P-sync: A Photonically Enabled Architecture for Efficient Non-local Data Access

David Whelihan*, Jeffrey J. Hughes*, Scott M. Sawyer*, Eric Robinson*, Michael Wolf*,
Sanjeev Mohindra*, Julie Mullen*, Anna Klein*, Michelle Beard*, Nadya T. Bliss*,
Johnnie Chan†, Robert Hendry†, Keren Bergman† and Luca P. Carloni‡
*Massachusetts Institute of Technology Lincoln Laboratory, Lexington, MA
†Department of Electrical Engineering, Columbia University, New York, NY
‡Department of Computer Science, Columbia University, New York, NY

***Abstract*—Communication in multi- and many-core processors has long been a bottleneck to performance due to the high cost of long-distance electrical transmission. This difficulty has been partially remedied by architectural constructs such as caches and novel interconnect topologies, albeit at a steep cost in terms of complexity. Unfortunately, even these measures are rendered ineffective by certain kinds of communication, most notably scatter and gather operations that exhibit highly non-local data access patterns. Much work has gone into examining how the increased bandwidth density afforded by chip-scale silicon photonic interconnect technologies affects computing, but photonics have additional properties that can be leveraged to greatly accelerate performance and energy efficiency under such difficult loads. This paper describes a novel synchronized global photonic bus and system architecture called P-sync that uses photonics' distance independence to greatly improve performance on many important applications previously limited by electronic interconnect. The architecture is evaluated in the context of a non-local yet common application: the distributed Fast Fourier Transform. We show that it is possible to achieve high efficiency by tightly balancing computation and communication latency in P-sync and achieve upwards of a 6x performance increase on gather patterns, even when bandwidth is equalized.**

I. Introduction

As Chip Multi-Processors (CMPs) continue to integrate increasing numbers of cores on a single chip, processor performance and power efficiency become limited by a system's ability to supply data to the cores. Packet-switched grid architectures have emerged as a solution for on-chip networks due to limitations in wire length and fanout imposed by electronic technology [1]. Integrated silicon nano-photonics have been considered for their bandwidth and energy efficiency [2], [3]. In addition to these benefits, the distance independence and relative fanout insensitivity of chip-scale photonics allow for scalable multi-point shared busses and tight global synchronization. The value of these chip-scale photonic innovations can be observed by measuring their impact on real-world applications.

Scatter/Gather communication patterns that distribute information to many receivers, or collect them at a single receiver occur in a wide variety of applications and are difficult to implement efficiently on current CMPs. An example of a scatter is the distribution of program and data from a shared memory interface to numerous processing elements on one or more chips prior to program execution, whereas storing results in the same shared memory at the end of execution represents a gather operation. The salient property possessed by both of these patterns is unsynchronized distributed non-local data access, wherein spatially or logically (address-wise) separate data must be efficiently co-located or re-distributed.

Both of these patterns are seen in applications that process multi-dimensional data structures mapped into a one-dimensional address space. An example is accessing both rows and columns of a memory-mapped matrix, which is an application bottleneck in fields such as astronomy, medical imaging, and intelligence, surveillance, and reconnaissance (ISR). The distributed access of data which is logically separate creates great inefficiencies within the memory hierarchy, especially when the access granularity is small. This inefficiency is oftentimes the central concern of a developer for applications targeting a CMP platform. While many solutions exist to deal with this problem, most require the addition of significant specialized hardware, such as in [4]. Unfortunately, this additional hardware not only costs silicon area, but also power. The latter is problematic in embedded and mobile systems and increasingly in large data centers.

We take a holistic approach to architecture design that reconsiders application needs in the context of the intrinsic properties of photonics. This paper introduces two new modes of operation on a photonic waveguide to mitigate the problem of unsynchronized distributed non-local data access: the Synchronous Coalesced Accesses (SCA) and the Inverse Synchronous Coalesced Accesses (SCA^{-1}). Both the SCA, or scatter operation, and the SCA^{-1}, or gather operation, are enabled by the novel Photonic Synchronous Coalesced Access Network (PSCAN). The PSCAN, implemented on a shared photonic bus topology, was developed to greatly accelerate non-local accesses by reorganizing data in-flight on a photonic waveguide.

Further, the novel P-sync architecture, which utilizes the PSCAN to provide high performance in real applications is proposed. P-sync is evaluated analytically and experimentally

*This work is sponsored by Defense Advanced Research Projects Agency (DARPA) under Air Force contract FA8721-05-C-0002. Opinions, interpretations, conclusions and recommendations are those of the author and are not necessarily endorsed by the United States Government. Distribution Statement A. Approved for public release; distribution is unlimited.

1530-2075/13 $26.00 © 2013 IEEE
DOI 10.1109/IPDPS.2013.56

in the context of a difficult yet common application kernel: the distributed Fast-Fourier Transform (FFT). The FFT presents many challenges which make it difficult to efficiently implement on general-purpose multi-core computing architectures: large multi-dimensional data distribution/retrieval, synchronization of compute resources, and non-local data dependencies.

This paper makes the following contributions:

- The development of a novel photonic communication mode that excels at non-local data accesses, and a network to support it called PSCAN
- The creation of a new computer architecture called P-sync that utilizes PSCAN to achieve high efficiency by tightly interleaving data delivery and computation
- A generalized quantitative analysis of the P-sync architecture performance vs. a wormhole routed electrical mesh
- Experimental results showing P-sync's efficiency on a common, but highly non-local application kernel: the parallel FFT.

The paper is organized as follows: Section II surveys previous work in state-of-the-art chip-scale interconnection networks and introduces the FFT as a highly non-local application kernel. Section III introduces the PSCAN and its transmission mode: Synchronous Coalesced Access. Section IV describes the P-sync computer architecture based upon the PSCAN. Section V is an analytical explanation of the complexities of the FFT and a performance comparison of a mesh-style CMP and a P-sync CMP. Section VI presents high-level application simulations of the full FFT on both of these architectures. Sections VII and VIII conclude with a discussion followed by future work.

