Silicon Nanophotonic Network-On-Chip Using TDM Arbitration

Gilbert Hendry[1], Johnnie Chan[1], Shoaib Kamil[2], Lenny Oliker[2], John Shalf[2], Luca P. Carloni[3], and Keren Bergman[1]

[1]Lightwave Research Laboratory, Columbia University, New York, NY
[2]CRD/NERSC, Lawrence Berkeley National Laboratory, Berkeley, CA
[3]Department of Computer Science, Columbia University, New York, NY

***Abstract*—Silicon nanophotonics is an emerging technology platform for offering high-bandwidth connectivity with extreme energy efficiency for future networks-on-chip. Using circuit-switching as an arbitration mechanism takes advantage of the low transmission energy in end-to-end communication and high bandwidth density of waveguides using WDM. However, pure circuit-switching requires an electronic control network which suffers from unfairness under heavy loads and can lead to high latencies, low network utilization, and an overhead in power dissipation. We propose time division multiplexed distributed arbitration, which provides round-robin fairness to setting up photonic circuit paths. Our design can supply 2-4× the bandwidth at network saturation for random traffic, and is an order of magnitude more efficient when simulated with scientific application traces compared to both electronic and other photonic network architectures.**

I. Introduction

Current trends in computer architecture suggest that the network-on-chip (NoC) will play a critical role in future high-performance microprocessors. How this role is played out will impact many areas of computing, including the programming models, architecture designs and manufacturing.

Though electronics offers a convenient and cheap medium-term solution for networks-on-chip, it is becoming apparent that purely electronic solutions may not scale to solve all the communication challenges in future high-performance chip multiprocessors (CMPs). The electronic interconnect is already responsible for up to 50% of dynamic power on-chip [1], a number that is too high given that the NoC is central to performance scaling through CMP parallelism.

Photonics offers key advantages in these areas, and has recently gained consideration as the leading emerging communications technology of future high-performance CMPs. Photonics can achieve a bandwidth density orders of magnitude higher than electronics using wavelength division multiplexing (WDM), a technique of simultaneously transmitting many optical signals on different wavelengths in parallel on a single transmission line. In addition, unlike electronics, the energy dissipated for optical signal propagation through waveguides and switches is independent of the datarate.

Recent numerous advances in silicon photonic integration and the emerging field of CMOS photonics [2]–[6] allows us to consider practical designs of full-scale interconnects using this technology platform. Many such novel photonic-enabled network architectures have been recently proposed that can deliver performance improvement over equivalent electronic interconnect designs [7]–[13].

In this work, we propose an all-optical broadband network architecture that uses time-division multiplexing (TDM) to arbitrate photonic communication channels. We describe the problem of statically scheduling photonic network transmissions in TDM slots, and implement a genetic algorithm to search the solution space for a near-optimal solution. We evaluate an instantiation of our network design for a 64-core network using both random traffic and traces of different scientific applications, and show that our TDM design achieves higher resource utilization and lower energy per bit compared with equivalent electronic and previously proposed photonic network architectures.

II. Related Work

Research into photonic circuit-switched networks-on-chip has progressed in the past few years, leading to a more complete understanding of the challenges both at the system level and the device level. Many advances have been made towards the integration of silicon nanophotonic devices into the traditional CMOS production line [2]. Ring-resonators have become a prevalent building block for broadband spatial switches, wavelength filters, and modulators because of their compact area and low power consumption [14].

System-level implications of photonics have made a large impact on the way architects are thinking of future CMPs. Reducing the cost of cross-chip, off-chip, and chip-chip communication allows a system designer to rethink programming models, memory hierarchy, and cost-performance optimization.

Next-generation NoC designs using silicon nanophotonic technology have been proposed in other work. The Corona network is an example of an architecture that uses optical arbitration via a wavelength-routed token ring to reserve access to a full serpentine crossbar made from redundant waveguides, modulators, and detectors [7]. Similarly, wavelength-routed bus-based architectures have been proposed which take advantage of WDM for arbitration [8], [9].

Batten *et al.* proposed an architecture for off-chip communications which takes advantage of WDM to dedicate wavelengths to different DRAM banks, forming a large ring-resonator matrix as a central crossbar [10]. Phastlane was designed for a cache-coherent CMP, enabling snoop-broadcasts and cacheline transfers in the optical domain [11].

Finally, much work has been done using TDM with optical communications for multiprocessor networks [15]–[17]. While many of the principles discussed in these works can still apply to the NoC design space, some of the mechanisms required to implement them may be infeasible or unfavorable on the chip-scale.

