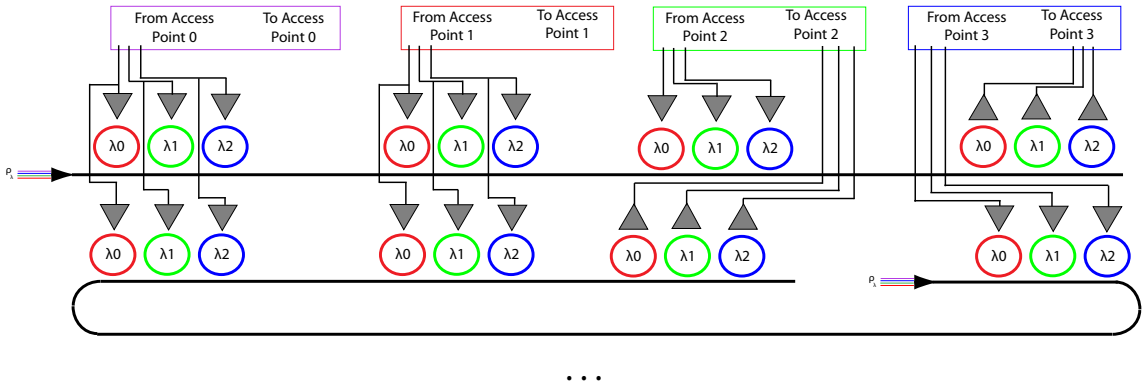


Figure 3.49: Multi-Write Single-Read bus implementation with ring modulators and filters.

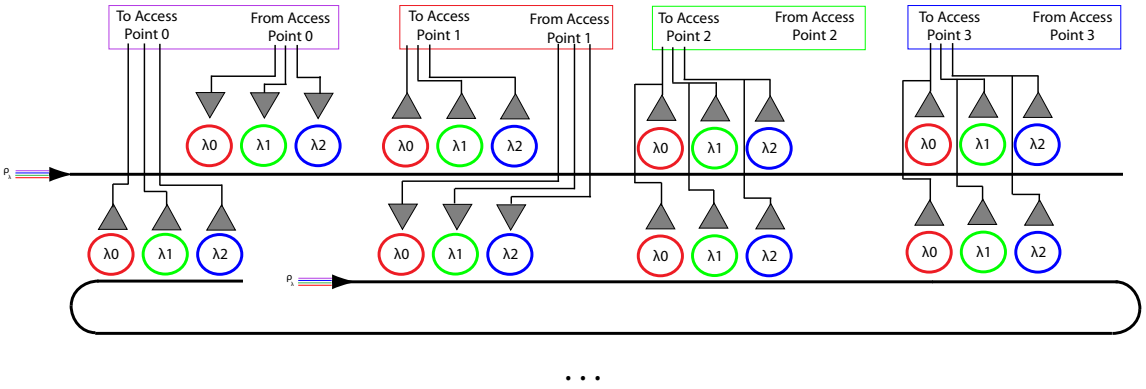


Figure 3.50: Single-Write Multi-Read bus implementation with ring modulators and filters.

3.3.1.4 Single-Write Multi-Read

The opposite of MWSR is the single-write multi-read, found in Figure 3.50. Like MWSR, SWMR has a separate waveguide per access point which passes through all other access points. In SWMR, however, each waveguide has one set of *modulators*, and the detector bank is turned ON-resonance once arbitration has signaled the destination. Again, SWMR uses $M \times N$ rings on N waveguides for MN^2 rings. The only advantage gained in using SWMR buses as opposed to MWSR is if the filters can be designed such that they only tap off a small part of the power, allowing broadcast-and-select functionality which eliminates the need for arbitration.

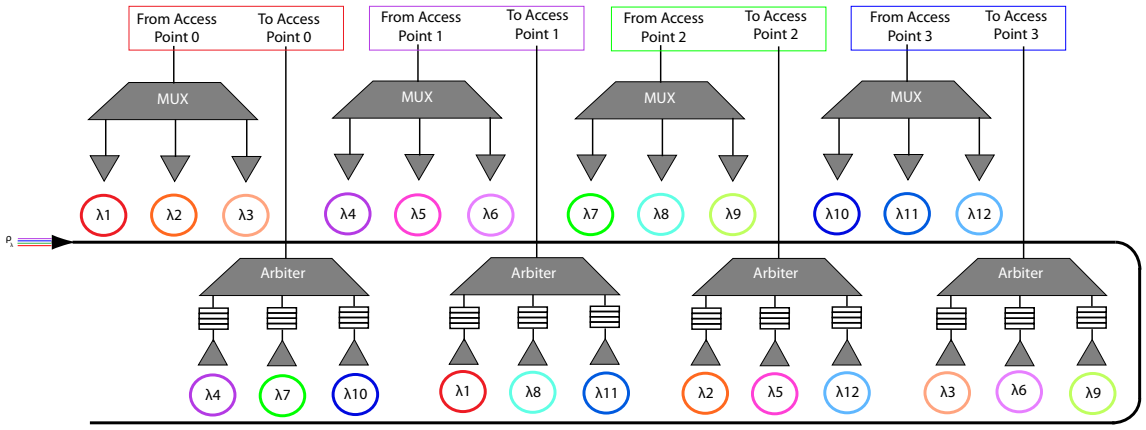


Figure 3.51: Wavelength crossbar connecting 4 access points using ring modulators and filters.

3.3.1.5 Wavelength Crossbar

One down side of the above methods is that they require *a priori* arbitration (or a difficult implementation for the SWMR). To avoid having to arbitrate, we can combine the source- and destination-routed techniques into a full wavelength crossbar, shown in Figure 3.51. Here, each source sends on a different wavelength depending on the destination, and each destination receives on a different wavelength depending on the source. Because an access point could now receive transmissions from two different sources, messages must now be buffered and arbitrated, assuming a single data bus going to the access point. This implementation also uses more ring and wavelength resources, requiring $2N \times (N-1)$ rings and $N \times (N-1)$ wavelengths. Finally, unlike the above implementations, the wavelength crossbar does not require any tuning or retuning of modulators or filters, a large advantage in design complexity of the tradeoff between power, loss, crosstalk, and channel spacing. However, the crossbar does require many wavelengths: $N(N-1)$ for an N-node network. For any reasonable sized network, it is not feasible to put this many wavelengths on the same waveguide. Many designs, including some discussed in Section 3.3.2, are designed around this limitation.

3.3.1.6 Token-Ring

One final structure for implementing network functions in the wavelength domain is the optical token ring, first proposed by Vantrease (66). This structure, found in Figure 3.52, allows access points to pass around optical tokens, one on each wavelength. Each optical token is designated a ”home” access

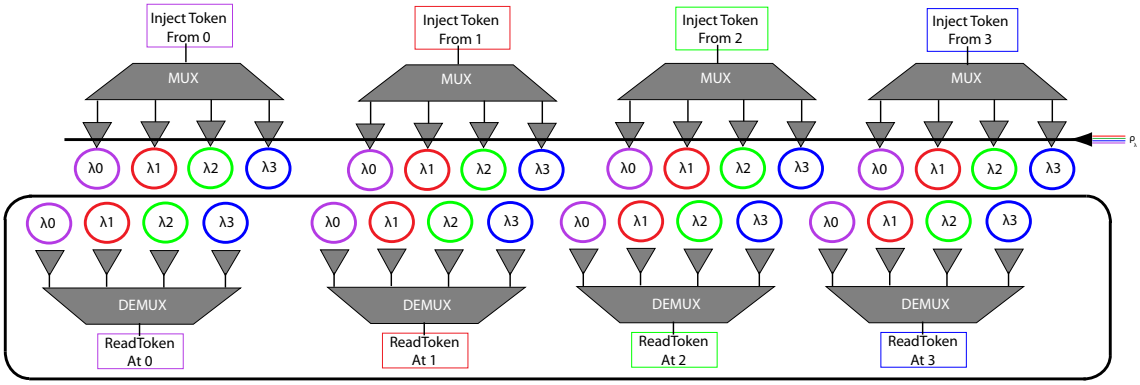


Figure 3.52: Token ring wavelength bus for optical arbitration.

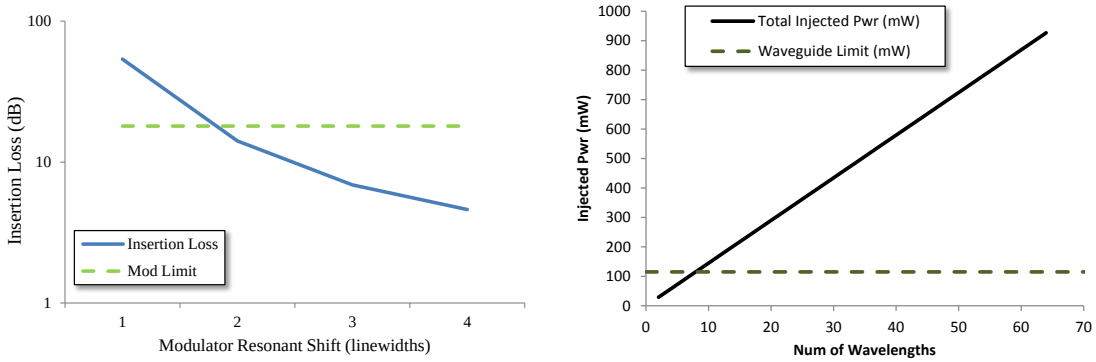
point, which usually indicates some arbitration information. For example, AP0 injects λ_0 (purple) from the power/source waveguide. If AP2 wants to send data to AP0, it turns its λ_0 filter ON-resonance to capture the optical signal. If AP2 is receiving λ_0, that means that no other access point can receive the token because the read is destructive, and therefore AP2 has exclusive access to AP0. Once AP2 is done sending to AP0, it can release the token by re-injecting it from the source waveguide. By default, each access point has the filter for its own wavelength channel ON-resonance, so that it can re-send its own token if no access point wanted it in one cycle. The token ring requires $2N^2$ rings and N wavelengths, and requires tunable filters for both the token-inject logic and the token-request logic.

3.3.2 Analysis of Wavelength-Arbited Architectures

In this section, we will take a look at how different wavelength-routed architectures perform. Our default network size will be 8×8 for comparison, unless otherwise stated.

3.3.2.1 Corona

The Corona network (66) is one example of a wavelength-routed architecture which uses an optical token ring to arbitrate a massive MWSR crossbar implemented with one million rings and 256 serpentine waveguides. Because it uses MWSR buses, all modulators must have their OFF state (not injecting carriers) as OFF resonance, so that modulated signals from other access points can pass by on their way to the photodetector bank. One critical design choice for these devices is how large the spectral shift should be between the OFF-state modulator resonance and the wavelength channel. The farther away



(a) Insertion Loss

(b) Injected Power

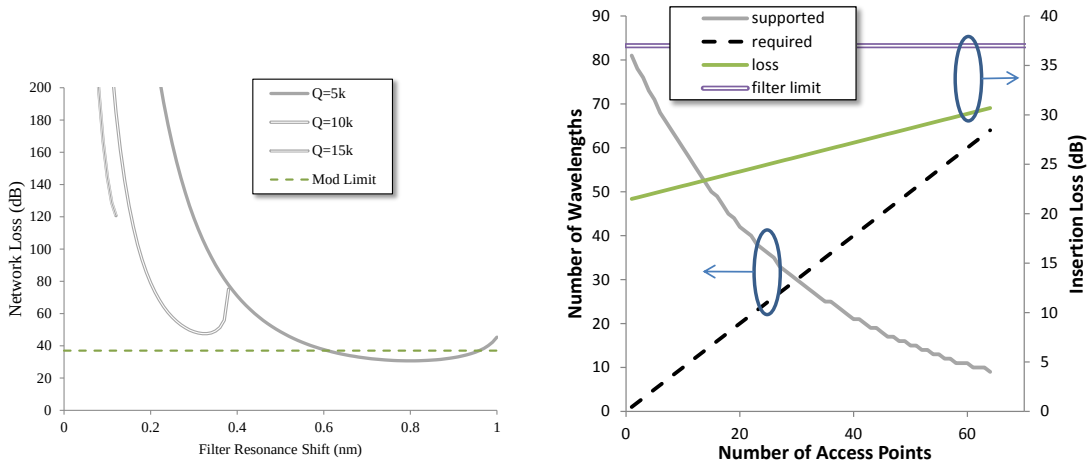
Figure 3.53: Insertion loss analysis for Corona's crossbar network. (a) Worst-case insertion loss versus modulator resonance shift in linewidths. (b) Injected optical power required versus number of wavelength channels, limited by the nonlinear threshold.

the resonant peak, the lower the OFF-resonance through port loss (from the tail of the resonance), which is critical for a MWSR bus, but more power is required since the resonance is shifted farther.

First, we will take a look at an analysis of the insertion loss in Corona's crossbar network, the details for which can be found in Section 4.2.2.1. Figure 3.53(a) shows the worst case insertion loss along one crossbar path for different amounts of resonant shift (in modulator response linewidths). As shown by the modulator optical power limit, a resonant shift of at least 2 linewidths is needed for the network to be feasible without incurring too much off-resonance insertion loss.

Adhering strictly to Corona's design with four bundled waveguides per channel and a single power waveguide sourcing each channel, Figure 3.53(b) shows required injected power to source the entire crossbar with three-linewidth shift modulators. Since we chose appropriate modulators such that the insertion loss falls under the modulator power optical limit, the total injected power in the power waveguide will ultimately limit the number of wavelength channels that is able to be used in each crossbar waveguide. Because the nonlinear threshold in silicon is around 110 mW, Corona's crossbar can only support 7 wavelengths per waveguide.

Corona also contains an optical token ring, as discussed in Section 3.3.1.6, for arbitrating the MWSR crossbar. Figure 3.54(a) shows the insertion loss for this subnetwork. Because we are not modulating data but just passing tokens, we can consider lower-Q resonators of 5k, 10k, and our normal 15k. Only Q=5k resonators are supported because of the much higher power budget that using filters and not



(a) Insertion Loss

(b) Injected Power

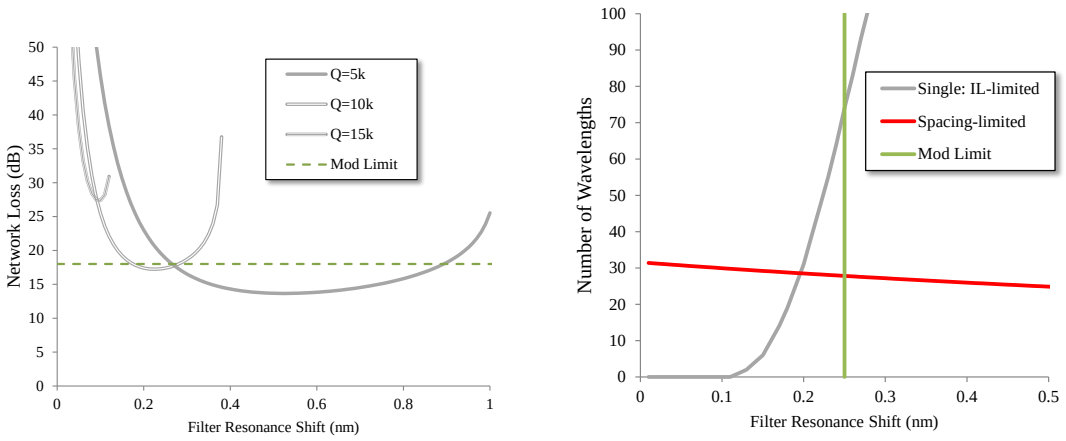
Figure 3.54: Insertion loss analysis for Corona's token ring network. (a) Worst-case insertion loss versus modulator resonance shift in linewidths. (b) Injected optical power required versus number of wavelength channels, limited by the nonlinear threshold.

modulators provides. Unfortunately, an 8×8 network will still not work because of waveguide-limited power.

Figure 3.54(b) shows how many wavelengths can be used if we consider networks smaller than 8×8. The "required" line is the number of wavelengths required for the token ring to work for various number of access points (clearly, one per access point). The "supported" line is the number of wavelengths which fit inside the waveguide power limit. Also shown is the network loss and the filter power limit for reference. The token ring only ends up being able to support around 30 access points. Therefore, we're not going to consider Corona for simulation without design changes.

3.3.2.2 Firefly

The Firefly network (67) was proposed to reduce the complexity of a wavelength-routed architecture to save power and area by using electrical links for local communication and a serpentine photonic layout for global communication. One important difference of Firefly from Corona is that it uses SWMR buses instead of MWSR, arbitrated by separate reservation channels. This means that it requires tunable filters which are OFF by default to save power, and can shift onto resonance to detect a signal. Because these filters need to be as small as possible (to maximize their FSR) with a silicon slab (for electrical



(a) (b)

Figure 3.55: Insertion loss analysis for Firefly's data network. (a) Worst-case insertion loss versus filter resonance shift for microring resonators with different quality factors. Lines stop when the quality factor cannot be maintained. (b) Number of wavelength channels possible in optical power budget versus resonance shift, ultimately limited by modulator power and FSR.

actuation), these filters will have a relatively high amount of loss (about 20 dB/cm) due to radiation. We can design the filter to trade-off through port loss for drop port loss by varying the amount of resonant shift from OFF to ON state. If the resonance peak in the OFF state is farther away from the wavelength of light, making the through loss lower, it will take more injected carriers to shift it back to the ON state, which incurs more loss when dropping the wavelength. For Firefly, we investigate the design of this critical component instead of using regular passive filters. The details of this analysis are discussed in Appendix 4.2.2.2.

Figure 3.55(a) shows the worst case insertion loss for one assembly channel in Firefly versus filter resonance shift for filters with different quality factor values. First, we notice an optimal point for each curve as through port loss is traded for drop port loss. Unfortunately, even though resonators with large quality factors could be supported in terms of insertion loss, the optical power in a single modulator limits the maximum insertion loss to 18 dB. This means that the only microring resonators that are feasible here are ones with quality factors of 5,000.

Though searching for the optimal lowest insertion loss point will save on laser power by requiring less injected optical power, using higher resonant shift distances will require larger wavelength channel spacing. We define the required channel spacing as:

$$\delta_{spacing} = 6 * \delta_{linewidth} + \delta_{shift} \tag{3.17}$$

This maintains six linewidths of channel spacing at all times (regardless of the shifting), which guarantees less than –20 dB of filter crosstalk. Figure 3.55(b) shows the number of wavelength channels, limited by both insertion loss and wavelength channel spacing. However, both of these values are ultimately limited to the modulator optical power limit to 27 wavelengths. However, we will not consider Firefly for simulation because its arbitration network, though feasible in terms of insertion loss according to their design, relies on broadcast functionality based on either slightly OFF-resonance rings or fractional absorption of a ring-detector, neither of which have been shown in practice to work reliably, and therefore can't be modeled in PhoenixSim's physical-layer modeling.

3.3.3 Impact of Deposited Multi-Layer Devices

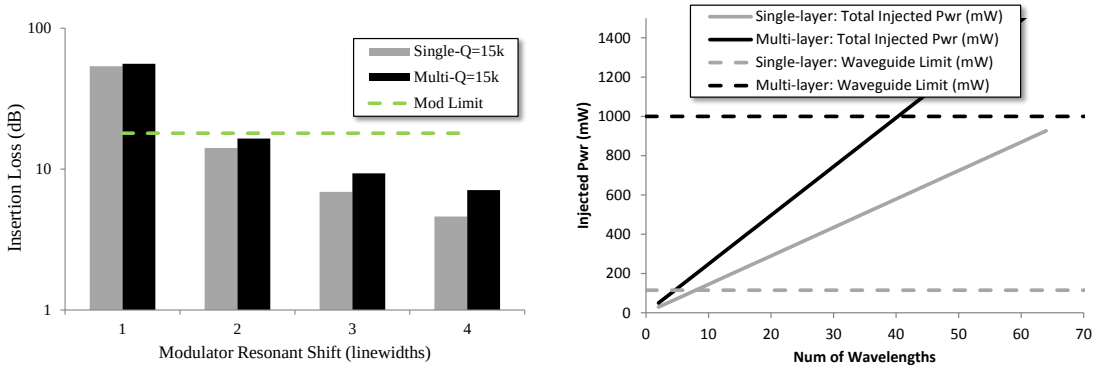
Like circuit-switched networks, wavelength-routed networks can benefit greatly from the use of multi-layer deposited materials, mostly due to the lower propagation loss and higher nonlinear threshold of silicon nitride.