II. Background

Modern general purpose processors are highly optimized to deal with locality in one-dimensional data because data is physically mapped to a linear address space. Unfortunately, they perform comparatively poorly when dealing with locality over multiple dimensions, which results in a strided access pattern. A good example of this problem is accessing first the rows, then the columns, of a matrix. In this case, it is easy for a processor to access successive values on the major dimension, stored linearly in memory, but is much more difficult to access successive values stored along the non-major dimension. Common solutions to avoiding non-locality on modern processors, such as caching, do little to help this strided access [5], as caches also have a preferred access dimension and limited size. This problem is important enough that an entire class of architectures, Graphics Processing Units (GPUs) [6], include hardware for optimizing this type of data access. While GPUs are able to more efficiently retrieve data locally across two dimensions, they do this at the expense of silicon area and power.

The Fast Fourier Transform, a highly non-local application kernel, is used in this work to explore the effectiveness of the proposed P-sync architecture. The FFT is a vital kernel in many applications, especially signal processing. While both 1D and 2D FFTs can be found in many applications, large 1D vector FFTs are typically implemented as 2D matrix FFTs to improve overall performance [7]. Therefore, the optimization of the 2D FFT is generalizable to the 1D case. This paper considers specifically the 2D FFT matrix operation as a case study for the challenges of performing a data transpose. This two-dimensional data structure must be accessed in both the row and column dimensions, resulting in highly non-local access patterns.

This non-locality presents design challenges for CMP networks, in which interconnect limitations are a significant driver of architecture. In this work, architectures utilizing electronic interconnect are contrasted with a novel photonically interconnected architecture. As the number of on-chip components increases due to process scaling, electronic communication architectures have moved away from point-to-point globally interconnected networks primarily due to wire delay [8]. Though there has been much research into electronic architectures exhibiting constant delay via delay-optimized repeater insertion [9], this comes at the cost of increased power and area. Instead, the trend has been for on-chip communication to minimize or even abandon global wires in favor of network-on-chip architectures. Although there is no clear consensus on the most appropriate on-chip network topology, mesh networks are commonly employed due to their short but regular interconnect allowing ease of design and routing [1], [10], [11].

Recent advances in silicon photonics, compatible with CMOS [12]–[14], and 3D integration [15] provide for the opportunity to replace global on-chip electrical interconnects with a medium providing greater ability to scale while providing lower-power high-bandwidth communications. Architectures based on fully connected [16]–[18] photonic networks and those based on scalable electronic equivalents [19]–[21] rely heavily on non-optimal arbitration policies or additional electronic communication networks and buffers to be practical at optimizing non-local data access. This is primarily due to the fact that photonic networks are incapable of buffering in-network messages without incurring an unreasonable optical-electrical-optical conversion penalty.

In the following sections, a new communication mode and encompassing architecture will be presented that overcomes these limitations to achieve high performance on non-local access patterns by exploiting the properties of silicon photonics.

III. Photonic Synchronous Coalesced Access Network (PSCAN)

The patterns described thus far require rapid co-location of spatially separate data. Wire delay makes this difficult in electronically connected systems constraining communication to a decentralized hop-by-hop model in which it is difficult to efficiently schedule data delivery over a scalable array of processing elements. If wire delay were not an issue one might build an electronic bus circuit like that shown in Figure 1 to shuffle spatially separate data efficiently to a single data receiver. In that circuit, four frequency-locked clocks with

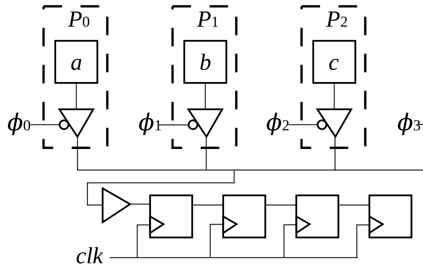


Fig. 1. Electronic time domain multiplexing bus with four processors, P_0–P_3, clocked by ϕ_0–ϕ_3 and containing data bits a–d, respectively. Processor outputs feed an electronic shift register with clk frequency locked to ϕ_i.

phase offsets ϕ_0–ϕ_3 are used to drive a shared bus which terminates at a destination node, represented as a buffer and simple shift register. Assuming that each processing element P_0–P_3 has one bit of data that must be aggregated, the utilization of the bus will be 100% over the duration of the four-cycle transaction. However, two problems prevent this circuit from scaling in size and bandwidth. First, the differently phased clocks require low-skew distribution or generation, which is very difficult over large spatial distances. Second, at high clock rates, the bus will not scale effectively beyond tens of nodes because timing in that bus would be highly variable depending on the location of the driving node relative to the terminus. These limitations on bus scaling are fundamental to traditional electronic interconnect technology.

In contrast, chip-scale photonic waveguides are transmission lines. A simple photonic link is comprised of a laser acting as a source of light, a modulation device, a transmission medium such as a silicon waveguide, and a detector such as a photodiode. The modulator interrupts the incident continuous-wave light from the laser to drive data onto the waveguide. The light travels along the waveguide until it is detected by the photodiode. The speed with which the signal travels is determined by the effective index of refraction of the guiding medium and is independent of the length of the waveguide. The only significant length-dependent parameter is the waveguide loss L_w, which determines the maximum waveguide length before the modulated signal is attenuated below the photodiode's detection threshold. Another important property of photonics is that the directionality of an incident signal is usually preserved at the output of most devices (e.g. modulators).

The photonic equivalent of the circuit described in Figure 1 would be modeled as shown in Figure 2. In the photonic bus, incident energy is modulated by nodes P_0–P_3 as it passes by and is detected at the receiver, where the photodiode converts it to an electrical signal. This particular arrangement scales until the combined attenuation of the waveguide and modulators decreases the signal strength below the detection threshold.