III. Photonic Circuit-Switching

On-chip hybrid photonic networks which use electronics for circuit-switching control have been proposed by Shacham *et al.* [18] and Petracca [12]. The fundamental switching unit in these designs is the photonic switching element (PSE), which is able to shift the periodic resonance of the device to align with the optical signals present in the nearby waveguide by injecting carriers through a p-n junction. This operation is shown in Figure 1.

These PSEs are strictly spatial switches, much like conventional electronic ones, which means paths from one port to another must be arbitrated before data can be sent through them. The arbitration function is further complicated by the fact that no photonic equivalent of a buffer exists, requiring that the path be completely set up from source to destination before data can be passed through the network. This challenge has been solved by using a conventional packet-switched electronic control network which circuit-switches a photonic data plane [18]. The idea behind this method is that once the optical path is set up between two nodes, the transmission of the data can amortize the setup latency with high-bandwidth WDM.

However, dynamically allocated circuit-switching via an electronic control network contains no implicit mechanism which ensures fairness, and can lead to degraded performance due to path blocking if messages are short or interfere with each other [19]. This work improves on these designs by removing the electronic control network responsible for allocating network resources and replacing it with a time-division-multiplexing distributed arbitration of photonic switches.

IV. TDM Arbitration

We propose using TDM to arbitrate end-to-end photonic circuit paths in a network of ring-resonator based photonic switches. 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

$$t_{slot} = t_{setup} + t_{transmission} + t_{propagation} \quad (1)$$

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

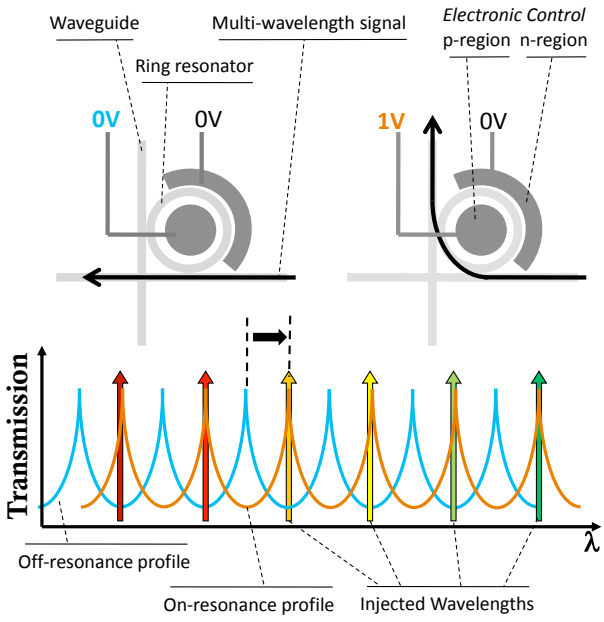


Fig. 1: **PSE operation, switching from OFF to ON state, shifting wavelengths.**

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 [20]–[24]. 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

$$T_{TDM} = N_{slot} \times t_{slot} \quad (2)$$

Our design 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

$$N_{slot} = N \times (N-1) \tag{3}$$

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. As we will see, it is possible to statically allocate the network to many transmissions during a single time slot.

A. TDM Scheduling using Genetic Algorithm

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 can accomplish this by searching the solution space using a *genetic algorithm*.

Genetic algorithms attempt to optimize the solution to a problem by simulating the process of evolution on a *population* of valid solutions [25]. Starting with an initial population, genetic algorithms iteratively apply *selection*, *reproduction*, and *mutation* methods until a specified number of *generations* have been simulated, or a sufficient solution has been found. Candidate solutions to the problem must have a *fitness* function, which allows the comparison of members of the population. In our case, a solution is a set of time slots containing individual communications which collectively implement full network communication coverage. We define the following methods according to our problem:

a) Initialization: Our initial population is generated from the naive solution by randomly merging two time slots. Valid merges must adhere to source, destination, and topology constraints throughout the genetic algorithm process. This is done until the population contains P valid solutions.

b) Selection: We employ *tournament selection* for ease of implementation [26]. A *tournament* is begun by selecting a subset k_{TS} of the population at random. Fitness is measured for each solution, which is the number of time slots in the solution (lower number = better fitness). The best solution is chosen with probability p_{TS}, the second best is chosen with probability $p_{TS} \times (1 - p_{TS})$, the third best is chosen with probability $p_{TS} \times (1 - p_{TS})^2$, and so on. Another tournament is started until the selection reaches a proportion p_S of the population size P. This selection process has a higher chance of selecting the more fit solutions, but can still include some that are not as fit to maintain population diversity.