3.3.3.1 Multi-layer Corona

If we take a look at a multi-layer version of Corona's crossbar, we actually see slightly higher insertion loss, shown in Figure 3.56(a). The details of this analysis can be found in Appendix 4.2.2.1. This is due mostly to the vertical nitride-poly couplers and propagating in poly when in modulator or filter regions. However, referring to Figure 3.56(b), because silicon nitride has a much higher nonlinear threshold and Corona is limited by it's power-delivery waveguide, the multi-layer version can support up to around 40 wavelengths.

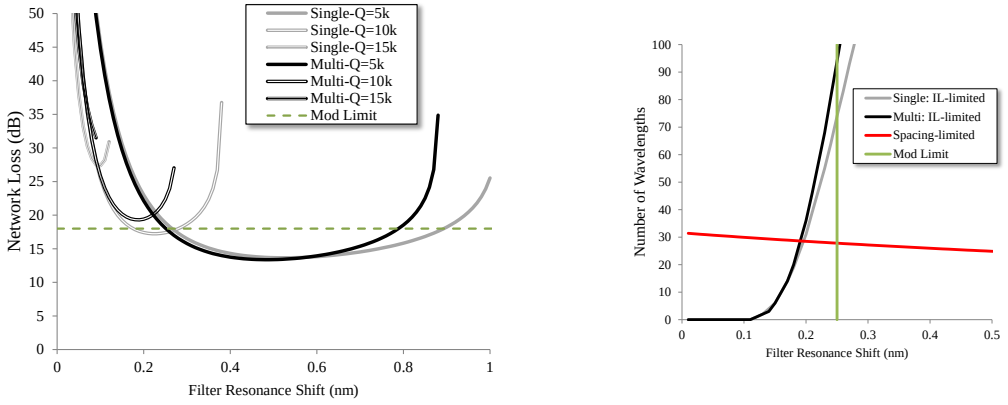
3.3.3.2 Multi-layer Firefly

If we take a look at a multi-layer version of one of Firefly's assemblies, we see little improvement for multi-layer, shown in Figure 3.57(a). The details of this analysis can be found in Appendix 4.2.2.2. Compared to Corona's crossbar, Firefly has relatively longer propagation lengths which save insertion



(a) (b)

Figure 3.56: Insertion loss analysis for Corona's crossbar using multi-layer devices.



(a) (b)

Figure 3.57: Insertion loss analysis for one of Firefly's assembly using multi-layer devices.

loss using silicon nitride. However, referring to Figure 3.57(b), Firefly is still limited by modulator optical power and modulator FSR to around 30 wavelengths. Multi-layer devices would only save on injected laser power for Firefly.

3.4 Summary

As we have seen, there are many ways to design and implement a photonic NoC, and given an application space or target environment, it is not always clear how best to go about it. Circuit-switched networks are good at providing high bandwidth energy efficient end-to-end links over large distances (at the chip

scale) or even off-chip to memory, but suffer in the face of frequent small messages because of path-setup contention. Some important design improvements can help circuit-switched network designs, such as selective transmission, gateway concentration, and TDM arbitration.

Wavelength-routed or arbitrated networks can be useful for providing very low latency communications by using the wavelength domain to implicitly encode routing information. However, as we saw, they can be very difficult to implement when physical-layer constraints are taken into account, requiring a large number of ring resonators along the same waveguide path which can lead to high insertion loss and crosstalk due to imperfect filtering. Sometimes switching filters are needed to implement some of the functions necessary, which drives up insertion loss due to carrier injection.

Finally, multi-layer devices using deposited materials are beneficial to photonic NoCs regardless of the design, from the low propagation loss in silicon nitride and elimination of waveguide crossings from the ability to deposit the material. Though it is difficult to predict if and when full-blown photonic NoCs will make their appearance in CMPs, and the extent to which they are used in the CMP domain as a whole, many of the issues discussed in this chapter will continue to be extremely important for making photonics worth the effort.

4

Computer-Aided-Design of Photonic Interconnection Networks

As nanophotonic technology becomes more mature, with real devices in commercial products available today (15), providing the correct tools to both increase the understanding of the interaction between the devices and enable scaling of the level of integration will become necessary. Traditional CMOS electronics have extensive toolsets from high level functional descriptions down to fabrication-specific layout. This chapter discusses some tools and methodologies that bring us closer to realizing the architectural concepts described in Section 3

4.1 Simulation and Analytical Optimization

In Chapter 3 we explored the design space of photonic networks a little to get an understanding of the different effects that design choices can have. In this section, we will briefly go over PhoenixSim (68), a simulation framework for investigating photonic networks, and some optimization problems that are interesting in phtonic network design.

4.1.1 PhoenixSim

PhoenixSim (Photonic and Electronic Network Integration and eXecution Simulator) was originally designed to allow us to investigate silicon nanophotonic NoCs taking into account components physical

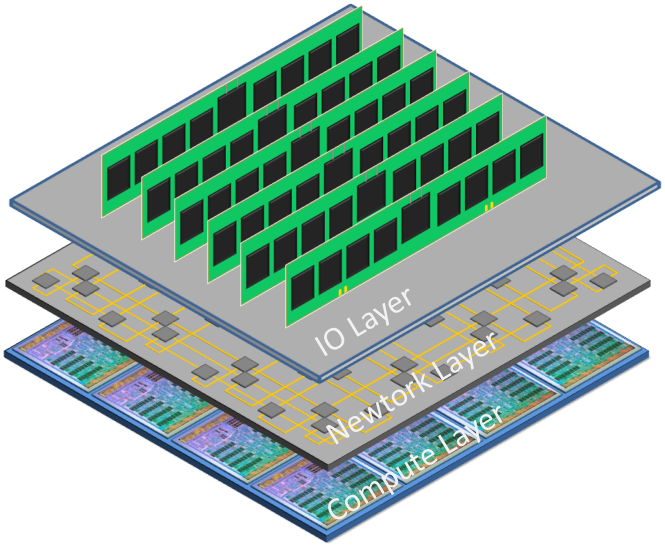


Figure 4.1: Basic structure of a PhoenixSim simulation.

layer characteristics. Because photonics often requires electronic components around it for control and processing, we also incorporated models of some typical electronic network components. We ended up with a simulation environment that is suited to investigate both electronic and photonic NoCs.

PhoenixSim is built on OMNeT++ (69), an environment for creating any event-driven simulator. Making PhoenixSim event driven means its relatively efficient for what it does. However, many of the events that occur in the simulator are on a clock-cycle granularity, which means it does not sacrifice too much accuracy. OMNeT supplies functional libraries, mechanisms for instantiating components, managing parameters, and executing batches of simulations.

4.1.1.1 The Big Picture

Most of the computing systems that can be modeled in PhoenixSim look like Figure 4.1. The Processing Plane consists of many cores executing applications, which produce comminucation events to the Network Plane. The IO Plane can consist of anything off-chip, including DRAM modules or inter-chip communication.

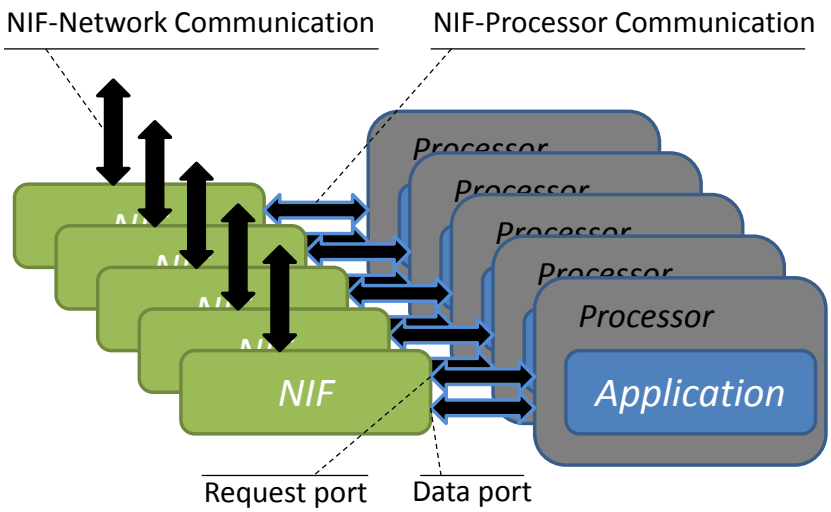


Figure 4.2: Structure of the Processing Plane

4.1.1.2 Processing Plane

The Processing Plane consists of multilpe Processors, which represent cores or other processing elements, each having an instance of an Application class which dictates how communication events are generated. The structure of the Processing Plane varies slightly depending on if and how *concentration* is implemented, but generally looks like Figure 4.2. An instance of a Processor exists to model each independent processing core in the CMP. A Network InterFace (NIF) translates a Processor's communication request event into the network-specific protocol needed to complete it.

Network Interfaces (NIFs) The NIFs in PhoenixSim provide a way for us to decouple the design of the network with subsystems connected to the network, such as the Processing Plane. A NIF initiates new communication on the network by sequentially calling 4 functions which are pure virtual in the NIF class, and therefore required by all subclasses:

- request() - called when a communication request arrives from the non-network side of the NIF. Returns true if communication should continue.

- prepare() - called after request() returns true. Usually creates a new message to be sent on the network. Returns true if communication should continue, which is almost always the case.

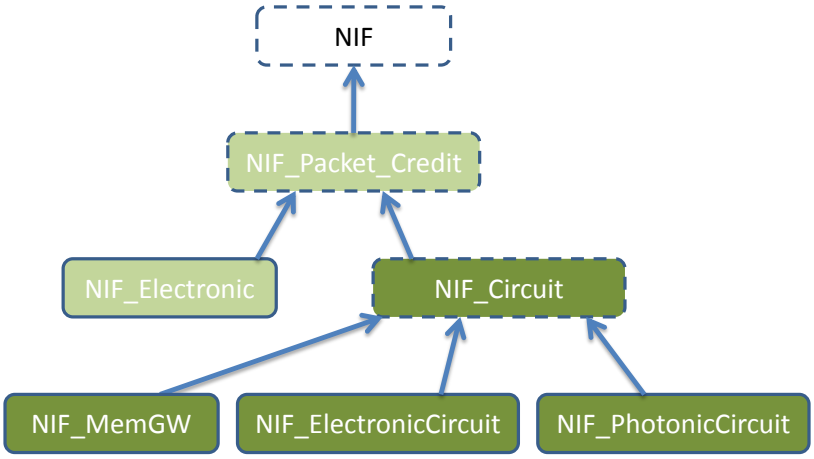


Figure 4.3: Hierarchy of current NIFs. Abstract classes shown outlined with dashes.

- send() - called after prepare() returns true. Responsible for sending the prepared message. Again, usually always returns true.
- complete() - called after send(). Performs anything that is required after the transmission has completed. Returns the amount of time that the NIF should wait before attempting to start a new transmission.

Figure 4.3 shows the hierarchy of existing NIFs. Currently, we have two other abstract classes that are available to inherit from: NIF_Packet_Credit and NIF_Circuit. The NIF_Packet_Credit class contains some useful functions for interacting with credit-based packet-switched networks, such as standard electronic ones. The NIF_Circuit class assumes a packet-switched network which controls a circuit-switched data plane, and therefore further builds on NIF_Packet_Credit.

Applications The application model in PhoenixSim works by informing the Processor that owns it what the next communication event will be, and how long the Processor should wait before sending it (*i.e.* how long it takes to process the code that generates the communication event). An Application can respond to any of 4 events that the Processor informs it of: first message, data arrived, sending data, and message sent. A separate instance of an application runs on each Processor, which enforces any communication to take place on the network. This is a good way to make sure that an application model does not have any information that would not actually be available to it on a real system.

Parameters to an Application are passed through general OMNeT parameters called appParam1, appParam2, appParam3, and appParam4, which are all of type *double*. There is also an appParam5 of type *string* which may be used. These parameters are made available to the Appplication superclass, and therefore to all its subclasses. It is up to the specific instance of the Application how to use these parameters. We will spare the reader discussing all of our application models, but as you can imagine, the synthetic applications in Chapter 3 are among them.

4.1.1.3 Network Plane

The main purpose of PhoenixSim is in the Network Plane, which consists of routers and switches to transfer data from one point to another. First, lets go over the devices and components that we use to build routers, both electronic and photonic. Lets first look at electronic components.

Electronic Router Our electronic router model is your basic store-and-forward packet-switched router. A high level diagram can be seen in Figure 4.4. Functionally, we model it as having a 3-stage pipeline: message arrival and request, arbitration, and switch traversal. Currently, we use credit-based Bubble flow control (70), though we will be looking to implement more sophisticated mechanisms in the future.

For power modeling, ORION (71) is used for nearly every electronic component in PhoenixSim. In most cases, we use ORION's record(...) and report(...) functions to record the energy for each individual event as they happen, as opposed to their stat_energy(...) functions, which try to calculate energy dissipation based on an activity factor.

We will now go into detail on how each part is modeled.

RouterInport The microarchitecture for the RouterInport can be seen in Figure 4.5. A single inport has a separate SRAM array for each virtual channel. Minor control logic interacts with the RouterArbiter, handling requests and grants. The ORION_Array class is used to model the power for each virtual channel buffer.

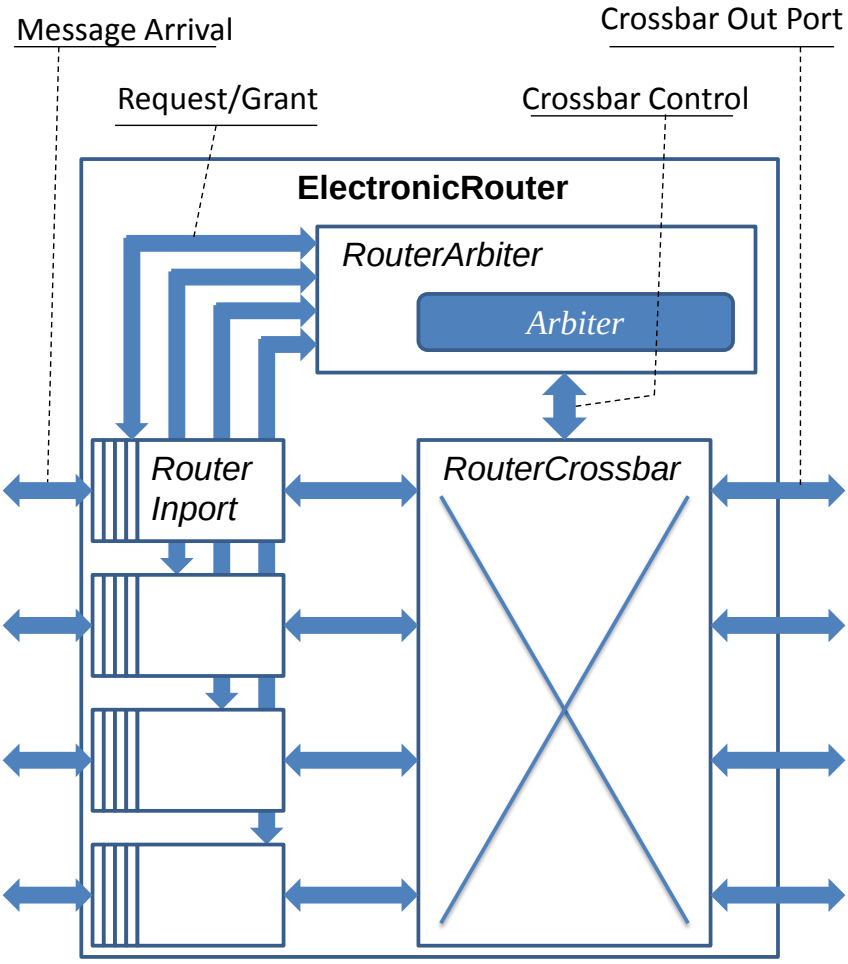


Figure 4.4: Structure of Electronic Router.

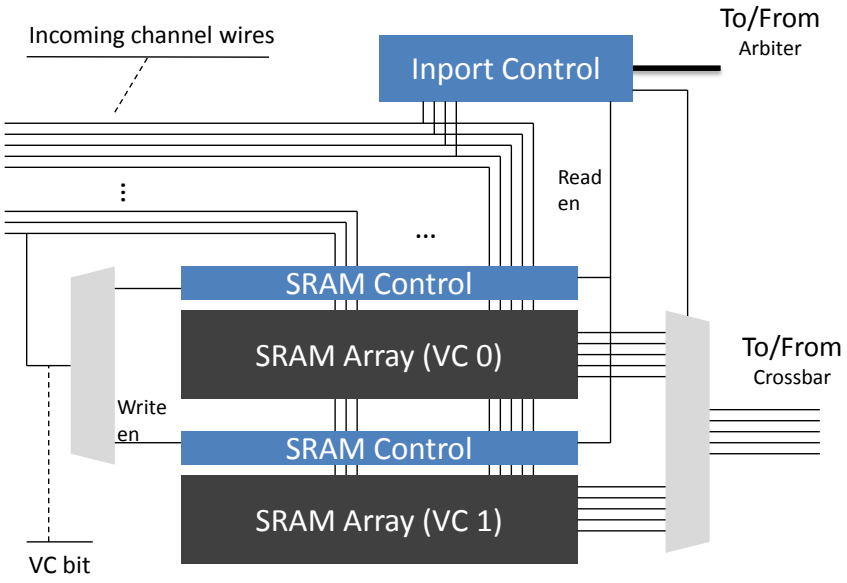


Figure 4.5: RouterInport microarchitecture.

4.1.1.4 RouterCrossbar

The RouterCrossbar is modeled as a dumb crossbar: it accepts commands from the RouterArbiter to set up input-output connections, and accepts messages arriving at an input, forwarding them to the correct output. The ORION_Crossbar is used to model the power when messages traverse it, modeling the actual switching of input-output connections and 50% probability of bit-changes on each line.

4.1.1.5 RouterArbiter

The RouterArbiter implements three main functions: routing, contention, and device setup. Because these things can be different depending on the network, we allow the user to specify his own subclass of Arbiter. The Arbiter class contains the basic algorithm for ensuring that a message is correctly routed.

Figure 4.6 shows the class hierarchy for arbiters. The following are the main abstract arbiter classes which capture different functionality:

- ArbiterCanRoute - parses and compares the NetworkAddress of the router and destination of the message. See Section 4.2 for more details on addressing in PhoenixSim.

- Arbiter - main superclass, contains main round-robin loop for examining waiting messages.

- ElectronicArbiter - implements Bubble credit-based flow control
- PhotonicArbiter - implements logic for handling various path-setup messages. See Figure 3.4 for more details on path setup.
- PhotonicNoUturnArbiter - checks for U-turns, which is not allowed in some photonic switch designs.
- Photonic4x4Arbiter - implements device control logic for the different designs of 4x4 photonic switch. See Section 3.2.2.1 for more details.

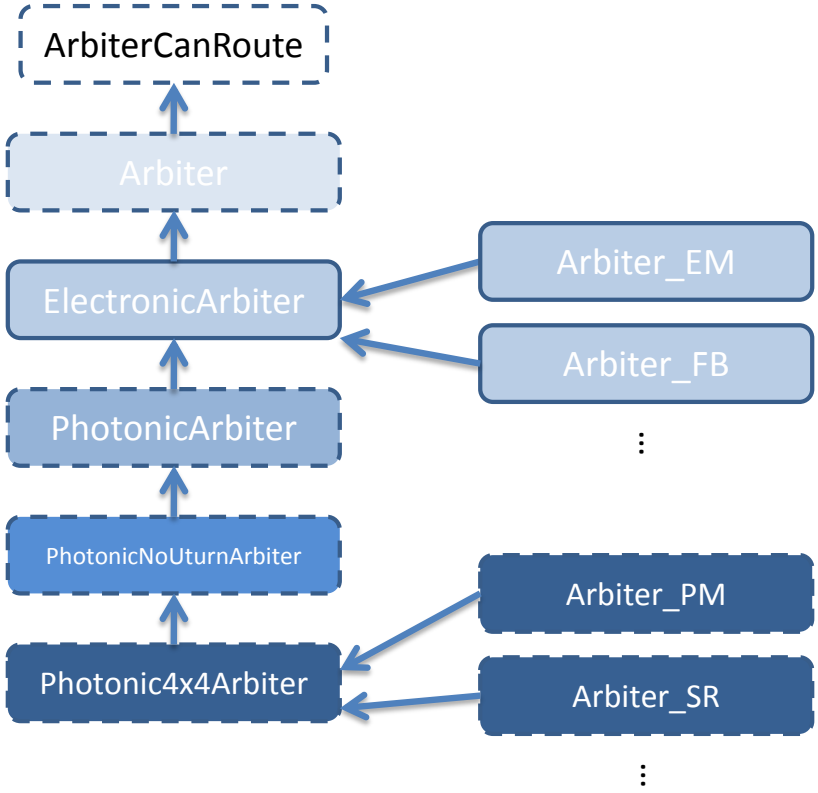


Figure 4.6: Arbiter class hierarchy.