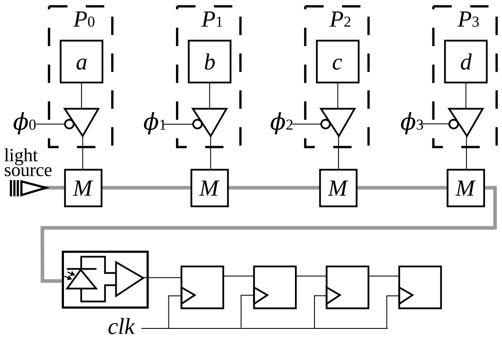


Fig. 2. Photonic time domain multiplexing bus in which each processor P_i has a modulator M sending data to the shift register via a photodiode receiver circuit.

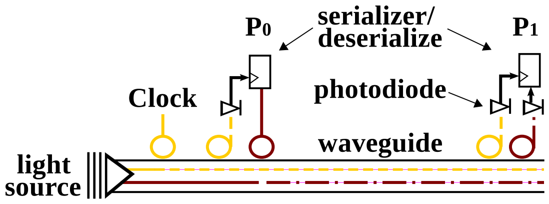


Fig. 3. Transmission and reception of a photonic message (red) between two processors P_0 and P_n synchronized by a global photonic clock (yellow).

Synchronization in this model requires the global coordination of each processor to avoid collisions when writing to the optical bus. This is challenging as the bus requires distribution of a global clock to a large number of processors with minimal skew. When discussing synchronization in this model, it is also important to consider signal flight time. Light with a wavelength of 1550 nm (common in photonic technology) will travel approximately 7 cm/ns in a silicon waveguide. Therefore, even with perfect global clocking, network throughput could be reduced because a processor must wait to use the waveguide until all previously transmitted data bits reach their destination.

Fortunately the directivity of photonics can be used to mitigate this problem. Consider the scheme shown in Figure 3, in which a clock signal is transmitted down a waveguide and detected at intervals by the processors. Due to flight-time delays, each processor's local frame of reference is unique such that a particular clock cycle will be detected at different times by each processor. Each processor can have a Communication Program (CP) that assigns a disjoint set of clock cycles to each processor.

A second wavelength is then modulated by a Serializer/Deserializer (SerDes) that is clocked by the received photonic clock signal qualified by the local CP across all receivers in PSCAN. There is a common skew between the reception of the clock and the modulation of the data wavelength. The modulated data wavelength is therefore synchronized with the clock at each processing element, albeit with constant skew. The CPs comprise non-overlapping portions of a global schedule that is relative to the waveguide clock. Therefore, the program specifies when the waveguide is available for any one processor to modulate light.

In Figure 4, two processors, P_0 and P_1, are interleaving data bits held in their local memory on the waveguide such

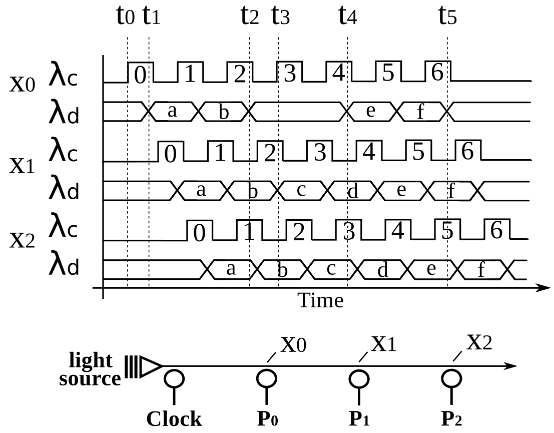


Fig. 4. SCA operation: Each processor location P_0–P_2 contains a detector and modulator. The timing diagram represents the clock and data wavelengths, λ_c and λ_d respectivly, traveling past the waveguide locations x_0–x_2.

that a detector at P_2 sees a continuous stream of spliced data arriving at the maximum data rate supported by the clock. The two independent wavelengths are λ_c, which carries a modulated clock, and λ_d, which is modulated by P_0 and P_1 and detected as data at P_2. The waveforms at three locations on the waveguide are shown as x_0, x_1 and x_2.

At time t_0, a clock edge is detected by P_0 on wavelength λ_c. After a short delay for P_0 to sense and respond to the clock, at t_1 the modulator transmits the SerDes output on wavelength λ_d, completing its transmission of two bits at time t_2. At this point, P_0's CP dictates that it allow unmodulated energy to pass by for use by downstream processors. At time t_3, P_1 begins modulating that energy in response to the received clock edge "2". At time t_4, P_0 begins modulating λ_d even though, based on absolute time, P_1 is still modulating it. This simultaneous modulation is possible because of the non-negligible delay along the waveguide between P_0 and P_1. After driving data for two clock cycles, P_0's CP dictates that it let all subsequent energy pass by. The energy P_1 passes is the energy that P_0 modulated at t_4. The entire transaction is complete from the perspective of P_0 and P_1 at time t_5. From the perspective of P_2 (at physical location x_2), a single six-cycle burst transaction was received, as if from a single source.

This in-flight synthesized transaction is called a Synchronous Coalesced Access (SCA). The inverse process (SCA^{-1}) is also possible, in which a single source sends a large burst of data that is synchronously distributed to a number of receivers. The SCA takes data from a number of spatially separate transmitters, and synthesizes it into a monolithic transaction. The SCA^{-1} is a scatter operation in which one transmitter sends a monolithic transaction that is broken apart on the fly to deliver pieces of data to a number of receivers. The network that supports the photonic SCA and SCA^{-1} is called the Photonic Synchronous Coalesced Access Network (PSCAN).