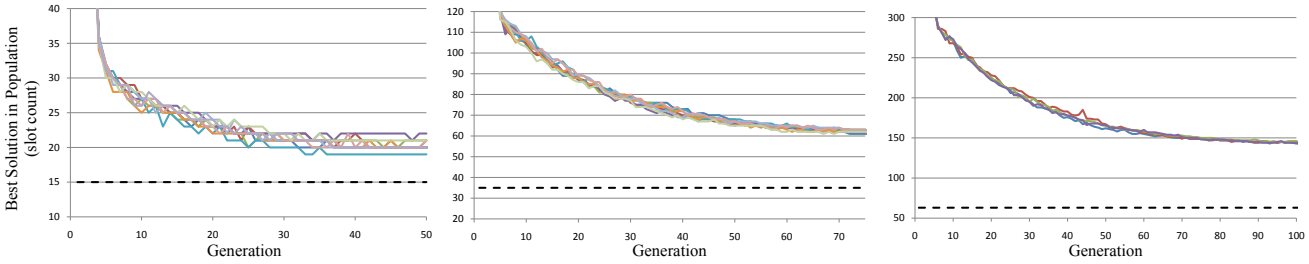
c) Reproduction: After selection, two solutions are chosen at random to mate, producing one child. This is done according to the pseudo-code in Algorithm 1. To form a child, two parents cycle through every possible communication, taking the larger time slot that contains each. The child's *Add* function filters out redundant communications. In this way, the reproduction process favors more dense time slots. This process continues until the population count is back to P.

d) Mutation: After the next generation is formed, each solution in the population has a chance p_m to undergo the mutation process. Mutation first "unrolls" a random time slot in the solution, forming a new time slot for each communication contained in it. Then, the solution is iterated over attempting to combine every time slot combination, thereby spreading the unrolled communication back among the other time slots.

We test our algorithm by running it multiple times for a mesh topology. Table I shows the parameters to the algorithm. Table II summarizes the results for N=16, 36, and 64. Initial experimentation indicated how many generations were necessary before solutions converged for each network size.

Algorithm 1 Reproduction Pseudo-code

```
Solution child = new Solution();
for i = 1 to N do
  for j = 1 to N do
    if i ≠ j then
      Communication c = new Communication(i,j)
      if !child.contains(c) then
        TS1 = time slot containing c from parent1
        TS2 = time slot containing c from parent2
        if TS1.Count > TS2.Count then
          child.Add(TS1)
        else
          child.Add(TS2)
        end if
      end if
    end if
  end for
end for
```



(a) N=16 (b) N=36 (c) N=64

Fig. 2: **Multiple runs of Genetic Algorithm on different network sizes for Mesh topology.**

TABLE I: Genetic Algorithm Parameters

Parameter	**Description**	**Value**
P	Population size	50
k_{TS}	Tournament size	5
p_{TS}	Selection probability	0.8
p_S	Selection size	0.6
p_m	Mutation probability	0.8

TABLE II: Genetic Algorithm Results

	N=16	N=36	N=64
Generations	50	75	100
Best Solution	18	61	142
Execution time (s)	6	231	2855
% Selection	0.19	0.006	0.0005
% Reproduction	47.2	50.9	51.5
% Mutation	50.8	48.2	47.9

Execution time and the percentage of time spent in each stage is reported, indicating that the computational complexity of Reproduction and Mutation are dependent on network size because they involve multiple iterations over all time slots.

Figure 2 shows the evolution along generations from 10 runs (each differently colored) using the parameters in Table I. Solutions begin at $N \times (N-1) - 1$, derivatives of the naive solution. The black dotted lines indicate the number of slots required for a network without topological constraints (non-blocking), the minimum possible (N - 1) under this architecture. Solutions from other runs with different parameters (varying P, k_{TS}) typically came within 2 slots of the best solution found, indicating that the algorithm is relatively insensitive to its parameters. The next section discusses the implementation of the photonic mesh for this network architecture.

V. Network Implementation

Figure 3 shows a 64-core example of a CMP using a 4x4 instantiation of our photonic TDM network, consisting of three basic components: a photonic switch, switch controller, and network gateway. Switches are arranged in a mesh, each controlled by their controller. Each gateway connects four cores, known as gateway *concentration*. In addition, our network design has an added advantage that it is tiled, aligning with today's chip design flow and manufacturing techniques.