For now, all of the arbiters for the current networks we have implemented inherit from one of these abstract classes.

ElectronicChannel The ElectronicChannel module models optimally-repeatered intermediate/global electronic wires in parallel connecting routers together. The channel uses a basic formula for computing the time it takes to transmit data:

$$t_{transmission} = clockPeriod * (numRepeaters + messageSize/(channelWidth)) \quad (4.1)$$

ElectronicChannel uses ORION_Link to model it's power, and obtain the optimal number of repeaters based on the length. The length of the channel is specified as number of inter-router spaces and number of router widths. Router width is calculated in the RouterStat module (see below), and inter-router spaces are calculated as the width of a core minus the router width, assuming one router per core.

RouterStat The RouterStat module is included in the ElectronicRouter to calculate both the router area and the router's clock power, both as estimated by ORION. The inports, arbiter, and crossbar all send an OMNeT message to the RouterStat on simulation startup, indicating their parameters which RouterStat uses to calculate area and clock power. ORION provides an estimate of total area, and RouterStat assumes that the router is square.

Virtual Channels We've mentioned our support for virtual channels. Currently, they are implemented using separate physical buffers. To avoid packets from a large application-level messages from arriving at a destination out of order, we do *not* allow a packet to change virtual channels in a network. Once a virtual channel id is assigned to a packet, it stays that way until it gets to where it's going. The one exception is when virtual channels are used a priority channels in circut-switching control (not going to get into that). An electronic network expert might cite this as a severe implementation flaw, and he might be right, as electronic network performance could be enhanced using fancy flow control techniques, etc. We'll be adding support for that kind of thing in the future.

4.1.1.6 Photonic Devices

Our library of photonic devices comprises of all photonic technologies required to generate, control, and receive an optical signal. This library of devices can be used to create any number of switch fabrics

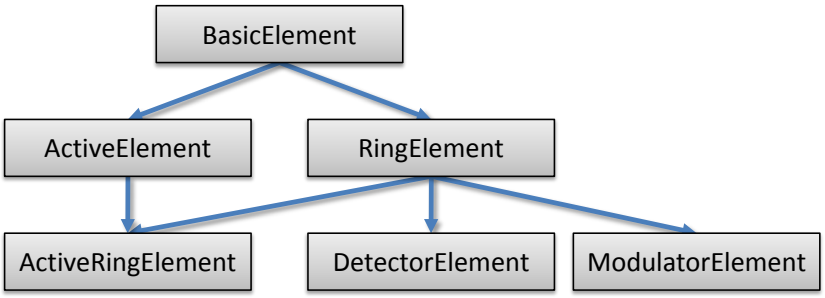


Figure 4.7: Hierarchy of the photonic modeling classes.

and network topologies. Our efforts in building a framework for describing these devices has required us to strike a balance between a physical accuracy and system level performance simulation. On one hand, we want to be able to model the devices in a physically accurate way without requiring a full FDTD simulation, on the other hand, we also want to be able to simulate the devices operating in a network environment to produce meaningful system-level performance results. This has resulted in the development of a Basic Element abstraction for describing all photonic devices.

The current class hierarchy is shown in 4.7. The BasicElement base class abstracts all the photonic characteristics common to all photonic devices. Practically, this describes characteristics of insertion loss and latency. Beyond the BasicElement class, we provide subclasses which are used to describe more specific types of devices. Currently this is used to describe passive devices such as straight waveguides, bending waveguides, waveguide crossings, and couplers.

The RingElement class inherits from the BasicElement class, and provides a means to describe the resonant behavior of rings (and in fact any resonator devices, such as Mach-Zehnders). In essence, RingElements are just BasicElement devices that exhibit wavelength dependency. This means that an optical signal coming it at a certain wavelength may behavior different from a second signal that has a different wavelength. RingElement is currently used as the inhereted class for ring-based filters.

The ActiveElement class also inherits from the BasicElement class, but additionally provides mechanisms for describes devices that can exhibit multiple states. This class should not be confused with active physical devices, which would generally describe a component that requires input power to function. ActiveElements describe devices that have multiple logic states, such as a switch that can be 'on' or 'off'. For example, the 'on' state might indicate that a switch is in a cross state, while the 'off' state

tbp

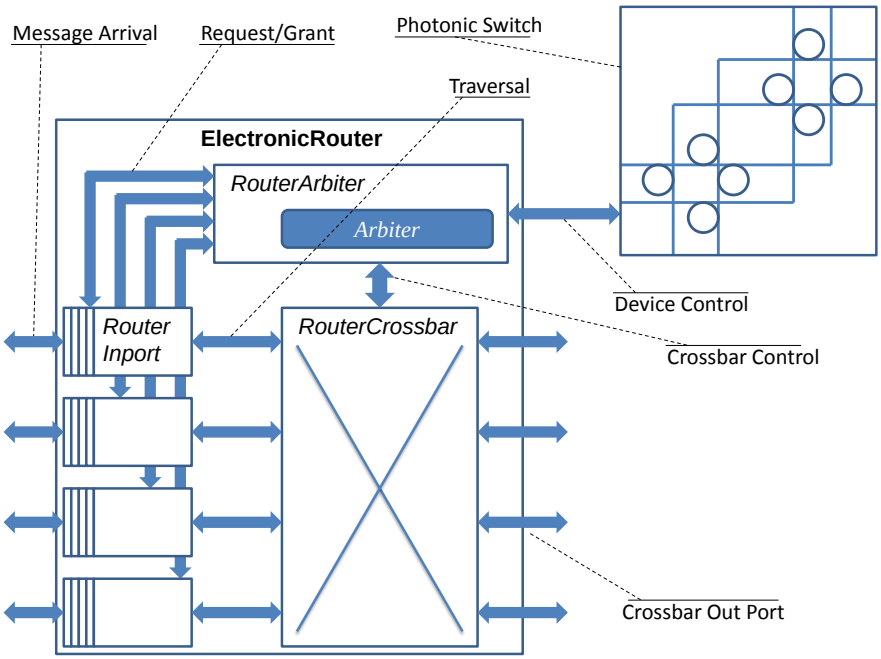


Figure 4.8: Hybrid router consisting of a photonic switch, controlled by an electronic packet-switched router.

will indicate that the switch is in a bar state.

While our currently library does not contain any strictly ActiveElement devices, we have a number of devices that belong in the ActiveRingElement group, which inherits from both the RingElement and ActiveElement class. This includes the 1×2 and 2×2 photonic switching elements (PSEs).

Lastly, we have also included the DetectorElement and ModulatorElement classes which are special classes for injecting and detecting optical messages. Specifically, our library currently only contains detector and modulator devices that are based on ring resonators, therefore the classes can in actuality be more accurately described as RingDetectorElement and RingModulatorElement.

4.1.1.7 Hybrid Router

Our HybridRouter module consists of an ElectronicRouter which controls a switch, seen in Figure 4.8. Using multiple hybrid routers, if we connect the electronic routers together and the switches together, we effectively form a circuit-switched network where the electronic routers are the control plane, and the switches are the data plane.

The HybridRouter module has two important parameters: elRouter and optSwitch, which specify the arbiter type for the electronic router, and the name of the switch which implements the module interface PhotonicSwitch, respectively. These two parameters allow you to use the HybridRouter module in different circuit-switched networks by simply writing a new Arbiter and implementing a new PhotonicSwitch.

4.1.1.8 IO Plane

The structure of the IO plane very much depends on the simulation being run. Currently, we put our DRAM models into this separate plane, but you could put anything there that represents off-chip components (other chips, outside networks, etc). We will briefly describe our DRAM device and control models, and how they interact with the network.

DRAMsim There are two models for the DRAM subsystem included with PhoenixSim. The first is DRAMsim (72), a well-known DRAM simulator from the University of Maryland. This model is usually used to simulate contemporary DRAM subsystems. It contains the usual DRAM devices and control you would expect to see in today's computers (well, maybe yesterday's). We're not going to describe the whole thing, so you can check out the website for more details http://www.ece.umd.edu/dramsim/. Typically, we use DRAMsim when simulating packet-switched electronic NoC's. Because packet-switched networks use relatively small packets that must fit into network buffers, they are closely aligned with today's memory access mechanisms (*i.e.* accessing single cache lines at a time).

DRAMsim contains all the models necessary for a complete DRAM subsystem, including the memory controller, transaction queue, bank, chip, rank, address translation, and many different configurations for control policies, as well as performance and energy consumption.

We've slightly modified DRAMsim to fit into PhoenixSim. First, we refactored it into C++. Why it was originally written in C is a mystery to us, as it lacks serious organization and characteristics of good software. We've also wrapped it into an OMNeT module, called cDramModule. Figure 4.9 shows the structure of the modules we typically instantiate when using DRAMsim.

While the ElectronicWire model is used for on-chip wires, ElectronicIOpad is used for off-chip ones. The cDramModule wrapper module does a couple of things:

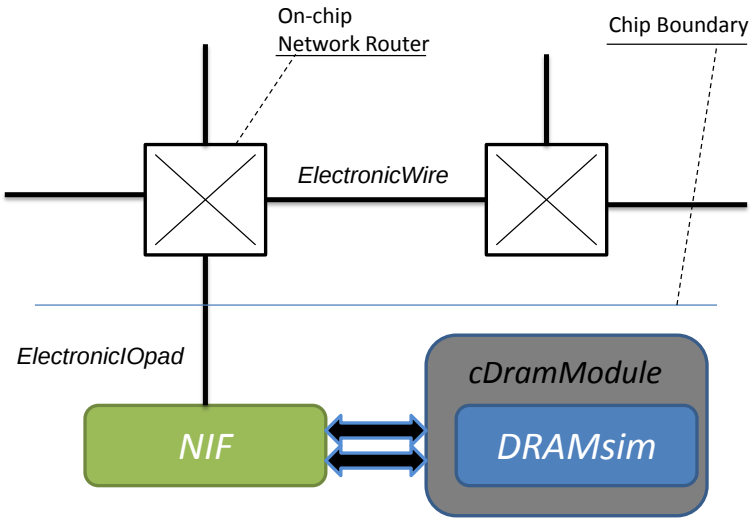


Figure 4.9: DRAMsim used in a packet-switched network.

- Interact with the NIF attached to it. In this way, the cDramModule implements the same interface that the Processor uses.

- Queue up transactions. DRAMsim comes with a transaction queue model which it handles. The cDramModule wrapper attempts to insert incoming transactions into DRAMsim. If the transaction queue is full, the wrapper is responsible for providing back-pressure to the NIF.

- Managing large messages. In PhoenixSim, we allow large messages to be broken up into small packets so they can fit into the router buffers. However, if a read transaction arrives at the cDramModule which requests a large amount of data, the cDramModule actually issues many small read transactions to DRAMsim, because that's what DRAMsim expects: cache-line size accesses.

Again, we use DRAMsim for modeling the memory subsystem for electronic networks, because DRAMsim was made to model today's shared-memory small accesses coming from a small number of shared caches, which can approximate how a packet-switched NoC might behave.

4.1.2 DRAM-LRL

The second model is one that we have developed, which we call DRAM-LRL, which is used to model a hypothetical memory subsystem design in the context of circuit-switched networks. Full details can be

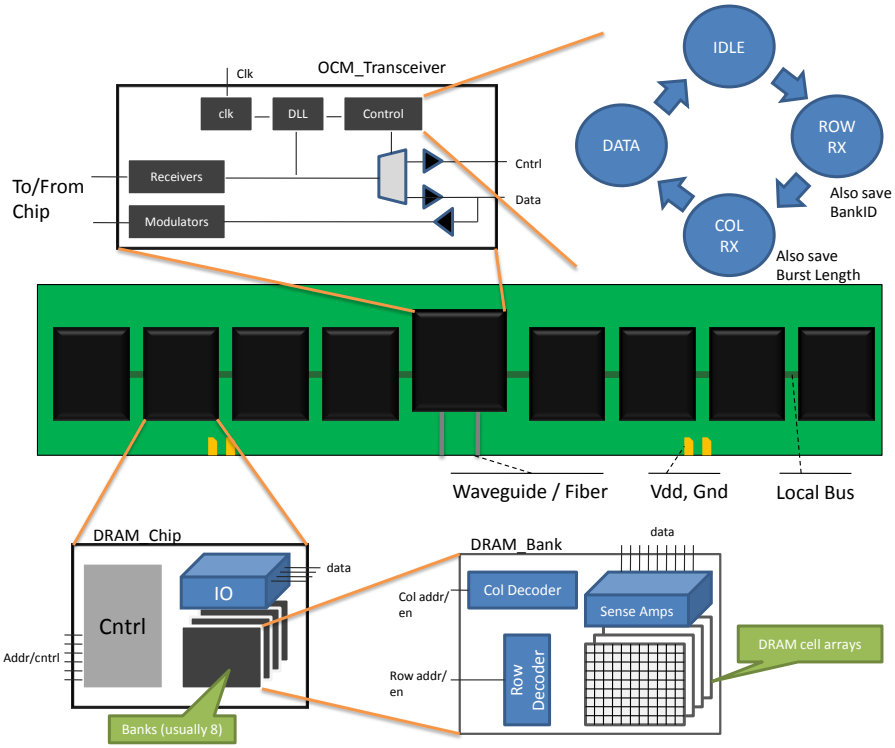


Figure 4.10: CAMM structure in DRAM-LRL.

found in (52). DRAM-LRL was made to fit into PhoenixSim more efficiently by staying event-driven, as opposed to DRAMsim which models every cycle. DRAM-LRL is further simplified by the fact that it is made to be used in circuit-switched networks. Figure 4.10 shows the components we model in DRAM-LRL. It shows a Circuit-Accessed Memory Module (CAMM), which has a central OCM_Transceiver which performs O-E-O conversion, and controls the address and data local bus usage according to the stage of the transaction. Only one transaction need be sustained at a time, due to circuit-paths having exclusive access to the module. Some of these things were discussed earlier in Section 3.2.4.

Core to Memory Mapping Each Processor in PhoenixSim can be viewed as having it's own local memory space. We use a class called DRAM_Cfg to dictate how a Processor locates its memory space. An Application should therefore refer to the DRAM_Cfg instance in the Application superclass to find out where it should go for memory accesses, if the programming model has such a thing as "local" memory spaces. DRAM_Cfg has two main functions:

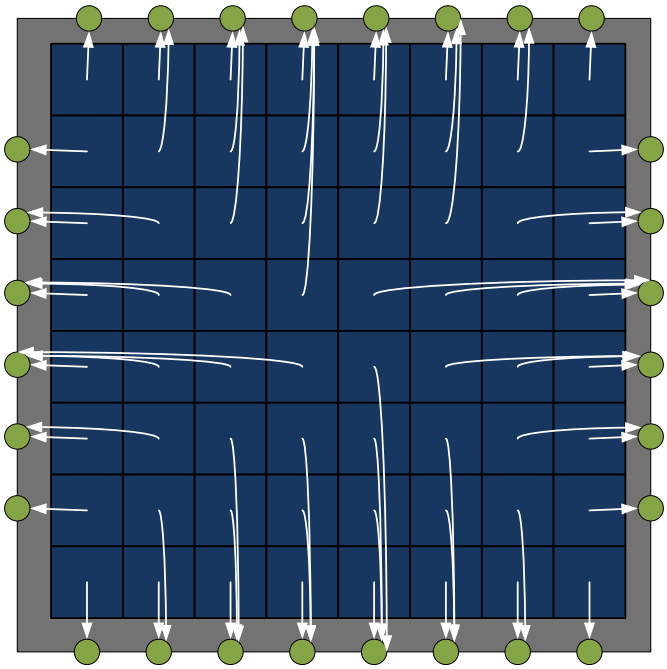


Figure 4.11: Default core-memory map.

- getAccessId(int coreId) - returns the id of the network node where this Processor's memory space is attached to.
- getAccessCore(int dramId) - returns the id of the network node that the dramId memory bank is attached to.

Using these functions, a Processor can ask the DRAM_Cfg where it's local memory address space is, and how to get there from here. Figure 4.11 shows how cores are usually mapped to memory gateways, assuming one memory gateway per peripheral network node.

For some networks, it may be useful to define other mappings. In that case, just override DRAM_Cfg and instantiate it in Processor, so that the Application can use it.

4.2 Addressing

So, how are cores in the network addressed? This was kind of a problem if we wanted to support many different kinds of networks. Sure, you could give them just some random unique integer, but it makes it easier on the routing logic if the id's make sense.

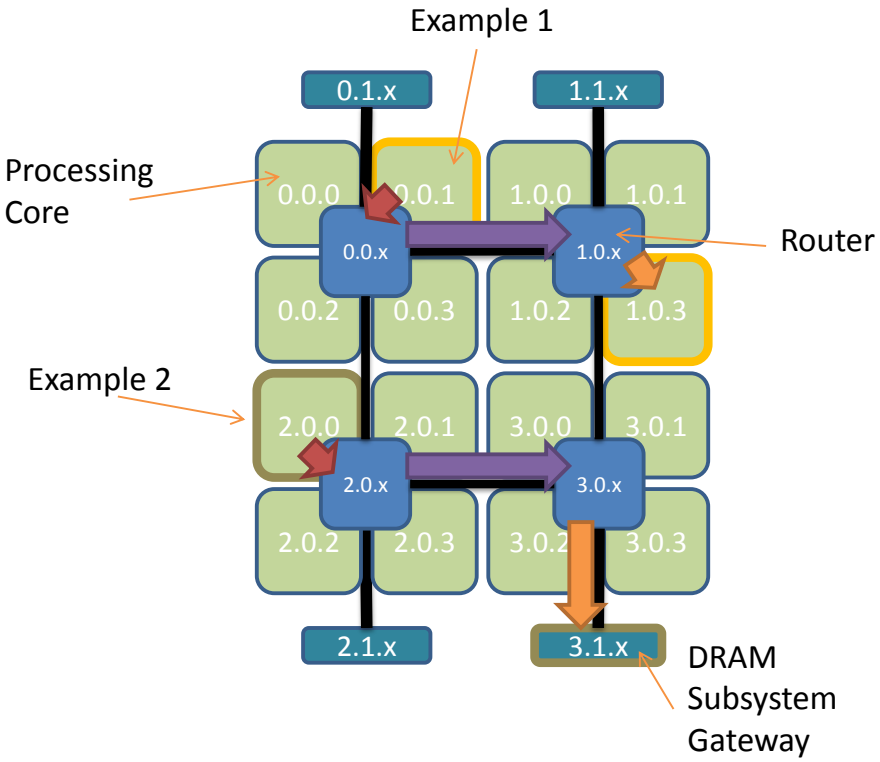


Figure 4.12: Example of how addressing and routing works.

So we came up with a hierarchical addressing scheme. Furthermore, we've made it so you can define the structure of your addresses however you want. We use the concept of address *domains*, much like an IP address. In PhoenixSim, the left-most address is considered the "top", and the right-most the "bottom". Lets do a simple example.

Consider the 2×2 network in Figure 4.12, where each router is concentrated with 4 cores attached to it. The format of our addresses is:

$$NET.MEM.PROC \tag{4.2}$$

where NET is the network domain, or the routers. MEM is the memory gateway domain. PROC is the processor domain. Note how the addresses are assigned in Figure 4.12. In addition, each Arbiter has a *level*, which specifies which domain it is in. Since the routers are the "top" level, they only have the NET domain defined.

The process for routing up and down the address domains is defined in ArbiterCanRoute. Recall

from Section 4.1.1.5 that every Arbiter must inherit from ArbiterCanRoute, because that class contains the mechanism for parsing the addresses. When defining an Arbiter, you can implement three functions:

- route(...) - decides where to forward the message when in the same address domain. This is required.
- getUpPort(...) - decides which port to go to when we need to go up a domain. Defaults to route(...).
- getUpDown(...) - decides which port to go to when we need to go down a domain. Defaults to route(...).