A. *Synchronization*

In order for the SCA and SCA^{-1} operations to be possible in the PSCAN, a synchronization mechanism needs to be implemented to ensure exact timing of data injection and extraction. Traditional global synchronization strives to minimize timing skew by providing a constant-phase signal throughout a chip. Standard methods use H-tree topologies, which ensure a uniform transmission line length and number of repeater traversals for all endpoints. In contrast, the optical propagation delay of the PSCAN requires an exact amount of phase skew between nodes to account for timing offsets induced by the bus topology. In fact, constant phase in PSCAN would result in data overlap or wait times, lowering network utilization. The full utilization achieved during the SCA and SCA^{-1} operations requires exact temporal alignment of data elements. This form of packet construction has been previously demonstrated [22].

The network utilizes open-loop distribution of the clock. Open-loop distribution forgoes usage of a Phase-Locked Loop (PLL) or Delay-Lock Loop (DLL) and the clock edge used by the input/output memory elements is taken directly from the distributed photonic clock. Traditional distribution networks implement circuits and designs to minimize phase skew, in contrast to the circuit in Figure 3, which requires it. In practice, the clock signal is either propagated along the same waveguide as the data transmission as shown in Figure 3 or along a separate parallel waveguide that is path-length matched to the PSCAN data waveguide. Comparatively, the single-waveguide design requires a more complex filtering and wavelength selectivity scheme, while the parallel waveguide design must deal with ensuring waveguide lengths remain uniform. In either case, the propagation delay provides the exact clock timing that needs to be utilized for data element alignment.

Each network node can utilize a dual-clock FIFO circuit to separate the disparate clock domains of the compute core and the PSCAN. For the SCA operation, the FIFO input would be clocked by the processor core, and the FIFO output would be clocked by the clocking wavelength from the PSCAN. This is reversed for the SCA^{-1}, with the network clock on the FIFO input and core clock on the FIFO output. This separation of clock domains also encapsulates the timing needs of the network from the compute cores.

B. *Scalability*

The PSCAN depends upon a relatively long-distance shared bus to drive re-organized data in bursts. Since physical memory occupies physical space, the longer the bus, the more non-local data that can be re-organized and co-located. The primary limiting factor for PSCAN is loss in the waveguide, since the length of the bus does not affect the speed of the signal.

As light moves through the waveguide, its intensity diminishes due to scattering and other factors. If the intensity of this light drops below a critical threshold, defined by the sensitivity of a receiving photodiode, then the signal is no longer detectable. Thus, the scalability of a single PSCAN segment is defined by the following equation:

$$P_i - L_w \geq P_{\text{min-}p_d} \tag{1}$$

where P_i is the incident optical power at the start of the waveguide, L_w is the loss across the waveguide, and $P_{\text{min-}p_d}$ is the minimum detectable power of the photodiode. Loss in the waveguide itself can be measured in terms of attenuation (in dB) per unit length, though this loss is different for straight and curved paths. Loss also occurs when a waveguide passes a ring resonator that is not tuned to any frequency on the waveguide. This analysis assumes that modulators are evenly spaced along the waveguide. Thus, a *segment* is defined to include a ring resonator and a section of waveguide equivalent in length to the modulator pitch. The loss in a segment is therefore:

$$L_{ws} = L_{r\text{-off}} + D_m L_w \tag{2}$$

where $L_{r\text{-off}}$ is the attenuation due to light passing near a detuned ring resonator, D_m is the inter-resonator pitch, and L_w is the loss in the waveguide. The maximum number of PSCAN segments N, is bound by:

$$\frac{P_i - P_{\text{min-}p_d}}{L_{ws}} \geq N \tag{3}$$

A PSCAN must traverse a chip in a serpentine pattern that includes a number of waveguide curves. The effect of the curves is to slightly decrease N. However, for simplicity they are ignored in this analysis. The primary loss mechanism is attenuation in the waveguide (L_w). The number of possible modulation sites can be quite large, but in practice will depend upon the size of individual processors and layout constraints. Thus, an in depth analysis is not presented here. It is important to note, however, that individual PSCAN segments can be linked via repeaters to form larger networks.

C. Energy

The network energy efficiency of the PSCAN SCA operation was compared to the logically identical operation on a mesh using the PhoenixSim simulator [23]. The electronic mesh and PSCAN simulation models both possess an equivalent 320 Gb/s link to memory. The electronic network is structured as a standard mesh topology with four memory interfaces at the corner network nodes each with a 80 Gb/s link-bandwidth (160 Gb/s for bidirectional.) The electronic routers of the mesh are composed with a 32-bit bus width, 2.5 GHz network clock, and an input buffer size of 480 bits. The PSCAN possesses a single 320 Gb/s link, composed of 32 wavelengths each modulated at 10 Gb/s.

Note the dichotomy of the two networks. The electronic network possesses four separate memory controllers each with one fourth of the total bandwidth of the single memory controller in the PSCAN. This allows electronics to properly leverage the advantage it has in alleviating network-level contention through communication-path diversity. However, it has the disadvantage of large pipeline depths between network nodes. The number of link repeater stages is calculated based on the ORION router model [24], and the router is assumed to have three-stage delay. The chip size was fixed to 2 cm × 2 cm in all simulations. Therefore, the link-repeater stages are inversely related to the number of network nodes. Figure 5 shows the energy-per-bit plots for both the electronic mesh and PSCAN. PSCAN achieves at least a 5.2x improvement for the networks simulated.

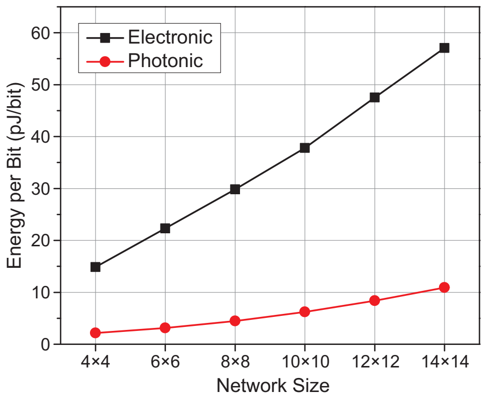


Fig. 5. SCA energy comparison.