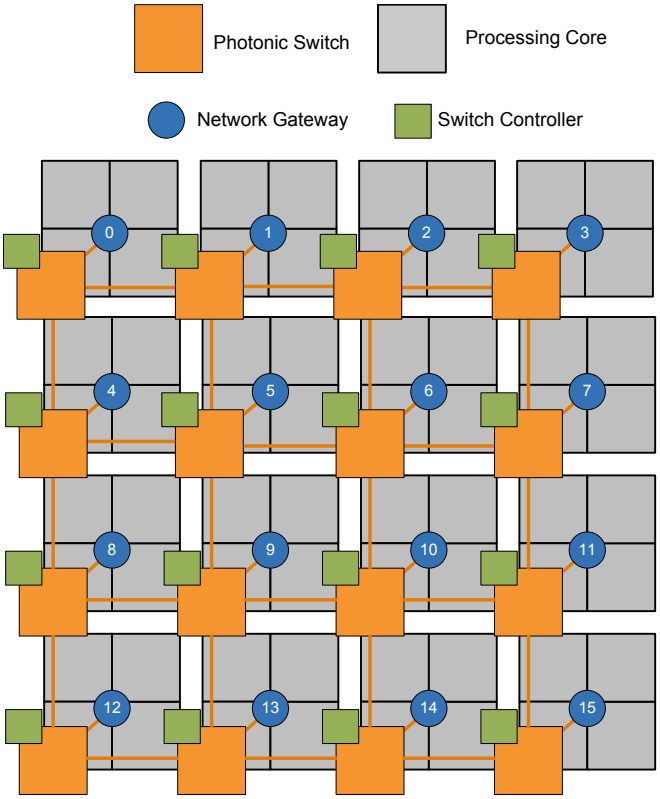


Fig. 3: **Network Architecture.**

A. Photonic Switch

Figure 4 shows the layout for the photonic switch in the 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 III shows the port combinations, and the PSE number that implements the path, referring to Figure 4. For example, we can see in Figure 4 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

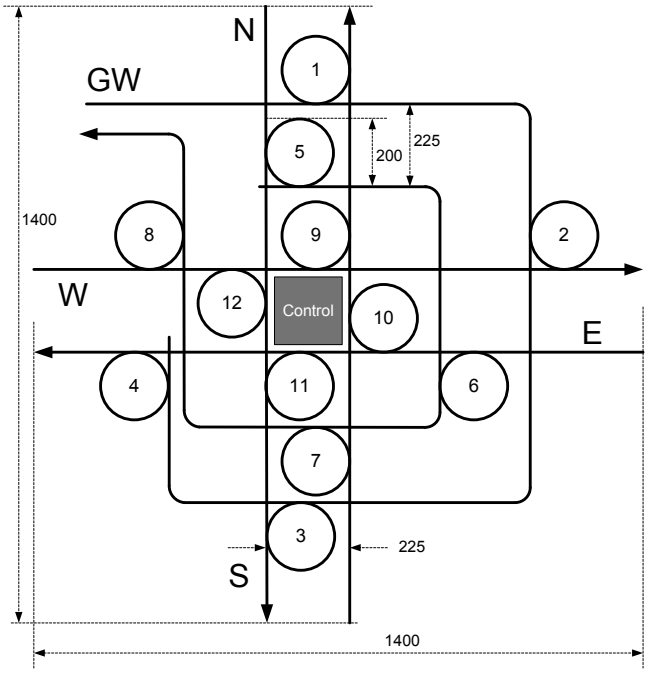


Fig. 4: **Layout of photonic switch, showing waveguides and ring resonators. Units in microns.**

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.

B. 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 of how the switch should be set. A global, synchronous TDM clock can be implemented with waterfall clock distribution, synchronous latency-insensitive design [27], or optical clock distribution [28]. The period of this clock must be the TDM period, t_{slot}. As indicated later in Section VI, 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.

TABLE III: 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	-

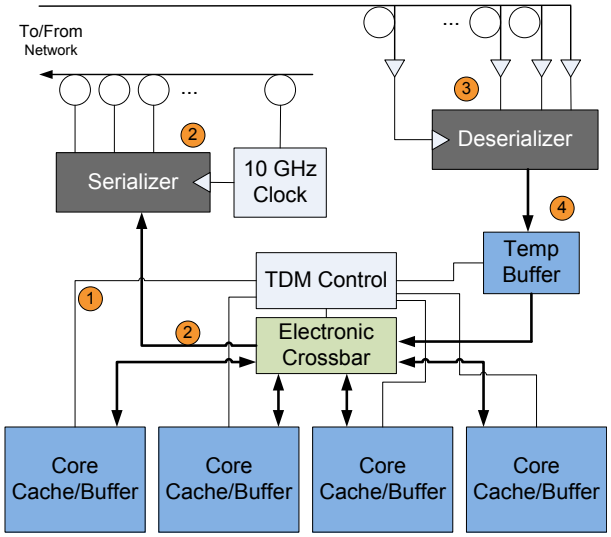


Fig. 5: **Network gateway microarchitecture.**

C. Network Gateway

Figure 5 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 5:

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.