When a message reaches an arbiter, it looks at the destination address, starting with the top level. Depending on which address does not match the arbiter's address, and which level the arbiter is will dictate which function gets called.

Lets look at the communication labeled Example 1 in Figure 4.12. Core 0.0.1 wants to send to Core 1.0.3. Here's how it works:

1. Core 0.0.1 must decide where to forward his message. It starts with the top level, NET. The destination is 1.0.3. The NET domain is 1 for the destination, and 0 for the sender. They are different. Also, the level of the Core is PROC. Since the first difference was encountered at a level higher than the Core's level, it calls getUpPort(...). This is indicated by the red arrow in Figure 4.12.
2. Now Router 0.0.x must decide where to forward the message. It compares its address (0.0.x) to the destination (1.0.3). Again, the difference occurs in the NET domain. Since the Router is in the NET domain, it calls route(...), which sends it East. This is indicated by the purple arrow.
3. Router 1.0.x must now decide where to forward the message. It compares its address (1.0.x) to the destination (1.0.3). The first difference occurs in the PROC domain, which is lower than the arbiter's level (NET). Therefore, the getDownPort(...) function is called. This is indicated by the orange arrow.

Example 1 was for Core-Core communication, basically bypassing the MEM domain. Example 2 illustrates how this domain can be used. Core 2.0.0 wants to send a message to DRAM gateway 3.1.x.

Note that the gateway itself does not have an address. Core 2.0.0 sends a message there by indicating the NET-domain address of the router connected to it. In this case, Router 3.0.x. It then specifies a 1 for the MEM domain, indicating that when the message gets to Router 3.0.x, it should call getDownPort(...) in response to the difference between it's MEM address (0) and the destination's MEM address (1). The arbiter must check for this case in the getDownPort(...) function.

Defining Network Addresses In PhoenixSim, you are free to define your own address format. The above examples used the NET.MEM.PROC format, which is common for a flat network structure. Indeed, any network could use this format. But you can make your life easier when writing your arbiters by using different formats.

By default, when you instantiate a processingPlane module, it tacks on the MEM.PROC domains. Typically, you define the network domains using the networkProfile parameter. This is done using a string which looks like

$$**.networkProfile = "name1.name2.;[N1].[N2]." \tag{4.3}$$

where name1 and name2 are domain names, and [N1] and [N2] are the number of nodes contained within these domains. The number of domains you can have is arbitrary. For instance, the format used above in our examples would look something like this:

$$**.netorkProfile = "NET.;" + string(X * Y) + "." \tag{4.4}$$

where you've already defined X and Y as the number of network nodes in the x and y coordinates.

Address Translation There's only one more detail you need to know. The Application classes use integers to denote other processing cores. This is done on purpose, because an application model should not have to worry about the network it's running on. The distribution of an application should be a logical construct, not a physical one to decouple how we write and distribute an application from the hardware it's running on.

So, we need a mechanism to translate from logical unique integers to addresses in the network. This

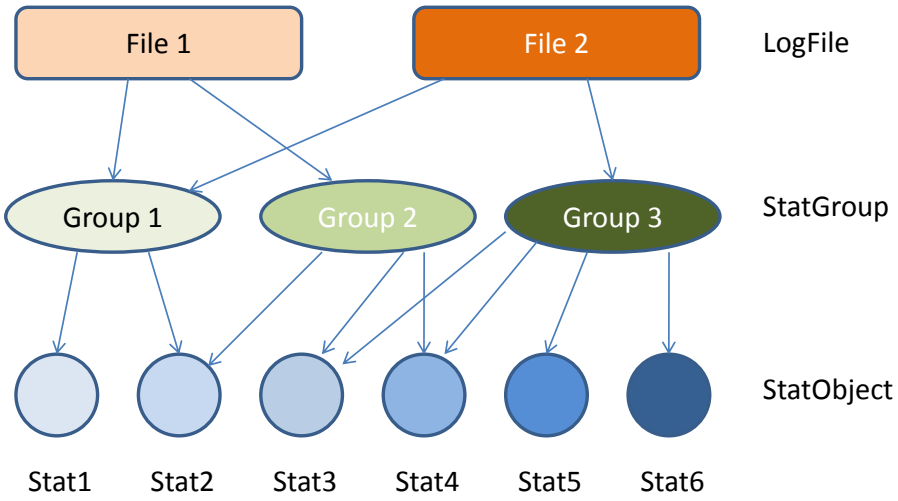


Figure 4.13: How statistics work.

is done using the AddressTranslator class, found in processingPlane/addressTranslation/. If you define your own addressing format, you need to write your own AddressTranslator. Don't worry, it's easy. All you have to do is convert an incoming ApplicationData to a NetworkAddress.

The AddressTranslator_Standard class is used for the regular NET.MEM.PROC format. Look at that to see an example.

4.2.0.1 Statistics and Results

The point of the simulator is, of course, to measure statistics to tell us what's going on, from energy to performance. We've set up a few mechanisms to make things easier. Every network should contain one and only one Statistics simple module. This module creates the results files, and is where all statistics are registered. This code is found in statistics/statistics.cc, and is where any result file changes must be made (other than just adding new statistics to the current single result file).

Figure 4.13 shows the logical organization of statistics in PhoenixSim.

There are three classes you could be aware of:

- LogFile - this describes a results file, which is a collection of StatGroups.
- StatGroup - this defines how to organize different StatObjects. For instance, whether just to list them all or add them all up and report the sum.

- StatObject - this defines how something is actually measured. For instance, taking the time average or just total of all measurements.

Basically, a LogFile can contain any number of StatGroups, which define what goes in the file. A StatGroup is a collection of different statistics. They basically end up getting their own section in the result file. When a StatObject is created in the simulation, it must first register itself with the central Statistics module. When that happens, any StatGroup that exists can pick it up. A StatGroup must know a few things (passed to the constructor) to gather only the right StatObject:

- Name - the section header to write in the file
- Range - the range of StatObject types that should be included in this group. For instance, ENERGY_STATIC - ENERGY_LAST specifies all energy-related statistics. See StatObject.h for types.
- Filter - a string as an additional filter. Only statistics coming in specifiying they are part of this group will be added.

Take a look at Statistics::Statistics() to see the StatGroups that are instantiated. Finally, when you want to measure something, you create a StatObject, and call its track(...) function to record the value.

Registering statistics When you want to measure something, first you need the right type of StatObject. The following are the ones currently defined:

- Count - counts the number of times this object is created. This is useful for counting the number of a particular type of components get created in your network.
- MMA - stands for Min Max Average. Calculates these stats for all values that are tracked.
- TimeAvg - reports the average over time for all values tracked.
- Total - sums up all the tracked values.
- EnergyEvent - sums up all discrete events that consume energy.
- EnergyOn - records the energy a component uses when it's considered "on".

- EnergyStatic - records the static power of the component at the beginning of the simulation, and reports it as total static energy at the end.

To enable a StatObject to be collected, ask the Statistics class, for instance:

```
StatObject* P_static = Statistics::registerStat("myLeakagePower",
        StatObject::ENERGY_STATIC,"electronic");
```

returns a StatObject that can be used to record the electronic leakage power, which you would do by calling

```
P_static->track(0.014);
```

which would specify 14mW of leakage power.

4.2.1 Insertion Loss Optimization of Broadband Network Architectures

For broadband circuit-switched architectures like those discussed in Section 3.2, the topology and physical layer design can have a large impact on power and performance. For photonic networks with ring-based broadband switches (PSEs, as we have been calling them), up until now we haven't been entirely honest with you about insertion loss. This was for your own benefit, as you will see, so we could isolate some issues to talk about in architecture. Here, we are going to take a look at the real nature of the design of the physical layer.

Figure 4.14 * shows the through loss, drop loss, and extinction ratio of a PSE in both crystalline silicon and nitride/poly as a function of ring radius. In circuit-switched networks, more wavelengths is more desirable because it increases the network bandwidth. However, as we see here, doing so means we need a larger ring to fit more wavelengths into a single modulator FSR, and thus have much more through loss and lower extinction ratio (more crosstalk).

This means that, to really design the physical layer of a circuit-switched network, we need to find the optimal point of maximizing wavelengths while keeping a low power budget, or find the largest ring tolerated by the optical power budget. Since insertion loss is not a time-dependent characteristic, we don't actually need to simulate the network to find it. Here, it is preferable to instead come up with an

*The numbers for this figure were generated by Kyle Preston at Cornell University, 2010

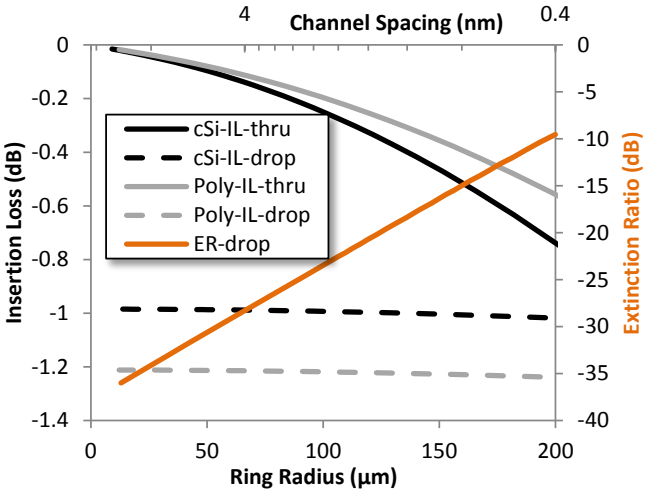


Figure 4.14: PSE characteristics versus ring radius

analytic model, so we don't have to re-run a simulation for each value of ring resonator radius. Lets do an example for our regular P-Mesh.

4.2.1.1 P-Mesh Optimization Example

For this analysis, we use the StraightPath switch from Figure 3.6(b) because, though it has more crossing loss, we want to show the benefit that multi-layer devices have on networks by eliminating those crossings. Considering the P-Mesh design of Figure 3.13, we come up with a closed form approximation for worst-case loss in the network:

$$\zeta_{network} = \zeta_{mod-det} + \zeta_{inj-ej} + (2N-1) \times (\zeta_{switch-thru} + \zeta_{prop}) + \zeta_{switch-drop} \tag{4.5}$$

where $\zeta_{mod-det}$ is insertion loss from passing through the modulator and filter bank, ζ_{inj-ej} is from dropping through rings to be injected into the network, $\zeta_{switch-thru}$ is passing straight through a switch (without dropping through a ring), ζ_{drop} is making a turn at a switch (dropping through a ring), and ζ_{prop} is the propagation loss (as a function of distance) between the switches. These terms are defined as follows:

$$\zeta_{mod-det} = \zeta_{mod} + 2\sigma_{filter} + \zeta_{filter-drop} \tag{4.6}$$

Table 4.1: Insertion Loss Optimization Parameters

Parameter	Symbol	Single-Layer (dB)	Multi-Layer (dB)
Modulation	ζ_{mod}	1.2	1.2
Filter-drop	$\zeta_{filter-drop}$	0.5	-
Filter-thru	$\zeta_{filter-thru}$	0.05	0.05
Crossing	ζ_{cross}	0.05	0
Waveguide	ζ_{wg}	0.5	0.5 (Si_3N_4) 5 (poly)
Bend	ζ_{bend}	0.005	0.005

$$\zeta_{inj-ej} = 2\zeta_{ring-drop} \tag{4.7}$$

$$\zeta_{switch-thru} = 6\zeta_{cross} + 4\zeta_{ring-thru} + 2\zeta_{bend} + \zeta_{wg} \times S_{switch} \times 1.5 \tag{4.8}$$

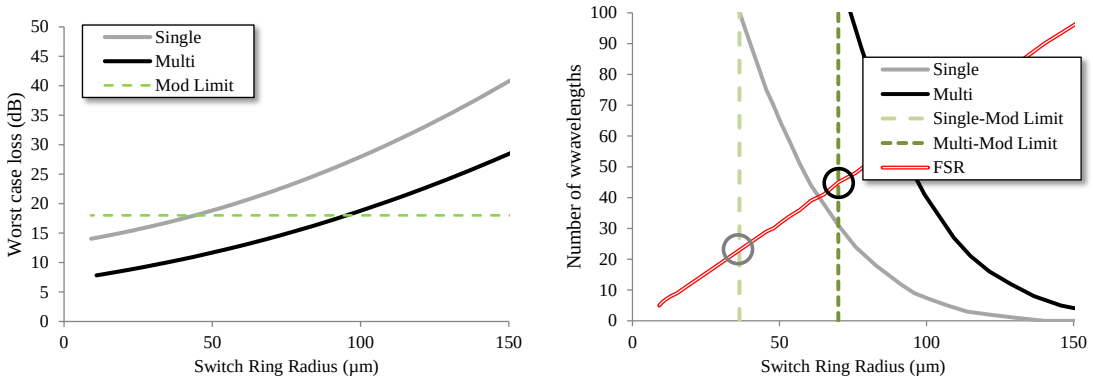
$$\zeta_{switch-drop} = 5\zeta_{cross} + \zeta_{ring-drop} + 2\zeta_{bend} + \zeta_{wg} \times S_{switch} \times 1.5 \tag{4.9}$$

$$\zeta_{prop} = (S_{chip}/N - S_switch)\zeta_{wg} \tag{4.10}$$

where S_{chip} and S_{switch} are the sizes (of one side) of the chip and switch, respectively, in the same units that waveguide propagation loss (ζ_{wg}) is expressed in. Referring to Figure 3.13 and Figure 3.6(b), you will agree this is a good approximation of worst-case loss. The parameters we used can be found in Table 4.1.

Now let's plot insertion loss for both single- and multi-layer devices for an 8×8 network, shown in Figure 4.15(a). We see that the multi-layer does much better because, as we saw from Section 3.2.6, waveguide crossings are a big deal. However, now we see how much ring radius impacts network insertion loss. Figure 4.15(b) brings it all together by showing the number of wavelengths possible in the optical power budget, the limit imposed by the modulator optical power, and the limit imposed by the modulator and filter FSR. The two circles show the fully-constrained number of wavelengths that are feasible, about twice as many for multi-layer devices.

Though this was just presented as an optimization exercise, this process of determining broadband ring loss is actually very important, and should be the standard practice when considering designs using this type of switching.



(a) Loss (b) Wavelengths

Figure 4.15: Optimization of P-Mesh versus ring radius showing (a) worst-case loss and (b) number of wavelengths feasible

4.2.1.2 Matrix Crossbar Optimization Example

Let's also take a look at the matrix crossbar topology from Section 3.2.6.2. We can also come up with a closed-form approximate expression for loss:

$$\zeta_{network} = \zeta_{mod-det} + \zeta_{crossings} + \zeta_{switch-thru} + \zeta_{switch-drop} + \zeta_{prop} \tag{4.11}$$

where $\zeta_{mod-det}$ is the same from the P-Mesh, $\zeta_{crossings}$ is the waveguide crossing loss, $\zeta_{switch-thru}$ is the loss from passing all the broadband ring switches, $\zeta_{switch-drop}$ is the loss from dropping through the single ring needed to select the correct column, and ζ_{prop} is the propagation loss. These can be expressed as follows:

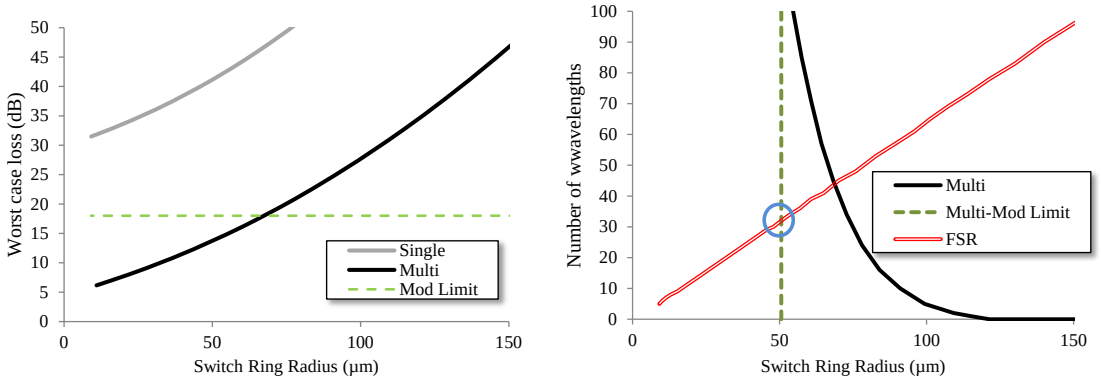
$$\zeta_{crossings} = 4N^2\zeta_{cross} + \frac{4N^2(N-1)}{N}\zeta_{cross} \tag{4.12}$$

$$\zeta_{switch-thru} = (N^2 - 1 + N(N-1))\zeta_{ring-thru} \tag{4.13}$$

$$\zeta_{switch-drop} = \zeta_{ring-drop} \tag{4.14}$$

$$\zeta_{prop} = 2S_{chip}\zeta_{wg} + 4\zeta_{bend} + \frac{(N-1)S_{chip}}{N}\zeta_{wg} \tag{4.15}$$

Recall that the insertion loss savings of the multi-layer network were huge compared to the single layer, mainly because we designed the topology not caring about waveguide crossings. Indeed, you



(a) Loss (b) Wavelengths

Figure 4.16: Optimization of Matrix Crossbar versus ring radius showing (a) worst-case loss and (b) number of wavelengths feasible

can see the N^2 terms in the expression for $\zeta_{crossings}$ which add up. Figure 4.16 shwos plots similar to those of the P-Mesh for both single- and multi-layer devices for an 8×8 network, using the same parameters in Table 4.1. We can see from Figure 4.16(a) that the single-layer network is nowhere near feasible given the modulator optical power limit, so we don't show it on Figure 4.16(b) for number of wavelengths feasible. Again, the number of wavelengths are limited by the modulator optical power limit and modulator FSR to around 30 wavelengths.

4.2.2 Insertion Loss Optimization of Wavelength-Routed Network Architectures

Wavelength-routed architectures, discussed in Section 3.3, also have insertion loss characteristics that are less than obvious. Here, we will take a look at the two architectures we mentioned before: Corona and Firefly.

4.2.2.1 Corona Insertion Loss

We can come up with an approximate expression for worst-case loss in Corona's crossbar network as follows:

$$\zeta_{network} = \zeta_{prop} + \zeta_{modulators} + \zeta_{filter} \tag{4.16}$$

where the components of this equation can be expressed as follows:

$$\zeta_{prop} = \zeta_{prop-wrap} + 3\zeta_{prop-tails} + (4(\frac{\sqrt{N}}{2} - 1) - 1)\zeta_{prop-btwn} + (\frac{N}{4} - 1)\zeta_{prop-group} \quad (4.17)$$

$$\zeta_{prop-wrap} = (\sqrt{N} - 2)\frac{S_{chip}}{\sqrt{N}}\zeta_{wg} + 2\zeta_{bend} \quad (4.18)$$

$$\zeta_{prop-tails} = 2\frac{S_{chip}}{\sqrt{N}}\zeta_{wg} + 2\zeta_{bend} \quad (4.19)$$

$$\zeta_{prop-btwn} = \frac{(S_{chip} - (S_{mod}N_\lambda\sqrt{N}))}{\sqrt{N}}\zeta_{wg} \quad (4.20)$$

$$\zeta_{prop-group} = S_{mod}N_\lambda\zeta_{wg} \quad (4.21)$$

$$\zeta_{modulators} = 4(\frac{N}{4} - 1)\zeta_{mod-thru} \quad (4.22)$$

Of course, these equations look a bit complicated, but if you are familiar with the Corona network, feel free to verify them. We broke the terms up into the modulator "groups", so that we could separately express the poly and silicon-nitride waveguide loss for the multi-layer version. Also, notice that a N_λ term is in there, which means that we need to iterate back and forth between determining the insertion loss and number of wavelengths.

Corona's token-ring network is actually a little easier to express:

$$\zeta_{network} = \zeta_{prop} + 2(N-2)\zeta_{filter-thru} + 2\zeta_{filter-drop} \quad (4.23)$$

where ζ_{prop} is similar to the crossbar network's propagation loss. For the token ring, however, the filter thru loss becomes extremely significant because it's an active filter (more like a very small broadband switch), and therefore much more lossy than a normal passive filter.