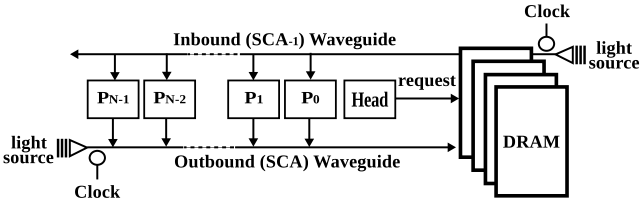


Fig. 6. The P-sync Architecture

IV. P-sync

The P-sync architecture, built around PSCAN, was conceived to perform non-local accesses such as scatter/gather patterns with extreme efficiency. The notion of inter-processor communication (other than with a data-serving head-node) is not supported by the particular implementation of the architecture described in this section. However, this does not mean that the P-sync architecture precludes communication between processors. PSCAN does not necessarily obviate the need for a relatively low-bandwidth photonic or even electrical network to handle such traffic. In fact, the PSCAN physical layer was deliberately designed to be generic, such that it could be shared with other traffic besides SCA and SCA^{-1} transactions. Thus, it is important to understand that PSCAN presents a communication mode on a multi-purpose physical channel, and P-sync, as presented here, is an architecture that is optimized for that mode.

To support the PSCAN communication modes, P-sync is based on a bus topology, with all processors sharing the same photonic waveguide as shown in Figure 6. The following are the important elements of the system:

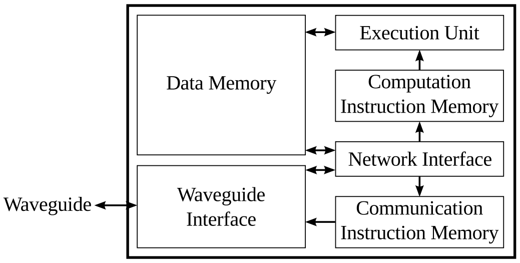


Fig. 7. The processor architecture of a P-sync node enables the SCA and SCA^{-1} communication modes.

- Each *processor* understands its position on the linear busses.
- *Photonic Clock Generators* establish the optical time reference on a waveguide.
- *Storage/DRAM* are a photonically enabled random access memory.
- The *Head Node* is a processor that understands the memory layout (via its own program) and performs requests to the memory such that data is streamed out on the SCA^{-1} waveguide.

A processing element of a machine that could realize this data pattern is shown in Figure 7. The computation core in that processor consists of a local Data Memory, an Execution Unit, and a Computation Instruction Memory. The Execution Unit contains all of the arithmetic and logical units needed to support the instruction set. The Computation Instruction Memory and Data Memory are fed via the Network Interface. The Network Interface coordinates distribution of data from the Waveguide Interface to the various memories in the processing element. The Waveguide Interface coordinates in-flight data reorganizations based upon a program stored in the Communication Instruction Memory.

The PSCAN facilitates tight scheduling due to the distance independent nature of photonics and the resulting ease of global synchrony. It is therefore possible to fuse computation with communication to achieve maximum hardware efficiency and performance. This implies that communication must be described at a similar level of detail as computation. However, the tight interaction between computation and communication does not mean that these functions are carried out in the same functional hardware unit. If so, it would be difficult to parallelize these operations to reduce apparent latency. Rather, the hardware units responsible for communication and computation in each processor must operate in tight synchrony.

Modern computer systems are comprised of relatively inflexible hardware that efficiently runs flexible software. The software generally is quite explicit about the computation operations, but the method by which data is stored and retrieved from other processors or memories is implicit, and is usually handled by hardware. In P-sync however, the communication is quite explicit, and is described by a Communication Program (CP). The CP is a simple schedule that is loaded by the hardware unit responsible for communication on the waveguide in every processor. It is derived from the high-level operational code in much the same way that the individual computations required by a multi-processor's processing elements are compiled into a list of instructions based on source code in a higher-level language.

All CPs on a PSCAN are linked together such that adherence to the PSCAN clock results in only one processor driving the bus and one processor reading the bus at a time. The Head Node also has a synchronous CP, though it has a different function: to make requests to memory such that data is available to scatter on the SCA^{-1} waveguide. CPs can be quite small, with the program for FFT being approximately 96-bits. In the P-sync architecture, all data, including communication programs and computation programs can be delivered on the SCA^{-1} PSCAN. CPs are delivered, along with operational code to the processor on SCA^{-1} operations, interleaved with data delivery. CPs form chains in which one CP loads data, and the CP for the SCA waveguide driver, followed by a CP for the next SCA^{-1} operation. This interspersing of different kinds of control and data information can result in high levels of efficiency, as data can be delivered "just-in-time" by the synchronous waveguide.

In the remainder of this paper, the P-sync architecture will be evaluated quantitatively for its effectiveness in handling non-local communication generally, and then specifically for a parallel FFT implementation.

V. Quantitative Analysis

This section begins with generalized equations for the ideal efficiency of computation based on the overlap of data delivery and computation. Next, the relationships derived are used to determine the potential efficiencies of the FFT application kernel on a mesh-interconnected multiprocessor and a PSCAN architecture.

A. Generalized Performance Model

The computation model used in this analysis decomposes a parallel computation into two parts:

1) Time to deliver the data to the processors
2) Time to compute on the data

Two models of data delivery are used in the analysis for comparison. In Model I a processor must receive all of its data prior to starting computation. We assume the processors share a single path to memory and that the memory controller distributes one processor's data at a time sequentially, such that data delivery is serialized. Figure 8 shows the initial distribution of data on a four-processor system using this model.