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.

VI. Network Evaluation

We evaluate our design using PhoenixSim [29], a photonic interconnection network simulator. We compare a 64-core instantiation of our architecture (P-TDM) to a similarly-configured packet-switched Electronic Mesh (E-Mesh) and a circuit-switched Photonic Mesh (P-Mesh).

E-Mesh routers have 4 virtual channels (VCs), 1024 bits per VC, and can issue 2 grants per cycle. Links are 128 bits wide, and optimally repeatered according to ORION [30], the electronic router model used in PhoenixSim. All E-Mesh components run at 2.5 GHz. The electronic control network for the P-Mesh has 32-bit links with 1 VC and 128-bit buffers at 1 GHz clock. Both P-Mesh and P-TDM use 128 wavelengths for WDM with 10 Gb/s signaling.

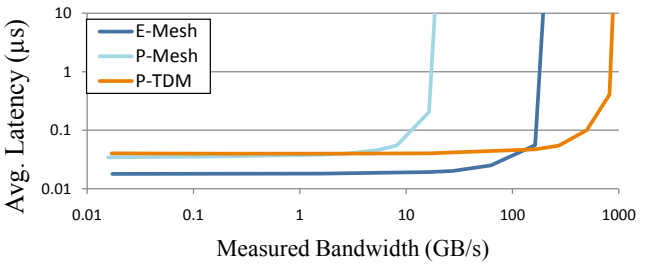
We run uniform random traffic in the network for 10ms with 256B, 8kB, and 256kB messages, and varying the arrival rate until saturation. Values for t_{slot} were set at 4, 13, and 30ns, respectively, assuming the network can be configured ahead of time for traffic with fixed-length messages. Section VII investigates applications with variable message sizes, fixing t_{slot}.

Figure 6 shows the latency vs. total bandwidth in the network for the different message sizes (S). We see that PhotonicTDM achieves superior bandwidth in all cases, between 2-4× over the next best network. This is partly due to the fact that the PhotonicTDM network gateway allows requests to be handled out of order, essentially bypassing head-of-line blocking which is enabled by the architecture. However, the P-TDM network also exhibits slightly higher latency under light loads, when requests must wait for their time slot to transmit.

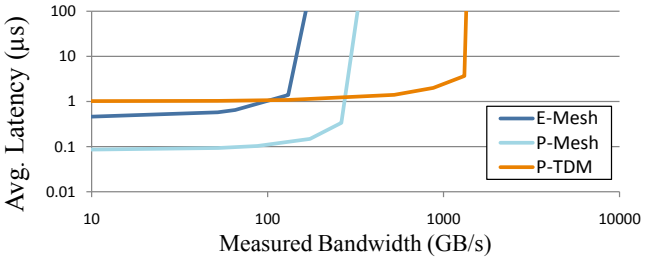
To gauge the power consumption of the networks, we can look at energy per bit, shown in Table IV. Our design achieves superior energy per bit in all cases, except very large messages where the P-Mesh is roughly equivalent.

VII. Case Study: Scientific Applications

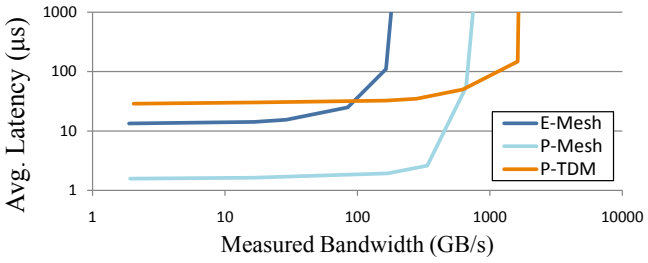
In addition to uniform random traffic, we evaluate our network design in PhoenixSim using scientific application



(a) S=256B



(b) S=8kB



(c) S=256kB

Fig. 6: **Latency vs. measured bandwidth for random traffic with different message sizes.**

TABLE IV: Energy Consumption at Saturation (pJ/bit)

Network	Message Size		
	256B	**8k**	**256k**
E-Mesh	8.43	9.11	9.24
P-Mesh	5.31	0.39	**0.22**
P-TDM	**0.011**	**0.19**	0.24

traces, also used in [19]. A custom-designed library was implemented to efficiently trace communication behavior of MPI code running on multiprocessor systems. MPI programs are appropriate for evaluating our network design because we envision a data-centric programming model which can take advantage of core-to-core *send* commands

Traces are organized into *phases*, in which all communication must take place in order to advance to the next phase. Synchronization broadcasts to and from a single "master" core enforce the barriers between phases. Communication events occur as fast as the network allows. The TDM slot time (t_{slot}) is set at 6ns, about 5kb of data.