4.2.2.2 Firefly Insertion Loss

Firefly also requires active filters, and we can express one assembly's insertion loss as follows:

$$\zeta_{network} = \zeta_{prop} + (\sqrt{N} - 1)\zeta_{filter-thru} + \zeta_{filter-drop} \quad (4.24)$$

$$(4.25)$$

while similar in form to Corona's token ring network, is sufficiently scaled back by the $\sqrt{N}$ term so that it is entirely more feasible than Corona.

4.3 Layout and Synthesis

Up until now, discussion of integrated photonics on chip has been the analysis and design of photonic networks and architecture. In this final section, we will address work that has been done in the area of computer aided design for integrated nanophotonics. We will begin by describing a CAD tool VANDAL in Section 4.3.1, and from there describe some of the novel and interesting things we can accomplish with it.

4.3.1 VANDAL: A Photonic CAD Tool

VANDAL is a fully functional photonic component place and route tool complete with GUI, written in C#. VANDAL should be available at http://lightwave.ee.columbia.edu/vandal. The organization of VANDAL is shown in Figure 4.17. The Photonic Component Library captures key knowledge of the operation and geometries of various photonic devices (discussed in Section 4.3.1.4), of which a user can instantiate via the graphical user interface or through SCILL (discussed in Section 4.3.2). After a project consisting of devices and waveguides has been built, it can be exported to a variety of output formats used for different purposes (discussed in Section 4.3.1.7). The remainder of this section describes the main components of VANDAL and the novel design methods that it supports.

4.3.1.1 Device Parameterization

Unlike traditional CMOS layout tools which allow a user to draw any geometry on any fabrication layer, we programmatically create functional building blocks which can be instantiated much like traditional hierarchical standard cells. We parameterize the layout of these components so that a user can control aspects of a component's behavior while retaining the geometries necessary for its correct operation. Figure 4.18 shows an example of the parameterization of a ring-modulator. Each geometry's position in the component can be specified using any of the parameters. For example, the y-position of the top edge of the waveguide ring relative to the top edge of the modulator is specified as $topPitch + ridgeGap +$

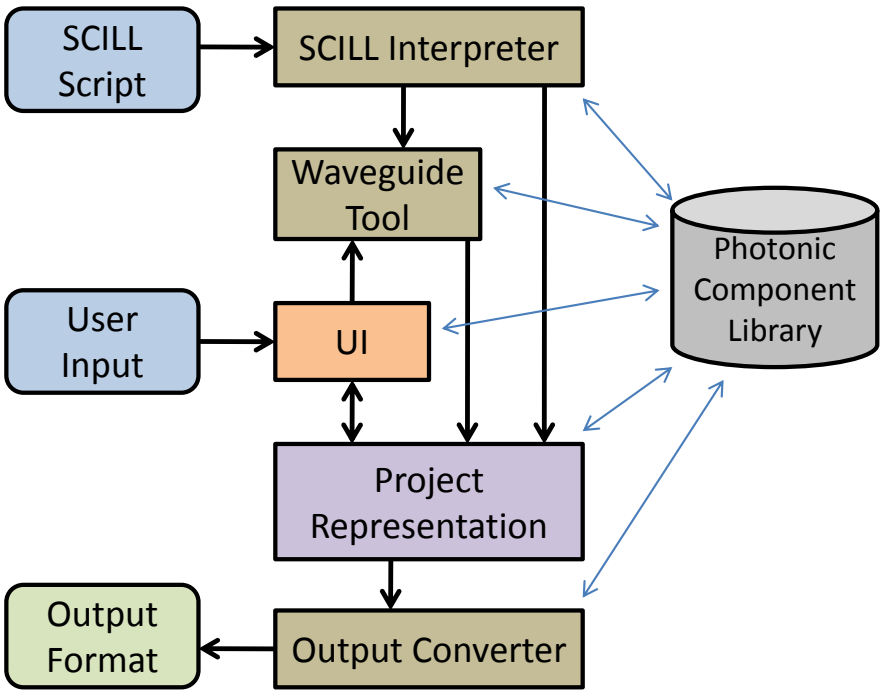


Figure 4.17: VANDAL composition block diagram

$dopingThickness + dopingGap$.

Table 4.2 lists each parameter and its description for the ring-modulator example in Figure 4.18. Component parameterization enables significant advantages over free-form geometrical layout. First, for example, if we change the value of ringD, the modulator is automatically resized by maintaining the relative proportions of all other geometries. This allows a user to rapidly configure, derive, and analyze alternative implementations. This also applies to an automation tool which could instantiate many components, controlling each precisely. Finally, parameterization enables verification and repeatability, providing a medium for mapping post-fabrication device characterization to precise parameters.

4.3.1.2 Device Modeling

One significant novelty of VANDAL is that components can describe their photonic transfer functions based solely on their geometries and materials. This allows a designer to see the specific impact on device functionality when changing different parameters, and enables high-level design automation flows. We incorporate models from PhoenixSim (68), a network simulator containing accurate physical-layer

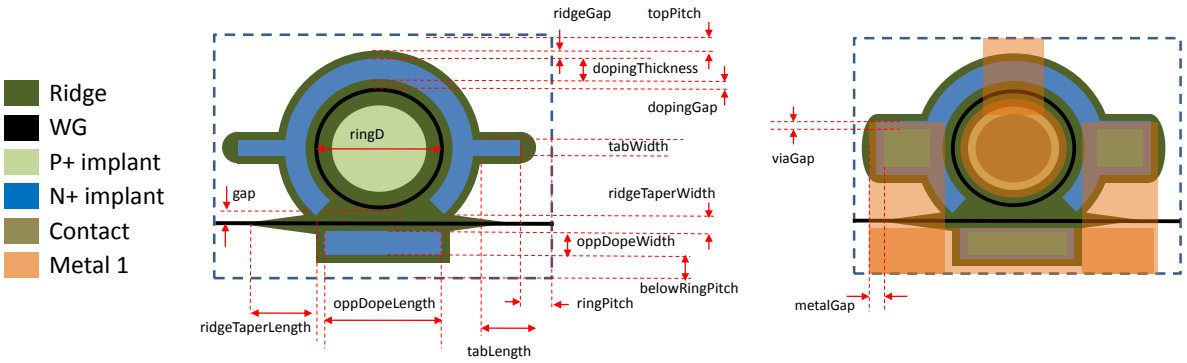


Figure 4.18: Example of component parameterization - ring modulator

Table 4.2: Ring Modulator Parameters

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

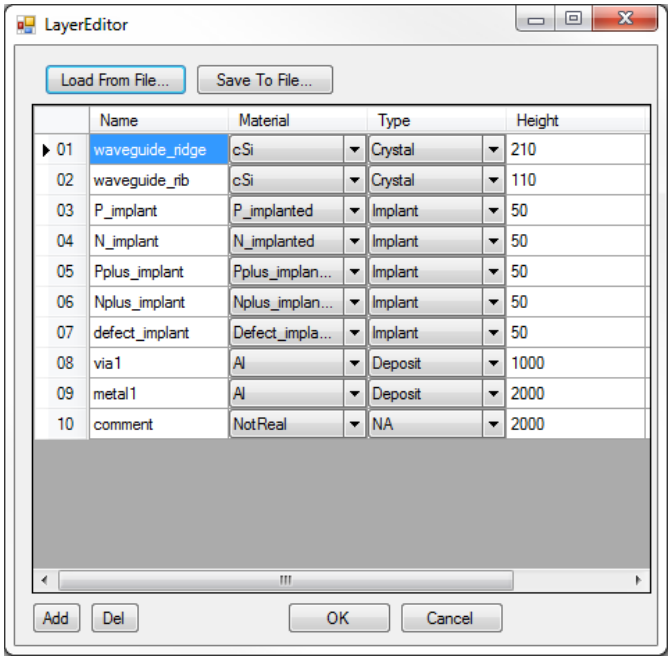


Figure 4.19: VANDAL's layer editor

models, into VANDALs photonic components. For the case of a ring resonator, as the one illustrated in Figure 4.18, we capture the resonance characteristic as a function of transmission wavelength based on Yariv's equation (28), discussed in Section 2.3.1. The optical power transmission coefficient, t, can be expressed from waveguide and gap dimensions from the Appendix in (29), thus providing a completely geometry and material dependent model. Capturing functional characteristics of photonic components also enables high-level design automation flows, such as reversing the modulator's equations or searching the parameter space for specific geometric dimensions that satisfy a given resonance profile.

4.3.1.3 Layers

VANDAL has support for specifying which layers the project is targeting in terms of fabrication process. This allows us to include the waveguide layers in crystalline silicon and support metal routing for contacting active areas. Figure 4.19 shows the layer editor. Editing layers also allows us to support deposited multi-layer devices, since components must specify which layer each of their geometries is on.

4.3.1.4 Photonic Component Library

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

Modulator There are four types of parameterized modulators, each with different characteristics: single-order (73), second-order for hitless operation (30), PINIP for higher speeds(31), and single-order attached to a Mach-Zehnder interferometer for more athermal operation (32).

Detector A single-wavelength filter/detector for compact layout based on a ring-resonator (42).

Coupler A waveguide-to-waveguide hybrid mode coupler, as described in (74).

Filter A ring-based wavelength filter with through and drop ports.

1x2, 2x2 Switch A broadband ring-resonator switch, as described in (40).

Taper For mode conversion, useful for coupling to off-chip fiber (75).

Waveguide, bends The fundamental medium for guiding light, with 450nm default width for single-mode low insertion-loss propagation (76).

Waveguide crossing Intersection of waveguides, engineered for low-loss (77).

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

4.3.1.5 Device Placement

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

The second method is by specifying the exact X-Y coordinates of the component. Coordinates are stored as unsigned positive integers, in nanometers. The third method is through the *connect* function, which aligns a component such that its indicated port is in the same location as another component's indicated port. These second and third methods are useful for automation or scripting processes, which will be discussed in more detail in Section 4.3.2.

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

4.3.1.6 Waveguide Tool

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

In the current version of VANDAL, components are considered to be in the same silicon substrate because of typical waveguide-fabrication techniques (19). Waveguide crossings are fabricated for low-loss and low crosstalk (77), but can still contribute a significant part of total network loss (78). Nevertheless, the geometry and pitch of waveguide crossings must also be taken into account in the path-finding process.

We implement this process with A* (A-Star) search (79), a common graph-search technique. This method requires that any node must be able to generate a finite set of successor nodes, and that traveling from one node to another has a defined cost metric (in our case, insertion loss of the waveguide added from node n_1 to node n_2). Here, we define the search space as a graph of nodes which can represent any x-y coordinate. The basic process is as follows, for any starting node n:

1. Check finish condition - if n is at the destination, exit;

2. Check quick solution;

3. Generate n's successors, adding them to the list of possible next moves;

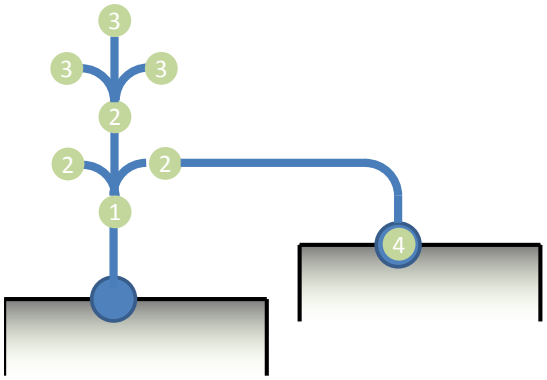


Figure 4.20: Example connection two North-facing ports

4. Sort the list by n's cost plus heuristic;

5. Repeat with next node, n'.

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

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

Consider the example in Figure 4.20, which shows the process of connecting two North-facing port of adjacent components. The iterations go as follows:

1. The only valid successor (1-N) to the starting point at the port.

2. North, East, and West successors are generated (2-N, 2-E, 2-W). Since bends have slightly higher insertion loss than straight waveguides, the method minimizes the cost by continuing North-ward, even though the East-facing successor would be closer to the destination.

3. North, East, and West successors are generated (3-N, 3-E, 3-W). Now the straight North-ward waveguide has overcome the loss by bending East-ward in Step (2), so 2-E is the new current node.

4. The *quick solution* directly to the destination becomes valid, and the process is completed.

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

4.3.1.7 Layout Output

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

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

NED The language used by OMNeT++ (69), and therefore PhoenixSim (68) to describe the instantiation and connectivity of simulation modules.

CIF - Caltech Interchange Format A standard format used to describe simple geometries (80), which can be imported into fabrication machines and standard CMOS layout tools.

Hotspot A thermal modeling tool, which takes simple geometry coordinates and size as inputs (81). This tool is compatible with PhoenixSim representation, enabling simulation of thermal fluctuations on the photonic components.

MEEP A tool for performing finite-difference time-domain (FDTD) electromagnetic simulations (82).

4.3.2 SCILL: A Layout Scripting Language

We developed SCILL (Simple Component Intuitive Layout Language) to automate the layout of photonic components such that the process is precise, convenient, flexible, repeatable, and insertable into

Table 4.3: SCILL Variable Types

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

an automation loop for optimization. SCILL is a language with C-like sequential execution, modifying a *namespace* containing variable definitions and values. SCILL also contains script-like directives to use features of VANDAL, such as the WGT. Unlike other representations which simply describe the *locations* of every element in the system, a SCILL script can describe the *process* by which elements are placed. This allows us to implement powerful automation methods, such as instantiating many elements with little code using loops.

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

1. Include - specify any external scripts that are called from this one;
2. Parameters - specify user-level parameters using the *param* keyword that can be input at run-time through VANDAL;
3. Execution - all other executable lines.

See Table 4.4 for descriptions of the primary statements defined in SCILL. SCILL variables are typed, according to the descriptions found in Table 4.3. SCILL also contains some useful built-in functions, as well as a mechanism for calling other scripts.

Table 4.4: SCILL Primitive Statements

Statement	Description	Format/Example
Assignment	Modifies the namespace by either declaring a new variable or naming an existing one, and updating its value. <TYPE> is required when declaring a previously-unused variable name. *value* can be a constant, variable, or result of a function.	<TYPE> [name] = [value]
New	Creates a new component of the type specified, and must be the name of a valid VANDAL component type (see Section 4.3.1.4).	COMP foo = new modulator
If	The usual control statement, ended with ENDIF. [*condition*] can use the usual ==, >=, <=, and ! = comparison operators.	IF [condition] ... ENDIF
For	The typical looping mechanism, simplified for integer-only, increment-by-1 operation. *var* is the loop variable, (implied integer) which ranges from *min* to *max*, inclusive.	FOR [var] in [min]..[max] ... ENDFOR
Foreach	A mechanism for iterating over an ARRAY variable. End with ENDFOR.	FOREACH <TYPE> [var] in [array]

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

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

rotate Simply rotates the given COMP variable 90 degrees clockwise. See Script 1 in Appendix B.1.0.1 for an example.

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

4.3.3 Case Study: Link Instantiation

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

In this case study, we first write a SCILL script to automatically generate an arbitrary number of modulators and detectors, which can be found in Script 1 in Appendix B.1.0.1. This script takes as parameters the number of modulators to instantiate, the minimum ring diameter, the change in diameter for adjacent rings, and whether to make modulators or detectors. We test our script by instantiating four modulators with minumum diameter of $5\mu m$ and a diameter change of $10nm$. The automatically generated resonance profile of the result can be seen in various screenshots in Figure 4.21, which also indicates important features such as free spectral range (FSR) and extinction ratio (ER).

Next, we write a script to use these gateways to instantiate photonic links, to connect cores or other IP across a chip. Script 2 in Appendix B.1.0.1 shows the example we use here, making 2 links spanning about $0.12mm^2$ using six and four wavelengths. Figure 4.22 shows the resulting screenshot, with the modulator banks in the upper-left corner, and the detector banks at the bottom and on the right side. Besides the transmission profile of the links, we can also automatically calculate the insertion loss for any link, which is 0.34 and 0.48 dB for this example.

4.3.4 Photonic Synthesis Methods

Now that we have a way of instantiating photonic devices and being able to determine their characteristics fairly easily, lets start to think about how we can do something useful. We'll start simply, by considering methods for implementing communication in 3D-stacked SoCs using photonic links. We assume 3D layers dedicated to computation cores or elements which can be separately laid out according to optimal area, power delivery, and thermal constraints. The locations of the electronic communication interfaces is therefore assumed to be fixed with regard to our photonic synthesis methodology. Along

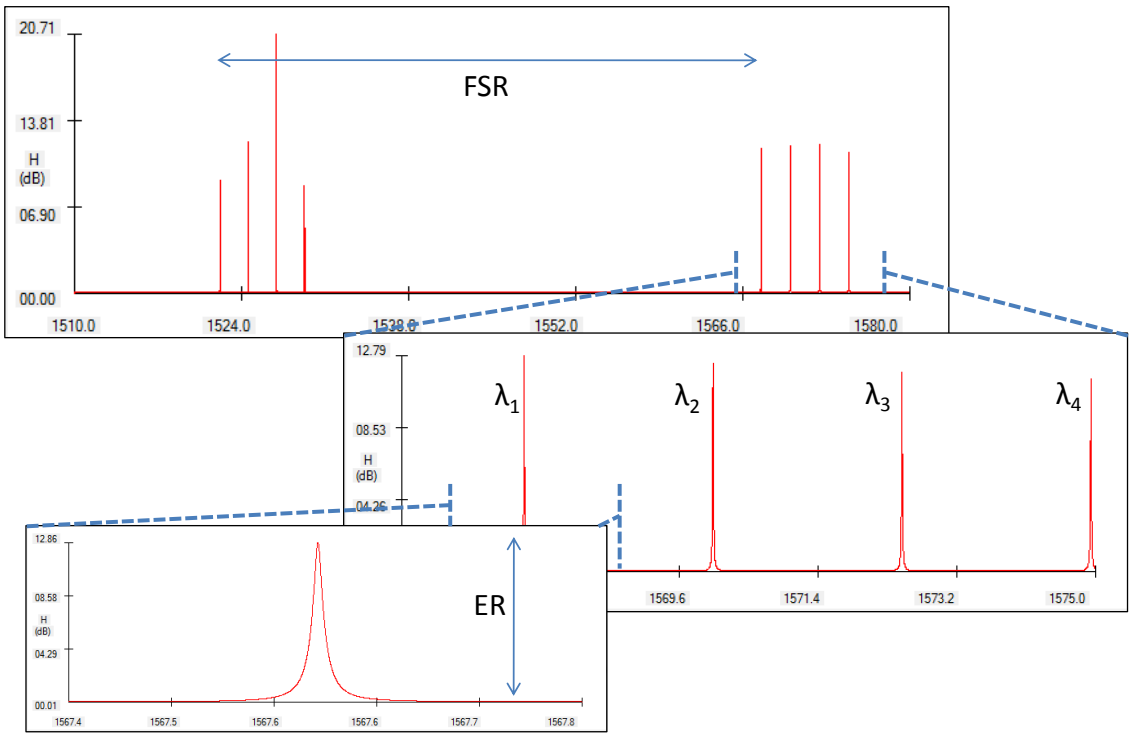


Figure 4.21: Screenshots of response of four-modulator WDM gateway, with freespace wavelength on the x-axes and loss (dB) on the y-axes

with communication bandwidth requirements, these two pieces of information make up our constraint/requirement specification, an example of which is illustrated in Figure 4.23.

4.3.4.1 Basic Link Synthesis

To enable optical communication between two cores of an SoC, we can design a photonic link between them which satisfies the bandwidth requirements. Consider the specification example in Figure 4.23. To fulfill the bandwidth requirement from core A to core B, we place two microring modulators at core A, each running at 10 Gb/s for separate wavelengths, and two detectors at core B to receive the signal. The resulting photonic link, shown in Figure 4.24 makes use of wavelength division multiplexing (WDM), or using multiple wavelengths in the same waveguide to encode separate signals, to achieve the required bandwidth with minimal time-of-flight latency for any distance on the chip (including going off chip). To satisfy all communication requirements between all cores, we can repeat this process for all source-destination pairs.