For Model I, let:

- S_b be number of data elements per delivery cycle delivered to a single processor
- S_s be the size of each data element in bits
- t_d equal the time to deliver data to a single processor
- t_c equal the time to compute on that data

Given the assumption that the data delivery to all the processors is serialized, the total delivery time is Pt_d, where

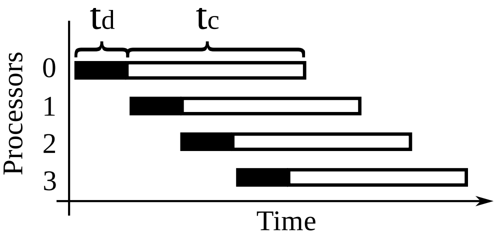


Fig. 8. Model I: Simple data delivery model.

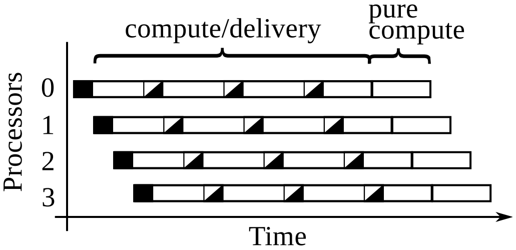


Fig. 9. Model II: Blocked data delivery model.

P is the number of processors. This assumes that data is distributed uniformly to all processors. If PN is the total number of elements of data in the parallel computation, each processor receives N elements each of size S_s bits. Because of serialization, there are two phases of a computation in which the processing nodes are not fully utilized. The first is the time when data is being loaded, called *start-up*, and the second is the time at the end of computation when the computed data is written back to memory, called *wind-down*.

Model II (Figure 9) relaxes the constraint that all data needs to be at a processor prior to computation and provides opportunities for greater efficiency. In algorithms where this is a feasible data delivery strategy, it is possible to deliver the data in k blocks to each processor in turn, repeating the round-robin data delivery until all Pk blocks of data are delivered. The goal of this scheme is to overlap communication and computation to reduce algorithm latency.

For Model II, let:

- S_b and S_s are defined as in Model I
- k be the number of blocks of data delivered to a single processor
- t_{dk} is the time to deliver a single block of data to a single processor
- t_{ck} is the time to compute on that data block

As in Model I, each processor receives N elements where the number of elements in each block of data are $\frac{N}{k}$. Note that Model I is the special case where $k = 1$.

If:

- σ_r is the realized operations / sec
- σ_t is the theoretical peak operations / sec

then, the computational efficiency of the multi-processor is defined as:

$$\eta = \frac{\sigma_r}{\sigma_t} \tag{4}$$

Let:

- ρ be the peak theoretical operations per seconds for each processor
- ϕ be the operations required to be performed on each processor by the algorithm

For distributed processing, maximum theoretical throughput is $P\rho$, but realized operations in both models will be reduced due to data delivery latency.

Consider Model I ($k = 1$) shown in Figure 8:

$$\sigma_r = \frac{P\phi}{Pt_d + t_c} \tag{5}$$

$$\sigma_t = P\rho \tag{6}$$

$$\eta = \frac{\frac{P\phi}{Pt_d+t_c}}{P\rho} = \frac{\phi/\rho}{Pt_d + t_c} = \frac{t_c}{Pt_d + t_c} \tag{7}$$

Relating it to architectural model parameters,

$$t_c = \phi/\rho \tag{8}$$

$$t_d = \lambda + S_b S_s / W_p \tag{9}$$

$$\eta = \frac{\phi/\rho}{P(\lambda + S_b S_s/W_p) + \phi/\rho} \tag{10}$$

where λ is the network latency, and W_p is the network bandwidth.

To determine the efficiency of Model II, we must account for the overlapping computation and communication. Denoting the total time for computation and communication as T,

$$T = Pt_{dk} + (k-1)\max(t_{ck}, Pt_{dk}) + t_{ck} \tag{11}$$

$$\sigma_r = \frac{P\phi}{Pt_{dk} + (k-1)\max(t_{ck}, Pt_{dk}) + t_{ck}} \tag{12}$$

$$\sigma_t = P\rho \tag{13}$$

$$\implies \eta = \frac{t_c}{Pt_{dk} + (k-1)\max(t_{ck}, Pt_{dk}) + t_{ck}} \tag{14}$$

The efficiency for $k > 1$ data delivery can be subdivided into two cases:

Case 1: $Pt_{dk} \leq t_{ck}$:

$$\eta = \frac{t_c}{Pt_{dk} + kt_{ck}} = \frac{t_c}{Pt_{dk} + t_c} \tag{15}$$

Case 2: $Pt_{dk} > t_{ck}$:

$$\eta = \frac{t_c}{Pkt_{dk} + t_{ck}} \tag{16}$$

Based on this analysis, Case 1 is compute bound, while Case 2 is communication bound. Efficiency can only be maximized in Case 1, when computation and communication are balanced, i.e. $Pt_{dk} = t_{ck}$. The previous equations provide insight into the optimal relationship between data delivery and computation in a parallel architecture. It will be shown in the next section that varying latency and network uncertainty makes it difficult to achieve this in a mesh-connected processor. However, using SCA^{-1} in a PSCAN to tightly interleave data delivery to multiple processors, these high levels of compute efficiency are possible.

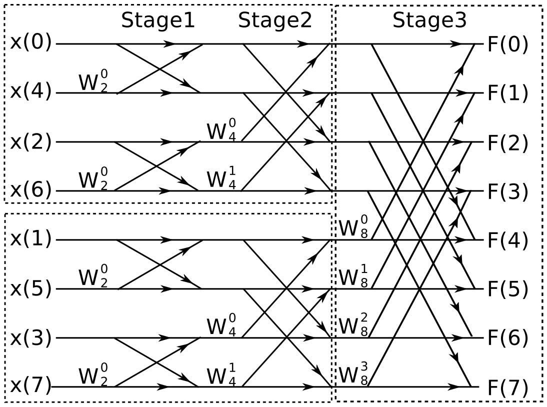


Fig. 10. The FFT can be broken down into smaller, more local computations. Dotted lines are shown for $k = 2$ delivery cycles, each four elements in size.