We investigate four SPMD-style applications of interest within the scientific computing community:

- Cactus - astrophysics computation toolkit designed to solve coupled nonlinear hyperbolic and elliptic equations

TABLE V: Benchmark Statistics

Benchmark	Num Phases	Num Messages	Total Size (B)	Avg Msg Size (B)
Cactus	2	285	7.3M	25600
GTC	2	63	8.1M	129796
MADbench	195	15414	86.5M	5613
PARATEC	34	126059	5.4M	43.3

arising from General Relativity [31].

- GTC (Gyrokinetic Toroidal Code) - 3D particle-in-cell application developed to study turbulent transport in magnetic confinement fusion [32].
- PARATEC (PARAllel Total Energy Code) - materials science application using Density Functional Theory (DFT) method [33].
- MADbench - benchmark based on MADspec cosmology code, calculating the maximum likelihood angular power spectrum of the cosmic microwave background [34].

Table V summarizes the characteristics of each application. An interesting feature of this set of applications is their diversity, ranging from a few large messages (cactus, gtc) to many small messages (PARATEC) and somewhere in between with many phases (MADbench).

Trace thread ID's are randomly mapped onto cores in the network, a practice commonly used in the scientific computing community when the best mapping is not known or it is not possible to choose it. Figure 7 shows the average execution time and energy consumption for the application traces for each network across fifty runs, with one standard deviation shown in the error bars.

Figure 7a shows the execution time of each application trace across the three compared networks. We can observe that the photonic networks outperform the E-Mesh for Cactus and GTC by about 3× with little difference between P-Mesh and P-TDM, while all three networks are essentially equivalent for MADbench. The P-Mesh network, however, suffers greatly under PARATEC, which is comprised of a large number of very small messages.

Figure 7b shows the network energy consumption of each application across the three networks. In a trend similar to execution time, the photonic networks use much less energy (over an order of magnutide) than the E-Mesh for Cactus, GTC, and MADbench. For PARATEC, the P-TDM network achieves an order of magnitude lower energy consumption than both the E-Mesh and P-Mesh.

When considering both execution time and energy consumption, the photonic networks equally outperform the electronic baseline for Cactus, GTC, and MADbench, all of which exhibit fairly large message sizes. Though our P-TDM design gives up a very small hit in performance and energy compared to the P-Mesh, this is outweighed by the order of magnitude less execution time *and* energy consumption the P-TDM gains from PARATEC. This directly illustrates the versatility of our network design which provides fair access to network resources compared to the purely circuit-switched P-Mesh, which exhibits path-setup blocking and latency.

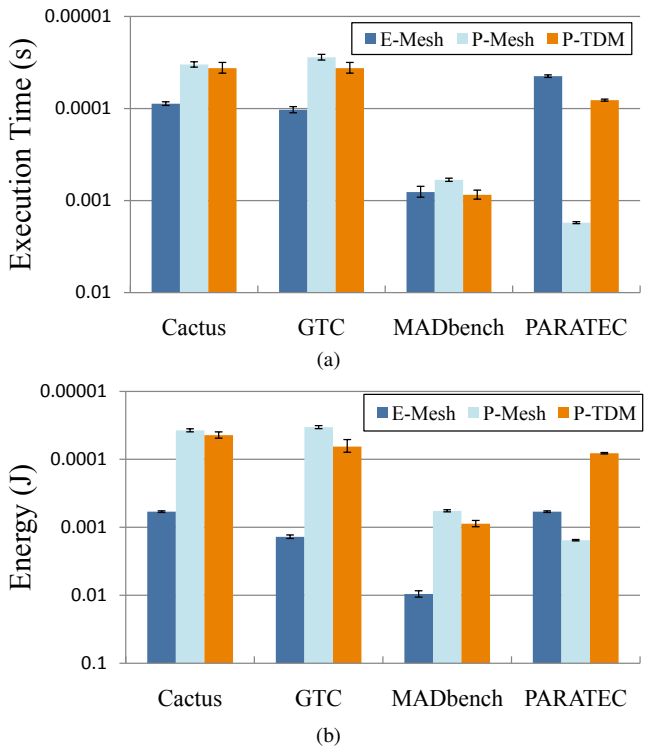


Fig. 7: **Execution time and energy consumption for application traces (values towards top of graph are better).**