Though this is a seemingly simple solution, complex problem specifications may result in compli-

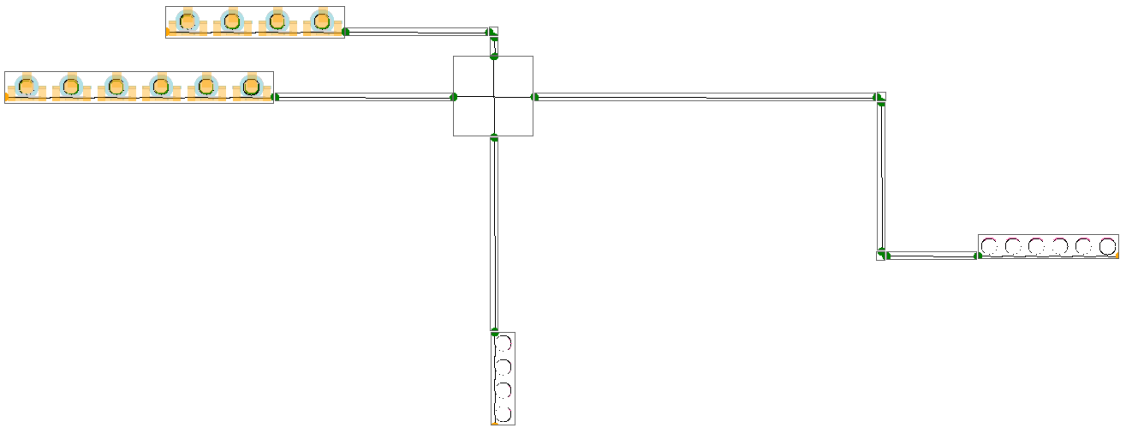


Figure 4.22: Screenshot of two photonic links

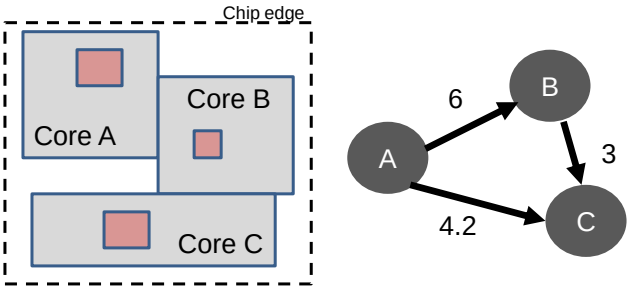


Figure 4.23: Example of a constraint-requirement specification of an SoC

cated photonic link layouts. Because here we consider only a single layer of silicon for instantiating photonic components due to typical waveguide fabrication techniques (19), waveguides that cross must have a specifically engineered region which minimizes loss and crosstalk (77). It is therefore in our interest to minimize the complexity, and particularly waveguide crossings, of the waveguide routing by arranging modulator and detector banks within gateways. The following problems present themselves when considering an algorithmic synthesis approach:

1. Where in the send and receive cores to place the modulators and detectors

2. How to optimally align the modulators and detectors (North-South, East-West)

3. Where to connect the modulators to a laser input at the chip edge

4. How to route waveguides between the laser input, modulators, and detectors

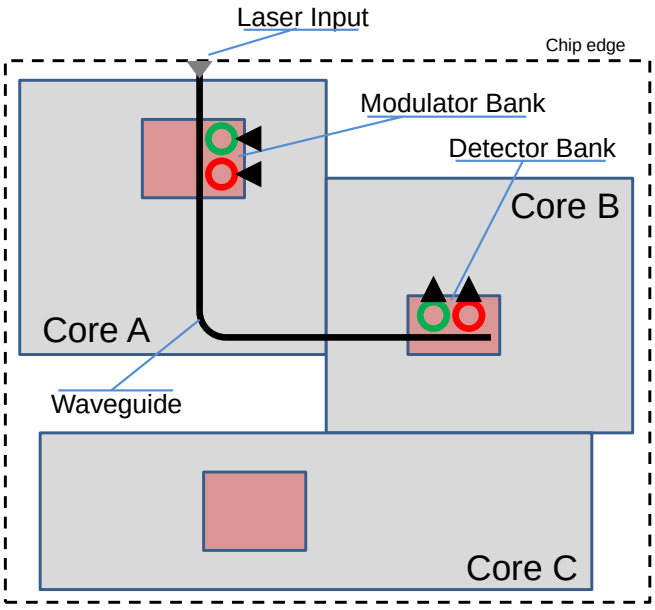


Figure 4.24: Basic photonic link instantiation

To simultaneously address items 1-3, we designate each core as *aligned* to its nearest chip edge. Each successive modulator or detector bank is placed with its input port closest to that edge, solving item 2 and making the solution to item 3 straight forward. Item 1 is also solved since all modulator and detector banks are regularly aligned in the same direction, making placing them an easy task. Figure 4.25 shows an example of placing modulator and detector banks in a gateway that is aligned to the top edge of the chip. Note that the third bank must rearrange itself to be able to fit within the gateway's constrained area.

4.3.4.2 Optimization: WDM Link Combination

One unique feature of silicon photonics is its ability to use WDM, combining multiple signals into one physical waveguide. In our approach, we can use WDM to combine links, saving on both input ports and the number of waveguides necessary to connect gateways, which ultimately saves insertion loss and therefore laser power. Combining links can occur in two forms: send-based and receive-based combination.

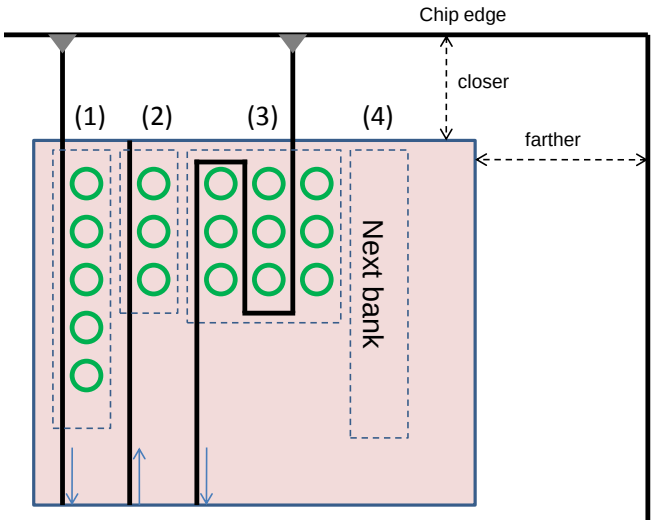
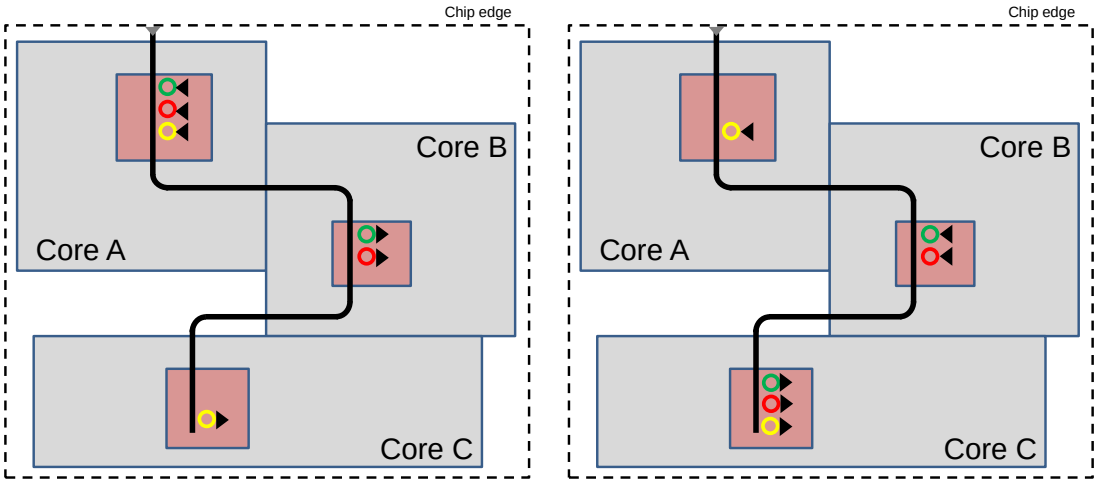


Figure 4.25: Basic layout of modulator and detector banks in a gateway



(a) Send-based (b) Receive-based

Figure 4.26: Link combination using WDM

Send-based combination If a sending core has multiple destinations which are located relatively close to each other, we can extend one link and route it to the next destination, tacking on the necessary modulators to the original modulator bank. Referring back to our original example in Figure 4.23 for a problem specification, this idea is illustrated in Figure 4.26(a). Core A must be able to send 2 GB/s to core B, and 0.2 GB/s to core C. Core A therefore needs 3 modulators (each at 10 Gb/s), two dedicated for Core B and one for Core C, thus making a single-write-multiple-read (SWMR) bus with a single waveguide. Note that the example in Figure 4.26(a) does not implement the communication between core B and core C, which would require a separate link. Though the idea of photonic SWMR buses is not itself new, automating the placement and routing of these buses is a non-trivial task for large communication graphs. Algorithm 1 in Appendix B.1.0.2 shows the psuedocode for satisfying requirements using send-based link combination.

Receive-based combination Receive-based link combination is similar to send-based combination, except we instantiate multiple-write-single-read (MWSR) buses instead, as shown in Figure 4.26(b). In that example, the communication from core A to core B is not implemented, requiring another link.

4.3.4.3 Post Processing: Gateway Reordering

Once all links are placed on the chip, depending on the order that banks are placed in gateways could result in connections that will inevitably lead to many waveguide crossings, and thus insertion loss and crosstalk. One way to improve the implementation is to reorder modulator and detector banks in the gateway to align to their connections. An example of this is shown in Figure 4.27. Note that laser input points are flexible, as they are not a part of the placement constraints.

To implement gateway reordering, we take the average of the locations that each port of each modulator and detector bank is connected to, and sort them by either the X- or Y-dimension, depending on the gateway alignment. This process can be found in Algorithm 2 in Appendix B.1.0.2.

4.3.5 Case Study: Video Processing

We implemented our automated synthesis methodology in VANDAL using SCILL. Our methodology performs send-based combination first of all links with greater than 2 communication pairs, then attempts

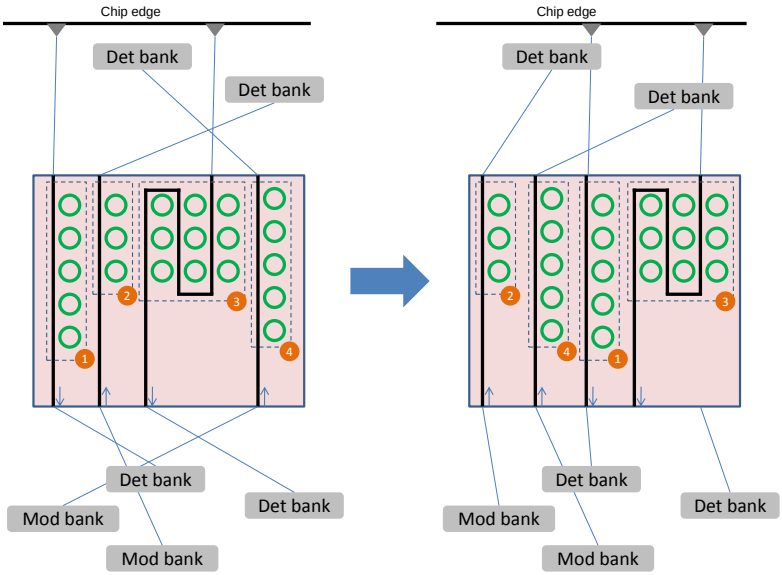


Figure 4.27: Gateway reordering for reducing waveguide crossings

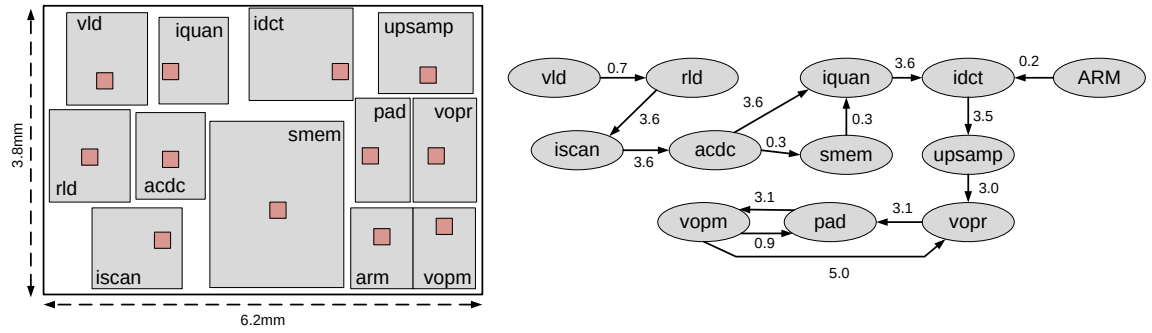
receive-based combination for all remaining links, and finally reorders the gateways. In this section, we present our solution on four video processing benchmarks.

4.3.5.1 Video Processing Benchmarks

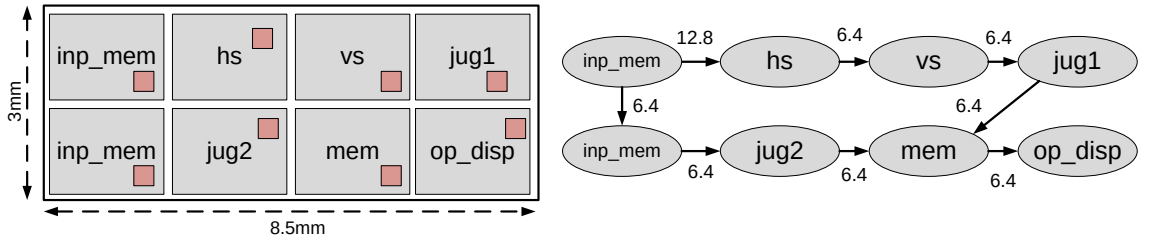
To test our automated synthesis methods, we use four standard MPSoC video-processing benchmarks: video object plane decoder (VOPD), picture-in-picture (PIP), MPEG4 decoder, and multi-window-display (MWD) (83, 84). Problem specifications for each benchmark can be seen in Figure 4.28. To scale the benchmarks to future SoCs, we use 10-100$\times$ the bandwidth requirements that have been used in previous work, requiring many gigabytes per second for each link in the communication graph. In addition, we add three off-chip DRAM nodes which communicate with the original ”SDRAM” memory controller in the MPEG4 benchmark, to illustrate how photonic links can easily be extended to off-chip communication with no additional complexity.

4.3.5.2 Power Estimation

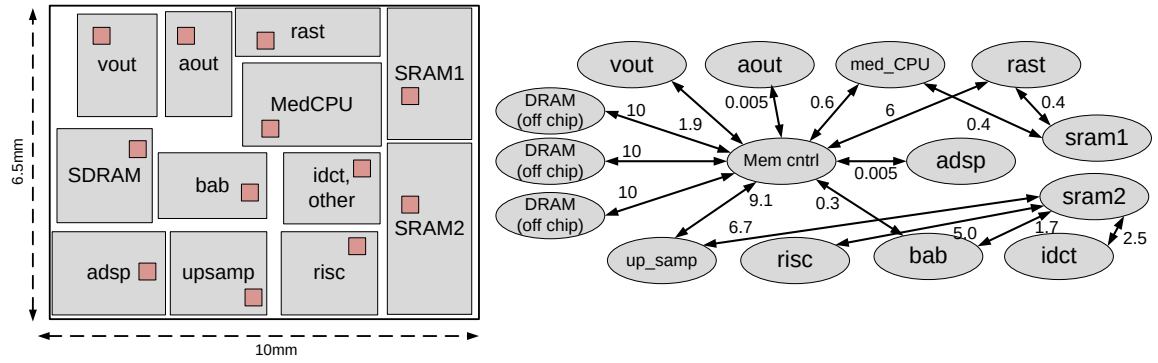
We can estimate the maximum power of the photonic communication subsystem by breaking it down into three components: laser power, driver power at full utilization, and receiver power at full utilization.



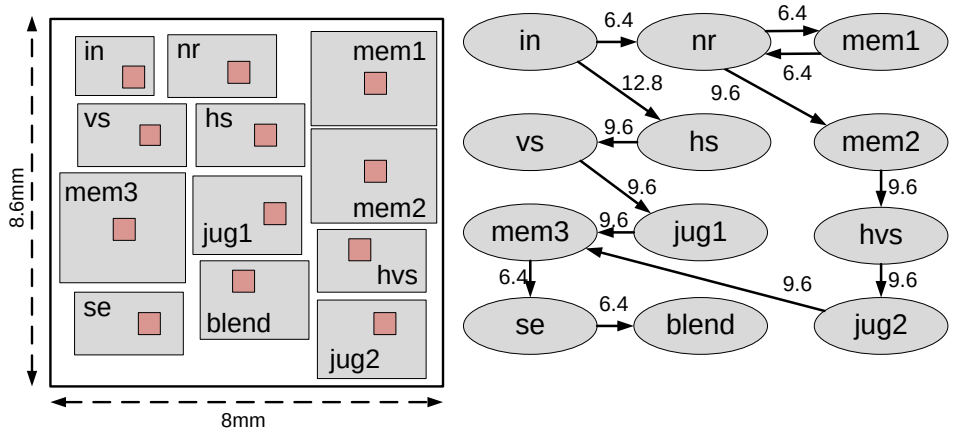
(a) VOPD



(b) PIP



(c) MPEG4



(d) MWD

Figure 4.28: SoC floorplans and communication graphs for 4 different video processing applications. Numbers on graphs indicate required bandwidth in GB/s. Colored regions indicate location of interface to photonic communication plane.

Table 4.5: Insertion Loss Parameters

Symbol	Parameter	Value
$IL_{coupler}$	Off-chip coupler	0.5 dB (85)
IL_{prop}	Waveguide propagation	1.5 dB/cm (86)
IL_{bend}	Waveguide bend	0.005 dB (86)
IL_{cross}	Waveguide crossing	0.1 dB (87)
IL_{thrus}	Ring thru loss	0.05 dB

The laser power can be estimated according to the following equation:

$$P_{laser} = \eta_{laser} \times \sum \lambda_i * 10^{(S_{det}+IL_i-30)/10} \tag{4.26}$$

where η_{laser} is the laser quantum efficiency, λ_i is the number of wavelengths (modulators and detectors) in link i, S_{det} is the detector sensitivity (in dBm), and IL_i is the worst-case insertion loss of link i (in dB). We can estimate the insertion loss for each link with the following equation:

$$IL_{link} = IL_{coupler} + IL_{wg} + \lambda \times IL_{thru} \tag{4.27}$$

where $IL_{coupler}$ is the loss from coupling the laser into the chip, $ILthru$ is the loss in a ring modulator or detector of passing wavelengths not on resonance, λ is the number of wavelengths in the link, and IL_{wg} is the insertion loss of traveling in the routed waveguides including propagation (IL_{prop}), bending ($ILbend$), and crossing ($ILcross$). Table 4.5 shows the parameters used for estimating link insertion loss.

Dynamic power at full utilization comes from driver/receiver circuit functionality, injecting carriers into the rings themselves, and circuit parasitics and leakage. The remaining consumption comes from thermally tuning the rings to correct for manufacturing variations and run-time thermal fluctuations, which we assume a conservative 100 μW per ring. Table 4.6 shows the parameters used for these estimations.

4.3.5.3 Results

We run our full automation script on the four benchmarks, and summarize the results in Table 4.7. Each solution took only a few seconds to run, which illustrates the computational advantage of decoupling the compute-plane floorplan from the communication-plane synthesis.

Table 4.6: Power Parameters

Parameter	Value
Laser efficiency (η_{laser})	30%
Detector sensitivity	-15 dBm
Thermal tuning	100 μW/ring
Driver (at full utilization)	320 fJ/bit (88)
Receiver (at full utilization)	690 fJ/bit (88)

Table 4.7: Results

	VOPD	MPEG4	PIP	MWD
IO Ports	12	14	8	11
Max wavelengths	5	14	11	17
Max Latency (ns)	0.04	0.08	0.024	0.06
Avg Loss (dB)	1.68	2.01	1.36	1.85
Max Loss (dB)	2.56	3.68	1.65	2.61
Laser Pwr (mW)	1.8	2.0	1.1	1.8
Static Pwr (mW)	3.3	10.2	5.3	9.7
Dynamic Pwr (mW)	333	1030	535	980
Total Power (W)	0.34	1.04	0.54	0.99

Max latency (time-of-flight) for all benchmarks is under 1 ns, illustrating one important advantage of instantiating photonic links for specific communication requirements. Power in each benchmark is dominated by the dynamic power of the high speed transceivers at full utilization, total power reaching a maximum of around 1 W for the entire photonic subsystem.