B. The FFT Application

The data delivery and transpose steps of the FFT are analyzed by computing the ideal efficiency based upon compute and delivery time. Then the effects on efficiency of a wormhole routed mesh are analyzed. The remaining discussion centers on the 2D FFT comprised of the following steps:

- Deliver P blocks of size N samples to the processor array
- Perform P row FFTs in parallel
- Transpose the data into off-chip DRAM
- Load the reorganized data back into the processor array
- Perform P column FFTs in parallel

1) Efficiency in Data Delivery: In many operations, especially the FFT, the complexity of long-distance communication between processors and memory results in the starvation of the compute hardware. In this section, the limits of efficiency, and the parameters that influence it are analytically described for the parallel FFT operation.

This analysis assumes a 1024 × 1024 sample 2D FFT, running on a 256 processor system. Because of memory serialization, the larger the FFT row size, the larger each processor's data delivery phase and the longer the other processors sit idle. However it is possible to use the data delivery mode described in Model II to increase efficiency. In the case of FFT, this is possible because the structure of a Decimation-in-Time FFT results in increasing non-locality as the computation progresses. Therefore, the non-locality as defined by the span in linear memory between two operands increases as 2^n, where n is the number of butterfly stages executed. Thus, small portions of the FFT can be executed on data local to the processor, followed by a pure computation (no data delivery) phase after all sub-blocks are computed, as shown in Figure 10.

The number of multiplication operations per delivery cycle is:

$$\frac{2N}{k}\log_2\frac{N}{k} \tag{17}$$

and the final, compute-only phase requires:

$$2N(\log_2 N - \log_2\frac{N}{k}) = 2N\log_2 k \tag{18}$$

TABLE I
COMPUTE EFFICIENCY FOR ZERO LATENCY

k	S_b	t_{ck} (ns)	t_{cf} (ns)	W_p (Gb/s)	η (%)
1	1024	40960	0	409.6	50.00
2	512	18432	4096	455.1	68.97
4	256	8192	8192	512.0	83.33
8	128	3584	12288	585.1	91.95
16	64	1536	16384	682.7	96.39
32	32	640	20480	819.2	98.46
64	16	256	24576	1024.0	99.38

operations which are computed in time t_{cf}.

As was concluded from Eq. 15 and Eq. 16, the following relationship should be maintained for optimal efficiency:

$$P = \frac{t_{ck}}{t_{dk}}. \tag{19}$$

Table I presents compute efficiency and required bandwidth for the block-based FFT for several block sizes ($S_b = \frac{N}{k}$) Here the peak chip bandwidth (W_p) is computed as follows:

$$W_p = \frac{S_b S_s P}{t_{ck}} \tag{20}$$

Where: S_b is the block size in samples S_s is the number of bits per sample

The numbers in Table I assume the following:

- 1024 point FFTs
- 256 processors
- Floating-point multiplies take 2 ns
- 4 32-bit multiplies per FFT butterfly
- $S_s = 64$
- Only Multiplies are counted

The result of these relationships is that efficiency can be improved by increasing bandwidth, but not for obvious reasons. In this case, the reduction of start-up and wind-down time achieved by decreasing the block size means that the data must be delivered in less time to avoid stalling the processors, which increases the required bandwidth. This occurs because the computational complexity of the FFT is $O(N\log_2(N))$, whereas the delivery time increases as $O(N)$. Thus, block size affects the balance of computation and communication because computation scales non-linearly. Larger block sizes result in a longer compute and delivery phase (related to t_{ck} and t_{dk}), while smaller block sizes result in a longer post-delivery computation phase (t_{cf}).

2) Data Delivery in an Electronic Mesh: For the electronic mesh the following is assumed:

- Square processor array
- Single channels between processors
- Flit Size = FFT element size
- Each processor can hold two flits of data in the case of a blocked channel
- Packets are injected into the network serially from the memory node on the network periphery.

The bandwidth between two neighboring processors is the same as the bandwidth to memory; thus, the bisection network

TABLE II
ELECTRONIC MESH COMPUTE EFFICIENCY WITH LATENCY

k	Delivery Efficiency, η_d (%)	Compute Efficiency, η (%)
1	98.46	49.23
2	96.97	66.88
4	94.12	78.43
8	**88.89**	**81.74**
16	80.00	77.11
32	66.67	65.64
64	50.01	49.70

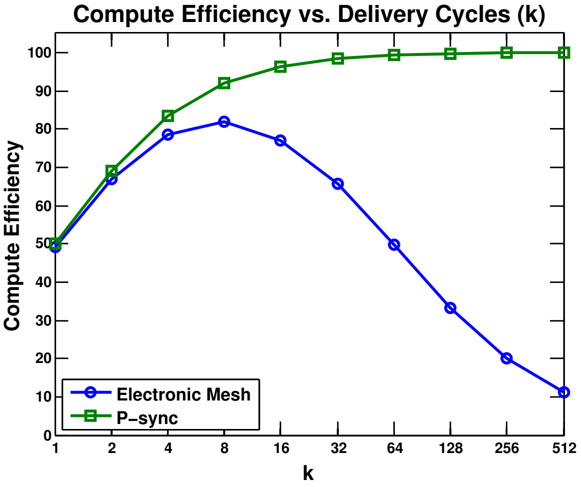


Fig. 11. Global synchrony and pre-scheduled communication allow P-sync to achieve near ideal FFT compute efficiency as k increases. Such efficiency gains in the mesh are limited by the increased overhead of routing smaller packets.

bandwidth far exceeds bandwidth to memory. Let t_r equal the time (in cycles) to route a wormhole header in any processor on the way to a packet's destination. Assuming data to be scattered is always available at the memory controller, and t_r is 0, then the total time to scatter the data is simply PF where P is the number of processors and F is the number of flits delivered to each processor

In real systems it takes at least a cycle ($t_r \geq 1$) for routing logic in each processor to determine the next step in a packet's traversal of the network. The equation for delivery time in cycles is then:

$$PF + P\sqrt{P}t_r \tag{21}$$

When F is large, this routing overhead is small, but when using Method II to increase efficiency the overhead becomes large. In Table II, the $k = 64$ case is half as efficient as the $k = 1$ case.