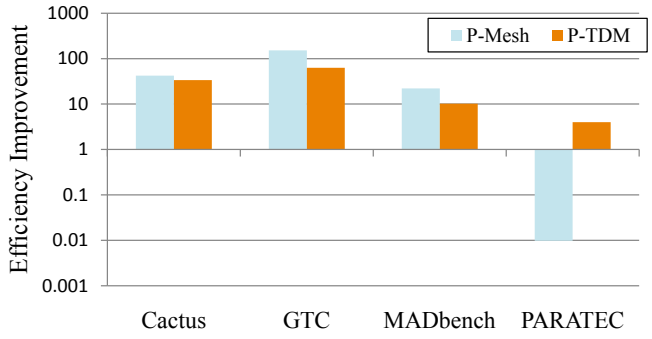


Fig. 8: **Network efficiency improvement over E-Mesh for scientific applications.**

To make this point clearer, we can combine execution time and energy consumption into a single value for *Efficiency*, η, defined as:

$$\eta = \frac{1}{E \times t} \tag{4}$$

where E is the total energy consumption, and t is the execution time of the application. Figure 8 shows the *improvement* over E-Mesh, where a value of 1 indicates even efficiency. We notice the dichotomy of the circuit-switched P-Mesh for changing traffic patterns: many small messages perform significantly worse compared to few, large messages. Our P-TDM design achieves efficiency which is an order of magnitude higher than E-Mesh for Cactus, GTC, and MADbench, while maintaining a 4× improvement for PARATEC.

VIII. Conclusions

We present a novel photonic NoC architecture which introduces the concept of fairness for end-to-end high bandwidth optical circuits by round-robin scheduling all possible communications in both time and space on an arbitrary topology. Studied on a 64-core concentrated mesh, this method proves an effective way to increase network utlization, leading to 2-4$\times$ improved effective throughput under random traffic at network saturation while achieving orders of magnitude better energy per bit compared with the electronic solution. TDM arbitration of photonic circuits enables at least 4$\times$ better efficiency for real scientific application traces over the baseline making it a highly versatile network architecture, compared to a traditional circuit-switched photonic mesh which can be highly sensitive to message size and traffic pattern. Future work will focus on gracefully scaling the network design, using solutions to the scheduling problem already explored in previous work, and incorporating off-chip memory access.

Acknowledgements

This research is partially supported by DARPA MTO under grant ARL-W911NF-08-1-0127, the NSF (Award #: 0811012), and the FCRP Interconnect Focus Center (IFC).

References

[1] N. Magen, A. Kolodny, U. Weiser, and N. Shamir, "Interconnect-power dissipation in a microprocessor," in *SLIP '04: Proceedings of the 2004 international workshop on System level interconnect prediction*. ACM, 2004, pp. 7–13.

[2] L. Chen, K. Preston, S. Manipatruni, and M. Lipson, "Integrated GHz silicon photonic interconnect with micrometer-scale modulators and detectors," *Optics Express*, vol. 17, no. 17, August 2009.

[3] B. Guha, B. B. C. Kyotoku, and M. Lipson, "CMOS-compatibile athermal silicon microring resonators," *Optics Express*, vol. 18, no. 4, Feb. 2010.

[4] H. L. R. Lira, S. Manipatruni, and M. Lipson, "Broadband hitless silicon electro-optic switch for on-chip optical networks," *Optics Express*, vol. 17, no. 25, Dec. 2009.

[5] M. R. Watts, "Ultralow power silicon microdisk modulators and switches," in *5th Annual Conference on Group IV Photonics*, 2008, pp. 4–6.

[6] A. Melloni, F. Morichetti, R. Costa, G. C. an dP. Boffi, and M. Martinelli, "The ring-based optical resonant router," in *IEEE ICC*, 2006, pp. 2799–2804.

[7] D. Vantrease *et al.*, "Corona: System implications of emerging nanophotonic technology," *Computer Architecture, International Symposium on*, vol. 0, pp. 153–164, 2008.

[8] Y. Pan, P. Kumar, J. Kim, G. Memik, Y. Zhang, and A. Choudhary, "Firefly: illuminating future network-on-chip with nanophotonics," *SIGARCH Comput. Archit. News*, vol. 37, no. 3, pp. 429–440, 2009.

[9] N. Kirman *et al.*, "Leveraging optical technology in future bus-based chip multiprocessors," in *MICRO 39: Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture*. Washington, DC, USA: IEEE Computer Society, 2006, pp. 492–503.

[10] C. Batten *et al.*, "Building manycore processor-to-dram networks with monolithic silicon photonics," in *HOTI '08: Proceedings of the 2008 16th IEEE Symposium on High Performance Interconnects*. IEEE Computer Society, 2008, pp. 21–30.

[11] M. J. Cianchetti, J. C. Kerekes, and D. H. Albonesi, "Phastlane: a rapid transit optical routing network," *SIGARCH Comput. Archit. News*, vol. 37, no. 3, pp. 441–450, 2009.