Max wavelengths is important to consider to ensure a reasonable wavelength spacing within one free spectral range of a modulator. In other words, if we use too many wavelengths, they will have to be packed closely together in the frequency domain which will cause interference when modulating and detecting. Our solutions used up to 17 wavelengths, which is well below dense wavelength division multiplexing (DWDM) standards. Max insertion loss is also important to ensure that enough optical power can be injected into the chip without inducing nonlinear effects in either the waveguides or modulators. Typical allowed max injected powers are around 10 mW (10 dBm), much higher than what is required in our solutions (S_{det} + IL_{total}, or -11.42 dBm for MPEG4).

5

Final Thoughts

So what did we learn? In this chapter we will summarize what happened, and see if it resembles what we originally set out to do. In Chapter 2, we learned about some nano-scale photonic devices in both crystalline silicon and deposited materials. These devices are extremely interesting because they enable the manipulation of light on such a small, precise scale. Ring-resonators are a highly useful structure for WDM because of their narrow wavelength response, enabling highly-selective functionality and densely-packed channel spacing.

5.1 Summary and Conclusions

In Chapter 3, we took a look at two network architecture design paradigms: circuit-switching and wavelength-arbitrated. Circuit-switched networks are similar to traditional purely electronic ones because of their spatial layout and behavioral characteristics. While seemingly clunky because of the overhead of setting up the circuit path, circuit-switched networks could hold promise for applications that need high bandwidth both between cores and off-chip memory. Given the probable move away from purely shared memory systems, circuit-switched photonic NoCs could be a good solution for a good part of the computing space. Referring to Figure 3.32, as cache/buffer line size grows, circuit-switching can grow with it as long as accesses are managed (not totally random).

Some design improvements to circuit switching were also introduced. Concentration, though not a new idea itself, is important because it reduces the size of the network, which can reduce the amount of

power by a factor of 2× or more (Figure 3.26), while keeping performance essentially the same (Figure 3.24). Selective transmission is also important to consider, as it can greatly increase the saturation load, which reduces latencies by orders of magnitude at high arrival rates (Figure 3.29). Time-division multiplexing is also a good alternative to pure circuit-switching because it can sustain 2× or more bandwidth (Figure 3.35) due to implementing fairness in the arbitration and out-of-order transmission at the gateways.

Wavelength-arbited networks could be great for low latency global communication. They take advantage of WDM, something which doesn't exist for electronics, to encode information in the wavelength that is being used. While a network like Corona may never be feasible, simpler designs exist which hold promise for the latency-bandwidth-power tradeoffs that need to take place.

One key thing about photonic networks in general, as we saw, is insertion loss. If the network design requires that we inject so much laser power that nonlinear effects are incurred in any part of the photonic circuit, then the network becomes completely unreliable. Designing for lower insertion loss also means designing for lower network power, as much of the power is usually spent in the lasers. For a simple mesh, 80-90% is laser power (Figure 3.21(b)), though multi-hop networks like the Enhanced TDM can show much less (Figure 3.43).

Along these lines, we saw one of the benefits of multi-layer deposited devices. Silicon nitride can provide even lower loss than crystalline silicon, and can be deposited onto either existing CMOS logic or other waveguides, completely avoiding crossings. Silicon nitride also has a much higher nonlinear threshold, which is important for networks that require many wavelengths. However, silicon nitride cannot be made "active." We therefore have to vertically couple to either crystalline silicon or, using another deposited material, poly-silicon. Though poly-silicon has higher loss, it does have some favorable characteristics such as the carrier depletion time and absorbing characteristics. When put together, deposited materials benefit both the feasibility and the likely performance of chip-scale photonics. For circuit-switched networks, a factor of 4× scaling is typical using deposited photonics over crystalline/-SOI (Figures 3.44 and 3.46).

Crosstalk is also a serious issue. While the waveguide medium does not incur inter-wavelength crosstalk by itself, modulating, switching, crossing, and especially filtering all contribute noise power due to imperfect extinction ratios and significant resonance-channel spacing leakage. Ultimately, more

crosstalk means lower SNR, which means retranmissions and/or error detection and correction, both of which are costly in terms of both power and performance. Solving crosstalk issues through both design and device improvements may provide more difficult than anything else.

In Chapter 4, we looked at the tools and methods we use to investigate photonic interconnections. We presented the development of PhoenixSim, a chip-scale system-level simulator which models various characteristics of physical devices, while enabling quick prototyping of various networks running various applications. As with any simulation, we tried to maximize the balance between simulation accuracy, simulation time, and flexibility. As proof of our work, any experiment presented in this work is available to run in PhoenixSim.

We also presented VANDAL, a layout and synthesis tool specifically for integrated nanophotonic devices. By developing a library of functional components, we can parameterize and model photonic devices in a way that is easier for a designer to instantiate, more accurate post-fabrication, and can be automated for high-level and large-scale synthesis. By having this tool, we can quickly design a network, simulate in system-level simulations like PhoenixSim, simulate in physical-level simulation like MEEP, and output to fabrication format like CIF all from the same representation. As one use case or validation of the tool, we demonstrated how it can synthesize photonic links for SoCs which support $10\times$ more bandwidth than a purely electronic solution for the same amount of power (Table 4.7). VANDAL is also available to the reader at the Lightwave Research Lab website.

5.2 Opportunities and Challenges for Photonics

This work has mainly addressed the tools and methods necessary for the design of photonic interconnection networks on chip. It has not directly motivated or de-motivated the use of photonics in any particular way, because in reality that is a question with many more factors than simply engineering, such as cost and marketing. However, some of the opportunities of photonics are clear.

Despite the many papers discussing photonics as a complete NoC solution, electronics is available now and will continue to be a viable solution in the near future. However, because IO bandwidth is already constrained by package area/pin-count, this is the largest opportunity for photonics. In effect, no good electronic solution exists which can address the bandwidth/power IO problem that is being

presented by high-performance large-scale computing as well as embedded computing. As we saw in Section 3.2.4, photonics supports orders of magnitude higher bandwidth density than electronics due to the use of WDM. Power will also be one of the major motivators, especially for large-scale computing where reducing the power for chip-to-DRAM communication can have an enormous impact.

To support the successful integration of photonics in this way, key device requirements must be met. First, a low-loss low-cost packaging solution will have to emerge. Since laser power already dominates much of the power dissipation for photonic links, and is always present regardless of link use, keeping photonic links low loss is critical. Considering that lasers are only 10% efficient, reliable coupling losses below 2 dB are necessary for power consumption to be competitive with photonics. Also, higher efficiency comb laser sources for WDM are necessary, preferably small, cheap, and integrated.

Ring-based devices also have some unsolved challenges, mainly their intolerance to both fabrication variations and temperature changes. Using ohmic heaters are one possible solution, though ring-resonator based structures may never achieve the kind of black-box reliability that transistors have, or at least not with full reliability. This will not be a show-stopper for photonics in general, as mach-zehnder modulators could be used which are more tolerant to temperature.

As the technology matures, it is entirely likely that the problems and challenges seen here will be solved. Or completely new devices and structures will be formulated. It is also entirely feasible that a new technology will prove more effective and more useful than integrated silicon photonics, such as free-space photonics or carbon nanotubes. Either way, the same rigor that was applied in this work should continue to be applied to the design and modeling of chip-scale interconnects.

References

[1] PETER M. KOGGE (ED). **Exascale Computing Study: Technology Challenges in Achieving Exascale Systems**. Technical Report TR-2008-13, University of Notre Dame CSE Department, 2008. v, 2, 4

[2] F. SCHELLENBERG. **A little light magic [optical lithography]**. *Spectrum, IEEE*, **40**(9):34 – 39, September 2003. 1

[3] M.S. BAKIR, C. KING, D. SEKAR, H. THACKER, BING DANG, GANG HUANG, A. NAEEMI, AND J.D. MEINDL. **3D heterogeneous integrated systems: Liquid cooling, power delivery, and implementation**. In *Custom Integrated Circuits Conference, 2008. CICC 2008. IEEE*, pages 663 –670, sep. 2008. 1

[4] JAE-SEOK YANG, KRIT ATHIKULWONGSE, YOUNG-JOON LEE, SUNG KYU LIM, AND DAVID Z. PAN. **TSV stress aware timing analysis with applications to 3D-IC layout optimization**. In *Proceedings of the 47th Design Automation Conference*, DAC '10, pages 803–806, New York, NY, USA, 2010. ACM. 1

[5] B. DANG, M.S. BAKIR, AND J.D. MEINDL. **Integrated thermal-fluidic I/O interconnects for an on-chip microchannel heat sink**. *Electron Device Letters, IEEE*, **27**(2):117 – 119, feb. 2006. 1

[6] J.P. COLINGE. **Multi-gate SOI MOSFETs**. *Microelectronic Engineering*, **84**(9-10):2071 – 2076, 2007. INFOS 2007. 2

[7] NIR MAGEN AND AVINOAM KOLODNY. **Interconnect-power Dissipation in a Microprocessor**. In *in Proceedings of the International Workshop on System-Level Interconnect Prediction*, pages 7–13, 2004. 2

[8] **The International Technology Roadmap for Semiconductors (ITRS)**. http://www.itrs.net. 2, 3

[9] SHANE BELL, BRUCE EDWARDS, JOHN AMANN, ET AL. **TILE64 Processor: A 64-Core SoC with Mesh Interconnect**. In *Proceedings of the IEEE International Solid-State Circuits Conference*, pages 88–89, 598, 2008. 2, 5

[10] DOE OFFICE OF SCIENCE. **The Opportunities and Challenges of Exascale Computing**, 2010. 4

[11] WILLIAM CARLSON, TAREK EL-GHAZAWI, BOB NUMRICH, AND KATHY YELICK. **Programming in the Partitioned Global Address Space Model**. Presented at Supercomputing 2003. Online at http://www.gwu.edu/upc/tutorials.html. 4

[12] NATHAN FARRINGTON, GEORGE PORTER, SIVASANKAR RADHAKRISHNAN, HAMID HAJABDOLALI BAZZAZ, VIKRAM SUBRAMANYA, YESHAIAHU FAINMAN, GEORGE PAPEN, AND AMIN VAHDAT. **Helios: a hybrid electrical/optical switch architecture for modular data centers**. *SIGCOMM Comput. Commun. Rev.*, **40**:339–350, August 2010. 5

[13] CAROLINE LAI, DANIEL BRUNINA, AND KEREN BERGMAN. **Demonstration of 8x40-Gb/s Wavelength-Striped Packet Switching in a Multi-Terabit Capacity Optical Network Test-Bed**. In *PHO Annual*, Nov 2010. 5

[14] LONG CHEN, KYLE PRESTON, SASIKANTH MANIPATRUNI, AND MICHAL LIPSON. **Integrated GHz silicon photonic interconnect with micrometer-scale modulators and detectors**. *Optics Express*, **17**(17), August 2009. 5

[15] ANDREW ALDUINO ET AL. **Demonstration of a High Speed 4-Channel Integrated Silicon Photonics WDM Link with Hybrid Silicon Lasers**. In *Proceedings of Hot Chips*, Aug. 2010. 5, 12, 85

[16] R. M. KNOX AND P. P. TOULIOS. **Integrated circuits for millimeter through optical frequency range**. In *Proceedings of Submillimeter Waves Symposium*, pages 497–516, Mar. 1970. 10

[17] B.G. LEE, A. BIBERMAN, A.C. TURNER-FOSTER, M.A. FOSTER, M. LIPSON, A.L. GAETA, AND K. BERGMAN. **Demonstration of Broadband Wavelength Conversion at 40 Gb/s in Silicon Waveguides**. *Photonics Technology Letters, IEEE*, feb. 2009. 10

[18] ALEKSANDR BIBERMAN, BENJAMIN G. LEE, AMY C. TURNER-FOSTER, MARK A. FOSTER, MICHAL LIPSON, ALEXANDER L. GAETA, AND KEREN BERGMAN. **Wavelength multicasting in silicon photonic nanowires**. *Opt. Express*, **18**(17):18047–18055, Aug 2010. 10

[19] BENJAMIN G. LEE, XIAOGANG CHEN, ALEKSANDR BIBERMAN, XIAOPING LIU, I-WEI HSIEH, CHENG-YUN CHOU, JERRY DADAP, RICHARD M. OSGOOD, AND KEREN BERGMAN. **Ultra-high-Bandwidth WDM Signal Integrity in Silicon-on-Insulator Nanowire Waveguides**. *IEEE Photonics Technology Letters*, **20**(6):398–400, May 2007. 10, 28, 116, 123

[20] JAIME CARDENAS, CARL B. POITRAS, JACOB T. ROBINSON, KYLE PRESTON, LONG CHEN, AND MICHAL LIPSON. **Low loss etchless silicon photonic waveguides**. *Opt. Express*, **17**(6):4752–4757, Mar 2009. 10

[21] MICHAEL J. SHAW, JUNPENG GUO, GREGORY A. VAWTER, SCOTT HABERMEHL, AND CHARLES T. SULLIVAN. **Fabrication techniques for low-loss silicon nitride waveguides**. *Micromachining Technology for Micro-Optics and Nano-Optics III*, **5720**(1):109–118, 2005. 10

[22] VILSON R. ALMEIDA, ROBERTO R. PANEPUCCI, AND MICHAL LIPSON. **Nanotaper for compact mode conversion**. *Opt. Lett.*, **28**(15):1302–1304, Aug 2003. 11

[23] LONG CHEN, PO DONG, AND MICHAL LIPSON. **High performance germanium photodetectors integrated on submicron silicon waveguides by low temperature wafer bonding**. *Opt. Express*, **16**(15):11513–11518, Jul 2008. 11

[24] LONG CHEN AND MICHAL LIPSON. **Ultra-low capacitance and high speed germaniumphotodetectors on silicon**. *Opt. Express*, **17**(10):7901–7906, May 2009. 11

[25] LING LIAO, DEAN SAMARA-RUBIO, MICHAEL MORSE, ANSHENG LIU, DEXTER HODGE, DORON RUBIN, ULRICH KEIL, AND THORKILD FRANCK. **High speed silicon Mach-Zehnder modulator**. *Opt. Express*, **13**(8):3129–3135, Apr 2005. 12

[26] S. MANIPATRUNI, QIANFAN XU, B. SCHMIDT, J. SHAKYA, AND M. LIPSON. % bf High Speed Carrier Injection 18 Gb/s Silicon Micro-ring Electro-optic Modulator. In . *The 20th Annual Meeting of the IEEE Lasers and Electro-Optics Society (LEOS).*, pages 537 –538, 2007. 13

[27] ALEKSANDR BIBERMAN, SASIKANTH MANIPATRUNI, NOAM OPHIR, LONG CHEN, MICHAL LIPSON, AND KEREN BERGMAN. **First demonstration of long-haul transmission using silicon microring modulators**. *Opt. Express*, **18**(15):15544–15552, Jul 2010. 13

[28] A. YARIV. **Universal relations for coupling of optical power between microresonators and dielectric waveguides**. *Electronics Letters*, **36**(4):321 –322, February 2000. 14, 114

[29] B. E. LITTLE, S. T. CHU, H. A. HAUS, J. FORESI, AND J.-P. LAINE. **Microring Resonator Channel Dropping Filters**. *Journal of Lightwave Technology*, **15**(6), June 1997. 14, 114

[30] HUGO L. R. LIRA, SASIKANTH MANIPATRUNI, AND MICHAL LIPSON. **Broadband hitless silicon electro-optic switch for on-chip optical networks**. *Opt. Express*, **17**(25):22271–22280, 2009. 14, 115

[31] SASIKANTH MANIPATRUNI, QIANFAN XU, AND MICHAL LIPSON. **PINIP based high-speed high-extinction ratio micron-size silicon electrooptic modulator**. *Opt. Express*, **15**(20):13035–13042, 2007. 14, 115

[32] BISWAJEET GUHA, BERNARDO B. C. KYOTOKU, AND MICHAL LIPSON. **CMOS-compatible athermal silicon microring resonators**. *Opt. Express*, **18**(4):3487–3493, 2010. 14, 115

[33] KYLE PRESTON, SASIKANTH MANIPATRUNI, ALEXANDER GONDARENKO, CARL B. POITRAS, AND MICHAL LIPSON. **Deposited silicon high-speed integratedelectro-optic modulator**. *Opt. Express*, **17**(7):5118–5124, Mar 2009. 14

[34] MARIA GÖPPERT-MAYER. **Über Elementarakte mit zwei Quantensprngen**. *Annalen der Physik*, **401**(3):273–294, 1931. 14

[35] HARTMUT HAUG AND STEPHAN W. KOCH. *Quantum theory of the optical and electronic properties of semiconductors. 5th ed.* World Scientific, 2009. 14

[36] KISHORE PADMARAJU, NOAM OPHIR, SASIKANTH MANIPATRUNI, CARL B. POITRAS, MICHAL LIPSON, AND KEREN BERGMAN. **DPSK Modulation Using a Microring Modulator**. In *Proceedings of Conference on Lasers and Electro-Optics (CLEO)*, 2011. 15

[37] SHIJUN XIAO, MAROOF H. KHAN, HAO SHEN, AND MINGHAO QI. **Multiple-channel silicon micro-resonator based filters for WDM applications**. *Opt. Express*, **15**(12):7489–7498, Jun 2007. 15

[38] H.T. PHILIPP, K.N. ANDERSEN, W. SVENDSEN, AND H. OU. **Amorphous silicon rich silicon nitride optical waveguides for high density integrated optics**. *Electronics Letters*, **40**(7):419 – 421, 2004. 15

[39] CHRISTOPHER J. KAALUND. **Critically coupled ring resonators for add-drop filtering**. *Optics Communications*, **237**(4-6):357 – 362, 2004. 15

[40] BENJAMIN G. LEE, ALEKSANDR BIBERMAN, NICOLAS SHERWOOD-DROZ, CARL B. POITRAS, MICHAL LIPSON, AND KEREN BERGMAN. **High-Speed 2×2 Switch for Multiwavelength Silicon-Photonic Networks-On-Chip**. *Journal of Lightwave Technology*, **27**(14), July 2009. 16, 115

[41] A. BIBERMAN, B.G. LEE, N. SHERWOOD-DROZ, M. LIPSON, AND K. BERGMAN. **Broadband Operation of Nanophotonic Router for Silicon Photonic Networks-on-Chip**. *Photonics Technology Letters, IEEE*, 2010. 16

[42] J. K. DOYLEND, P. E. JESSOP, AND A. P. KNIGHTS. **Silicon photonic resonator-enhanced defect-mediated photodiode for sub-bandgap detection**. *Opt. Express*, **18**(14):14671–14678, 2010. 16, 115

[43] KYLE PRESTON, YOON HO DANIEL LEE, MIAN ZHANG, AND MICHAL LIPSON. **Waveguide-integrated telecom-wavelength photodiode in deposited silicon**. *Opt. Lett.*, **36**(1):52–54, Jan 2011. 16

[44] ASSAF SHACHAM, KEREN BERGMAN, AND LUCA CARLONI. **On the Design of a Photonic Network-on-Chip**. In *First International Symposium on Networks-on-Chip*, 2007. 24

[45] HOWARD WANG, BENJAMIN G. LEE, ASSAF SHACHAM, AND KEREN BERGMAN. **On the Design of a 4x4 Nonblocking Nanophotonic Switch for Photonic Networks on Chip.** In *Proceedings of Frontiers in Nanophotonics and Plasmonics*, 2007. 26

[46] JOHNNIE CHAN, ALEKSANDR BIBERMAN, BENJAMIN G. LEE, AND KEREN BERGMAN. **Insertion loss analysis in a photonic interconnection network for on-chip and off-chip communications**. In *IEEE Lasers and Electro-Optics Society (LEOS)*, Nov. 2008. 26

[47] XUEZHE ZHENG ET AL. **Ultra-low energy all-CMOS modulator integrated with driver.** *Optics Express*, **18**(3):3059–3070, 2010. 30

[48] TOBIAS GENSTY, WOLFGANG ELSÄSSER, AND CHRISTIAN MANN. **Intensity noise properties of quantum cascade lasers**. *Opt. Express*, **13**, 2005. 33

[49] CHRISTOPHER MILLER. *Fiber Optic Test and Measurement*, chapter 7. Prentice Hall, 1998. 33

[50] PRABHAT KUMAR, YAN PAN, JOHN KIM, GOKHAN MEMIK, AND ALOK CHOUDHARY. **Exploring concentration and channel slicing in on-chip network router**. In *NOCS '09: Proceedings of the 2009 3rd ACM/IEEE International Symposium on Networks-on-Chip*, pages 276–285, Washington, DC, USA, 2009. IEEE Computer Society. 46

[51] GILBERT HENDRY ET AL. **Analysis of Photonic Networks for a Chip-Multiprocessor Using Scientific Applications**. In *The 3rd ACM/IEEE International Symposium on Networks-on-Chip*, May 2009. 50

[52] G. HENDRY, E. ROBINSON, V. GLEYZER, J. CHAN, L. P. CARLONI, N. BLISS, AND K. BERGMAN. **Circuit-switched Memory Access in Photonic Interconnection Networks for High-Performance Embedded Computing.** In *International Conference for High Performance Computing, Networking, Storage, and Analysis (Supercomping)*, Nov. 2010. 53, 98

[53] GILBERT HENDRY ET AL. **Silicon Nanophotonic Network-On-Chip Using TDM Arbitration**. In *Proceedings of IEEE Symposium on High-Performance Interconnects*, Aug. 2010. 55, 57, 62

[54] MIKAEL MILLBERG, ERLAND NILSSON, RIKARD THID, AND AXEL JANTSCH. **Guaranteed Bandwidth Using Looped Containers in Temporally Disjoint Networks within the Nostrum Network on Chip**. In *DATE '04: Proceedings of the conference on Design, automation and test in Europe*, page 20890. IEEE Computer Society, 2004. 56

[55] KEES GOOSSENS, JOHN DIELISSEN, AND ANDREI RADULESCU. **Æthereal Network on Chip: Concepts, Architectures, and Implementations**. *IEEE Des. Test*, **22**(5):414–421, 2005. 56

[56] MARTIN SCHOEBERL. **A Time-Triggered Network-on-Chip**. In *International Conference on Field-Programmable Logic and its Applications (FPL 2007)*, pages 377 – 382, Amsterdam, Netherlands, August 2007. 56

[57] ZHONGHAI LU AND AXEL JANTSCH. **TDM virtual-circuit configuration for network-on-chip**. *IEEE Trans. Very Large Scale Integr. Syst.*, **16**(8):1021–1034, 2008. 56

[58] CHRISTIAN PAUKOVITS AND HERMANN KOPETZ. **Concepts of Switching in the Time-Triggered Network-on-Chip**. In *RTCSA '08: Proceedings of the 2008 14th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications*, pages 120–129. IEEE Computer Society, 2008. 56

[59] ANDERS EDMAN AND CHRISTER SVENSSON. **Timing closure through a globally synchronous, timing partitioned design methodology**. In *DAC '04: Proceedings of the 41st annual Design Automation Conference*, pages 71–74. ACM, 2004. 59

[60] J-F. ZHENG ET AL. **On-chip optical clock signal distribution**. In *OSA Topical Meeting on Optics in Computing*, 2003. 59

[61] V. GLEYZER J. CHAN L. P. CARLONI N. BLISS G. HENDRY, E. ROBINSON AND K. BERGMAN. **Time-Division-Multiplexed Arbitration in Silicon Nanophotonic Networks-On-Chip for High-Performance Chip Multiprocessors.** *Journal of Parallel and Distributed Computing*, Jan 2011. 61

[62] J. SCHRAUWEN, F. VAN LAERE, D. VAN THOURHOUT, AND R. BAETS. **Focused-Ion-Beam Fabrication of Slanted Grating Couplers in Silicon-on-Insulator Waveguides**. *IEEE Photonics Technology Letters*, **19**(11):816–818, June 2007. 65

[63] GÜNTHER ROELKENS, DRIES VAN THOURHOUT, AND ROEL BAETS. **Continuous-Wave Lasing from DVS-BCB Heterogeneously Integrated Laser Diodes**. In *Integrated Photonics and Nanophotonics Research and Applications*, page ITuG4. Optical Society of America, 2007. 68

[64] A. BIBERMAN, K. PRESTON, G. HENDRY, J. CHAN, N. SHERWOOD-DROZ, J. LEVY, M. LIPSON, AND K. BERGMAN. **Photonic Network-on-Chip Architecture Using Multi-Layer Deposited Silicon Materials for High-Performance Chip Multiprocessors.** 69

[65] DAN-XIA XU, ANDRÉ DELÂGE, ROSS MCKINNON, MARTIN VACHON, RUBIN MA, JEAN LAPOINTE, ADAM DENSMORE, PAVEL CHEBEN, SIEGFRIED JANZ, AND JENS H. SCHMID. **Archimedean spiral cavity ring resonators in silicon as ultra-compact optical comb filters**. *Opt. Express*, **18**(3):1937–1945, 2010. 72

[66] DANA VANTREASE, ROBERT SCHREIBER, MATTEO MONCHIERO, MORAY MCLAREN, NORMAN P. JOUPPI, MARCO FIORENTINO, AL DAVIS, NATHAN BINKERT, RAYMOND G. BEAUSOLEIL, AND JUNG HO AHN. **Corona: System Implications of Emerging Nanophotonic Technology**. In *Proceedings of 35th International Symposium on Computer Architecture*, Aug 2008. 77, 78

[67] YAN PAN, PRABHAT KUMAR, JOHN KIM, GOKHAN MEMIK, YU ZHANG, AND ALOK CHOUDHARY. **Firefly: illuminating future network-on-chip with nanophotonics**. *SIGARCH Comput. Archit. News*, **37**(3):429–440, 2009. 80

[68] JOHNNIE CHAN ET AL. **PhoenixSim: A Simulator for Physical-Layer Analysis of Chip-Scale Photonic Interconnection Networks.** In *DATE: Design, Automation, and Test in Europe.*, Mar. 2010. 85, 112, 118

[69] ANDRAS VARGA. **OMNeT++ Discrete Event Simulation System**. Online: http://www.omnetpp.org. 86, 118

[70] V. PUENTE, R. BEIVIDE, J. A. GREGORIO, J. M. PRELLEZO, J. DUATO, AND C. IZU. **Adaptive Bubble Router: a design to improve performance in torus networks**. In *Proc. Of International Conf. On Parallel Processing*, pages 58–67, 1999. 89

[71] H. WANG ET AL. **ORION: A Power-Performance Simulator for Interconnection Networks**. In *35th International Symposium on Microarchitecture*, 2002. 89

[72] DAVID WANG ET AL. **DRAMsim: A memory-system simulator**. *SIGARCH Computer Architecture News*, **33**(4):100–107, Sept. 2005. 96, 142

[73] KYLE PRESTON, SASIKANTH MANIPATRUNI, ALEXANDER GONDARENKO, CARL B. POITRAS, AND MICHAL LIPSON. **Deposited silicon high-speed integrated electro-optic modulator**. *Opt. Express*, **17**(7):5118–5124, 2009. 115

[74] T. TAMIR AND ELSA. GARMIRE. *Integrated optics*, chapter 8. Springer-Verlag, Berlin ; New York :, 1975. 115

[75] VILSON R. ALMEIDA, ROBERTO R. PANEPUCCI, AND MICHAL LIPSON. **Nanotaper for compact mode conversion**. *Opt. Lett.*, **28**(15):1302–1304, 2003. 115

[76] JAIME CARDENAS ET AL. **Low loss etchless silicon photonic waveguides**. *Opt. Express*, **17**(6):4752–4757, 2009. 115

[77] WIM BOGAERTS, PIETER DUMON, DRIES VAN THOURHOUT, AND ROEL BAETS. **Low-loss, low-cross-talk crossings for silicon-on-insulator nanophotonic waveguides**. *Opt. Lett.*, **32**(19):2801–2803, 2007. 115, 116, 123

[78] J. CHAN, G. HENDRY, A. BIBERMAN, AND K. BERGMAN. **Architectural Exploration of Chip-Scale Photonic Interconnection Network Designs Using Physical-Layer Analysis**. *Lightwave Technology, Journal of*, **28**(9):1305 –1315, may. 2010. 116

[79] P.E. HART, N.J. NILSSON, AND B. RAPHAEL. **A Formal Basis for the Heuristic Determination of Minimum Cost Paths**. *Systems Science and Cybernetics, IEEE Transactions on*, **4**(2):100 –107, July 1968. 116

[80] ROBERT W. HON AND CARLO H. SEQUIN. *A Guide to LSI Implementation*, chapter 7: A CIF Primer, pages 79 – 123. Xerox Palo Alto Research Center, 1980. 118

[81] WEI HUANG ET AL. **An Improved Block-Based Thermal Model in HotSpot 4.0 with Granularity Considerations**. In *Proceedings of the Workshop on Duplicating, Deconstructing, and Debunking*, 2007. 118

[82] ARDAVAN F. OSKOOI ET AL. **MEEP: A flexible free-software package for electromagnetic simulations by the FDTD method**. *Computer Physics Communications*, **181**:687–702, Jan 2010. 118

[83] E.G.T. JASPERS AND P.H.N. DE WITH. **Chip-set for video display of multimedia information**. *Consumer Electronics, IEEE Transactions on*, **45**(3):706 –715, aug. 1999. 127

[84] ERIK B. VAN DER TOL AND EGBERT G. T. JASPERS. **Mapping of MPEG-4 decoding on a flexible architecture platform**. In *Media Processors 2002*, pages 1–13, 2002. 127

[85] VILSON R. ALMEIDA, ROBERTO R. PANEPUCCI, AND MICHAL LIPSON. **Nanotaper for Compact Mode Conversion**. *Optics Letters*, **28**(15):1302–1304, August 2003. 129

[86] F. XIA, L. SEKARIC, AND Y. VLASOV. **Ultracompact optical buffers on a silicon chip**. *Nature Photonics*, **1**:65–71, January 2007. 129, 140

[87] TATSUHIKO FUKAZAWA, TOMOHISA HIRANO, FUMIAKI OHNO, AND TOSHIHIKO BABA. **Low Loss Intersection of Si Photonic Wire Waveguides**. *Japanese Journal of Applied Physics*, **43**(2):646–647, 2004. 129, 140

[88] HIREN D. THACKER AND OTHERS. **Flip-chip integrated silicon photonic bridge chips for sub-picojoule per bit optical links**. In *Electronic Components and Technology Conference (ECTC), 2010 Proceedings 60th*, pages 240 –246, jun. 2010. 130

[89] M. R. WATTS. **Ultralow Power Silicon Microdisk Modulators and Switches**. In *5th Annual Conference on Group IV Photonics*, 2008. 140

[90] AJAY JOSHI, CHRISTOPHER BATTEN, YONG-JIN KWON, SCOTT BEAMER, IMRAN SHAMIM, KRSTE ASANOVIC, AND VLADIMIR STOJANOVIC. **Silicon-Photonic Clos Networks for Global On-Chip Communication**. In *3rd ACM/IEEE International Symposium on Networks-on-Chip*, May 2009. 140

[91] BENJAMIN G. LEE, ALEKSANDR BIBERMAN, PO DONG, MICHAL LIPSON, AND KEREN BERGMAN. **All-Optical Comb Switch for Multiwavelength Message Routing in Silicon Photonic Networks**. *IEEE Photonics Technology Letters*, **20**(10):767–769, May 2008. 140

Appendix A

Running Simulations with PhoenixSim

As of this writing, PhoenixSim should be available at http://lightwave.ee.columbia.edu/phoenixsim. If not, contact me and I will find it for you. Along with it should be a user manual, which will describe how to install and use the simulator. In this appendix, we assume some familiarity with using and running the simulator for brevity's sake, and only describe how to set up the main parameter file, and what the more important parameters mean, to run the same experiments that were run to obtain the results in Chapter 3.

A.1 Insertion Loss Analysis

One of the basic tools in PhoenixSim is to do an insertion loss analysis, by running the "all2all" application, which sends a message from every node to every other node. Insertion loss for each transmission is tracked, and the worst one is reported, which tells us the constraints we must design for. Table A.1 shows the most important insertion loss parameters that we typically use for devices (which can also be found in parameters/default/optical_realisitc_parameters.ini). Though a simulation is not actually required for an insertion loss analysis because insertion loss is not time-dependent, it is still a convenient tool once the network model has been set up.

Table A.1: Optical Device Loss Parameters

Device	Insertion Loss
Chip size	$400mm^2$
Waveguide Propagation	1.5 dB/cm **
Waveguide Crossing	0.15††
Waveguide Bend	0.005 dB/90°**
Passing by Ring (Off)	0.005‡‡
Insertion into Ring (On)	0.5‡‡

Most of the interesting simulations in PhoenixSim involve exploring a design space, typically by simulating different scenarios by doing a parameter sweep. While it is possible to run each of the simulations individually within the OMNeT 4 Eclipse IDE, it is much more convenient to run them all at once in batch mode. Referring to the OMNeT user manual (http://www.omnetpp.org/doc/omnetpp41/manual/usman.html) section 9.4.3, we can run multiple PhoenixSim simulations with different parameters by running:

opp_runall -j4 ./PhoenixSim -u Cmdenv -f parameters/yourParameterFile -c yourNetworkConfiguration -r rangeOfRuns

examples will be shown as needed. The -j4 term says how many concurrent simulations to run, so set this to the number of processors or cores in your machine. The -u Cmdenv command says to run only in the console, don't use the GUI. Sometimes, there can be a large number of runs. It would be advisable to write a script or other program which can parse the .csv-based results files to collect information you want automatically.

A.1.1 Example from Section 3.2.2.5

The parameter file for looking at the insertion loss of the P-Mesh (Figure 3.13) is parameters/InterconnectsBook/A.1.1-InsertionLoss.ini. After opening the simulator executable, choose the PhotonicMesh configuration. There should be 24 run number options to choose from. These correspond to the different

*Dynamic energy calculation based on carrier density, 50-μm ring, 320×250-nm waveguide, 75% exposure, 1-V bias.

†Based on switching energy, including photon lifetime for re-injection.

‡Same as *, for a 3 μm ring modulator.

§Based on experimental measurements in (89). Calculated for half a 10 GHz clock cycle, with 50% probability of a 1-bit.

¶Conservative approximation assuming femto-farad class receiverless SiGe detector with $C < 1fF$.

‖Same value as used in (90). Average of 20 degrees thermal tuning required.

**From (86)

††Projections based on (87)

‡‡Projections based on (91)

combinations of network size 2×2 through 16×16), and switch implementation (original, straight path, and symmetric). So on your command line, run:

opp_runall -j4 ./PhoenixSim -u Cmdenv -f parameters/InterconnectsBook/A.1.1-InsertionLoss.ini -c PhotonicMesh -r 0..23

A.2 Performance Characterizations

Investing performance (and power) is what PhoenixSim is really for. Here we describe how to run these simulations.

A.2.1 Example from Section 3.2.2.5 - Photonic Mesh

A.2.1.1 Latency, Bandwidth, and Power

Here we want to investigate the latency-bandwidth characteristics of the P-Mesh for different random applications. This can be done using the parameter file in parameters/InterconnectsBook/A.2.1-Performance.ini. At the top of the file is where we specify the application to run, in this case random, neighbor, hotspot, tornado, and bitreverse. Each of these will instantiate different application classes which model these different communication patterns. The parameters are arrival rate (appParam1), the number of messages (appParam3), and message size (appSizeParam1). If you want to run for a specific amount of simulation time and not fixed number of messages (like we do in our results), set appParam3 to -1, and set sim-time-limit to the simulation time limit. Run away:

opp_runall -j4 ./PhoenixSim -u Cmdenv -f parameters/InterconnectsBook/A.2.1-Performance.ini -c PhotonicMesh -r 0..149

One very important thing to notice here is that some of the simulations may take a very long time (such as the 1E-7 arrival rate ones). You may not actually have to run these to get the full meaningful region of the latency-bandwidth characteristics curves, you only have to run until the network saturates. This requires some trial and error, but you won't have to run arrival rates smaller than 1E-7, and most are saturated at 1E-6. Also, you may need more data points than the ones included in the .ini file to get a nice smooth curve, which will also take some trial and error.

A.2.1.2 Crosstalk

Measuring crosstalk is a little bit different, because it requires us to use a more detailed model for the photonic devices. For this, we set useWDM = true (refer to parameters/InterconnectsBook/A.2.1-Crosstalk.ini). Note that these simulations do take a lot longer to run, which is why we ran fewer of them in our discussion of crosstalk results for the P-Mesh.

A.2.2 Example from Section 3.2.3.1 - Concentration

The parameter file for running the concentration experiment is A.2.2-Concentration.ini. Here we experimented with the signaling rate, because it could make a difference by having to implement a bandwidth-matched electronic crossbar for the gateway switch. To change the signaling rate, uncomment the correct section which has **.O_..._data = ... lines in them. These are the technology parameters passed to Orion for power modeling. To change the concentration, change the **.processorConcentration parameter.

A.2.3 Example from Section 3.2.3.2 - Selective Transmission

The selective transmission experiments contained a ton of data. The parameter file is A.2.3-Selective.ini. There are sections at the top which can be commented out to select each size distribution as well as selection policy. Each distribution-policy combination involves 216 separate simulations, so get yourself a simulation server or cluster.

A.2.4 Example from Section 3.2.4 - Memory

Our memory experiments were fairly limited, because we want to get into a lot of application-specific discussions. However, the A.2.4-Memory.ini parameter file allows you to investigate circuit-switched memory access by running some random memory accesses. For the circuit-switched networks, the DRAM subsystem is configured with .NED parameters because we use a separate memory model. For the electronic packet-switched network, DRAMsim (72) is used, as so we specify the location to its parameter file with **.DRAM_config_file.

A.2.5 TDM

Just like with the other performance simulations, we're going to run A.2.5-TDM.ini, which has concentrated 4×4 network configurations for the TDM network, the P-Mesh, and an electronic E-Mesh for comparison. Random traffic is selected, though others can be run also. One important parameter is the tdmSlotPeriod. It is set to 30ns, though that is rather long, and a smaller one can be used to decrease latency when large messages are not expected.

Appendix B

Design Automation with VANDAL

B.1 Using VANDAL

B.1.0.1 Case Study from Section 4.3.3

Script 1 below instantiates some number of modulators or detectors sequentially along a waveguide, changing the radius slightly so that the resonant modes will be different. Parameters are number of devices to instantiate, starting ring diameter, and the amount to change each ring diameter by.

Script 1 SCILL code for modulatorand detector instantiation

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

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

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

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

Script 2 SCILL code for link instantiation

```
include makeGW

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

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

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

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

Script 2 instantiates a couple photonic links, in arbitrary locations. It calls Script 1 above to make the modulator and detector banks.

Algorithm 1 Communication graph synthesis with send-based link combination

```
for all sources s_i do
  for all destinations of s_i: d_{i,j} do
    add to bucket
  end for
  currentLink = s_i
  while !bucket.empty() do
    bucket.sort()
    if getDistance(currentLink, bucket.front()) < D_thresh then
      add link to chain
      currentLink = bucket.front()
      bucket.pop()
    else
      place all modulators at send core, connect to laser
      for each link in chain do
        place detectors in receive core
        connect to previous link
      end for
    end if
  end while
end for
```

B.1.0.2 Synthesis Methods from Section 4.3.4

Algorithm 1 shows the pseudo-code for doing send-based link combination, an optimization to SoC synthesis. Of course, we have the SCILL code implementation of this pseudo code, but trust me, you don't want to see it. It's pretty long and confusing. This pseudo-code basically iterates through all the required communication, and attempts to lump them together based on their spatial locality using a simple threshold.

Algorithm 2 Reordering banks within a gateway

```
for each gw in gateways do
  for each b in gw.banks do
    b.avgLocation = average(b.port[0].connectedTo, b.port[1].connectedTo)
  end for
  if gw.alignment == HORIZONTAL then
    gw.banks.sortByY()
  else
    gw.banks.sortByX()
  end if
  for each b in gw.banks do
    place b in it's new place
  end for
end for
```

Another useful optimization is bank reorering for reducing waveguide crossings. Script 2 shows the pseudo-code for this, basically just sorting the banks in a gateway by the average location of the modules connected to their ports.