[12] M. Petracca, B. G. Lee, K. Bergman, and L. P. Carloni, "Design exploration of optical interconnection networks for chip multiprocessors," in *HOTI '08: Proceedings of the 2008 16th IEEE Symposium on High Performance Interconnects*. IEEE Computer Society, 2008, pp. 31–40.

[13] A. Joshi *et al.*, "Silicon-photonic clos networks for global on-chip communication," in *NOCS '09: Proceedings of the 2009 3rd ACM/IEEE International Symposium on Networks-on-Chip*. IEEE Computer Society, 2009, pp. 124–133.

[14] Q. Xu, B. Schmidt, J. Shakya, and M. Lipson, "Cascaded silicon micro-ring modulators for wdm optical interconnection," *Optics Express*, vol. 14, no. 20, Oct. 2006.

[15] C. Qiao and R. Melhem, "Reducing communication latency with path multiplexing in optically interconnected multiprocessor systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 8, pp. 97–108, 1997.

[16] X. Yuan, R. Melhem, and R. Gupta, "Distributed path reservation algorithms for multiplexed all-optical interconnection networks," *IEEE Trans. Comput.*, vol. 48, no. 12, pp. 1355–1363, 1999.

[17] C. Qiao and R. Melhem, "Reconfiguration with time division multiplexed mins for multiprocessor communications," *IEEE Transactions on Parallel and Distributed Systems*, vol. 5, pp. 337–352, 1994.

[18] A. Shacham, K. Bergman, and L. P. Carloni, "Photonic networks-on-chip for future generations of chip multiprocessors," *IEEE Transactions on Computers*, vol. 57, no. 9, pp. 1246–1260, 2008.

[19] G. Hendry *et al.*, "Analysis of photonic networks for a chip multiprocessor using scientific applications," in *NOCS '09: Proceedings of the 2009 3rd ACM/IEEE International Symposium on Networks-on-Chip*. IEEE Computer Society, 2009, pp. 104–113.

[20] M. Millberg, E. Nilsson, R. Thid, and A. 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*. IEEE Computer Society, 2004, p. 20890.

[21] K. Goossens, J. Dielissen, and A. Radulescu, "Æthereal network on chip: Concepts, architectures, and implementations," *IEEE Des. Test*, vol. 22, no. 5, pp. 414–421, 2005.

[22] M. Schoeberl, "A time-triggered network-on-chip," in *International Conference on Field-Programmable Logic and its Applications (FPL 2007)*, Amsterdam, Netherlands, August 2007, pp. 377 – 382.

[23] Z. Lu and A. Jantsch, "Tdm virtual-circuit configuration for network-on-chip," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 16, no. 8, pp. 1021–1034, 2008.

[24] C. Paukovits and H. 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*. IEEE Computer Society, 2008, pp. 120–129.

[25] N. Barricelli, "Esempi numerici di processi di evoluzione," *Methodos*, pp. 45–68, 1954.

[26] B. L. Miller and D. E. Goldberg, "Genetic algorithms, tournament selection, and the effects of noise," *Complex Systems*, vol. 9, pp. 193–212, 1995.

[27] A. Edman and C. Svensson, "Timing closure through a globally synchronous, timing partitioned design methodology," in *DAC '04: Proceedings of the 41st annual Design Automation Conference*. ACM, 2004, pp. 71–74.

[28] J.-F. Zheng *et al.*, "On-chip optical clock signal distribution," in *OSA Topical Meeting on Optics in Computing*, 2003.

[29] J. Chan, G. Hendry, A. Biberman, K. Bergman, and L. P. Carloni, "Phoenixsim: A simulator for physical-layer analysis of chip-scale photonic interconnection networks." in *DATE: Design, Automation, and Test in Europe.*, Mar. 2010.

[30] H. Wang *et al.*, "ORION: A power-performance simulator for interconnection networks," in *35th International Symposium on Microarchitecture*, 2002.

[31] "Cactus," 2004, http://www.cactuscode.org.

[32] Z. Lin, S. Ethier, T. Hahm, and W. Tang, "Size scaling of turbulent transport in magnetically confined plasmas," *Physical Review Letters*, vol. 88, 2002.

[33] A. Canning, L. Wang, A. Williamson, and A. Zunger, "Parallel empirical pseudopotential electronic structure calculations for million atom systems," *Journal of Computational Physics*, vol. 160, pp. 29–41, 2000.

[34] J. Borrill *et al.*, "Integrated performance monitoring of a cosmology application on leading HEC platforms," in *International Conference on Parallel Processing (ICPP)*, 2005.