Comparing the time of delivery assuming zero latency and routing delays to the actual delivery time accounting for these delays results in the following delivery efficiency:

$$\eta_d = \frac{S_b S_c / W_p}{\lambda + S_b S_c / W_p} \tag{22}$$

Table I shows the maximum theoretical efficiency given start-up and wind-down time without network latency. Table II shows the maximum efficiency accounting for the latency of a mesh network. Therefore, the overall efficiency for the mesh will be the product of those efficiencies, since t_d is proportional to the network delivery efficiency. Even under ideal conditions, compute efficiency peaks at 82% when $k = 8$ (shown in Table II in boldface). Achieving this value in the mesh is complicated because network effects make tight synchronization increasingly difficult. Higher efficiency in the electronic mesh is possible only by increasing bandwidth proportional to the inverse of the delivery efficiency.

3) Data Delivery in PSCAN: Using a PSCAN, the P-sync architecture can achieve or come extremely close to these levels of compute efficiency by enabling monolithic transactions from memory that are scattered on-the-fly by receiving processors. The P-sync architecture does this by utilizing the global synchrony permitted by photonic's relative distance independence to tightly interleave the delivery of data from a single source (i.e. memory). The FFT compute efficiency of an electronic mesh-routed architecture and P-sync is compared in Figure 11.

C. Efficiency in Transpose

The distributed transpose operation begins when a number of processors have data to write back to memory, such that elements from one processor will be interleaved in the linear address space of main memory with elements from other processors. This is a gather operation in which the target is the memory.

The following analysis assumes that bandwidth to memory is held constant in both the P-sync (PSCAN) and the electronic mesh used for comparison. Let each system have $P = 1024$ processors and a single memory port. While a single port for 1024 processors may be unrealistic in a general case, the trends shown here apply to systems with more memory ports.

1) PSCAN: The decisions made for data block sizes assume a DRAM system with 2048-bit rows. In such a system, 32 64-bit complex samples can be bursted at a time before a costly row-precharge must occur. Therefore, the assumption is that the traffic to memory at the periphery of the chip should be optimally emitted in 32 × 64-bit blocks.

After each of the first FFTs are executed, each processor writes back one full row, or NS_s bits of data. That data would be divided into P_t transactions:

$$P_t = \frac{NS_sP}{S_r} \tag{23}$$

where N is the row size in FFT samples, S_s is the FFT sample size in bits, P is the number of processors in the array and S_r is the DRAM row size in bits.

Each transaction can be sent to memory in $\frac{S_r}{S_b}$ cycles, where S_b is the bus width. Assuming each transaction needs a S_h bit address header the total transaction time in bus cycles t_t is:

$$t_t = \frac{S_r + S_h}{S_b} \tag{24}$$

Therefore, the transpose can be completed on a PSCAN in $P_t t_t$ bus cycles. Using the following:

TABLE III
TRANSPOSE COMPLETION TIME IN CYCLES

t_p	Writeback Time(cycles)	Multiplier (vs PSCAN)
1	3526620	3.26
4	6553448	6.06

- $N = 1024$ samples
- $S_s = 64$ bits
- $P = 1024$ processors
- $S_r = 2048$ bits
- $S_b = 64$ bits
- $S_h = 64$ bits

The 2^{20} sample transpose writeback can be optimally completed in 1,081,344 bus cycles.

2) Wormhole Mesh: Because of the diversity in architectural options in modern many-core inter-communication networks, designing a one-size-fits-all model is an intractable task. Therefore, in this analysis, a simple wormhole routed mesh model written in SystemC/TLM with the following parameters is used to illustrate the limits of such networks under the transpose load:

- $N = 1024$ processors
- Minimal adaptive wormhole routed
- 1-cycle delay to route a packet header in each encountered router
- 2-flit deep buffers output to inter-processor channels
- 64-bit flits are sent between adjacent processors or memory in 1 cycle

One of the difficulties in transposing in a mesh network is the disorder imposed by the routing hardware on the packets of data. Since the transpose is a data reorganization, the order in which data arrives at the memory interface is very important. Since all of the data is spatially separated, in the simplest case, each element is output independently where it could either be forwarded directly to the main memory as multiple writes (extremely inefficient) or reassembled at the output node using buffers (preferred). The time to transpose a matrix in this environment depends on the ordering of the data, as well as the overhead of sending small packets through the network.

In general, in a fully electrical system, the intercommunication bandwidth available on-chip exceeds the bandwidth off chip. Therefore, there is an unavoidable bottleneck at the memory interface in the transpose, as all processors are attempting to communicate with a single lower bandwidth interface.

Reordering the data requires multiple cycles to account for address decode, transport to staging buffers and time for storage. Further latency is incurred when the data is written to memory. The parameter t_p represents the time spent (in cycles) reorganizing data at the destination node to enable efficient memory writes. For comparison, Table III contrasts the ideal case where $t_p = t_r = 1$ with the case where $t_p = 4$. The rationale for choosing $t_p = 4$ is based on the required reordering steps. The PSCAN performs the operation 3.2x and 6x faster for $t_p = 1$ and $t_p = 4$, respectably. This gain is principally due to network effects such as congestion, header overhead, and the impact of transposing in cache with a processor.

In the next section, a high-level simulation of a full FFT is presented to show how the effects studied here manifest themselves in a full application.

VI. APPLICATION PERFORMANCE RESULTS

For this paper, we have focused on the Fast Fourier Transform (FFT), a kernel in which performance, both in terms of time as well as power, is critical to many real-world applications. In this section, a full FFT flow is simulated at a high level to illustrate the gains possible with P-sync as bandwidth and processing power increases.

A. Simulation Environment

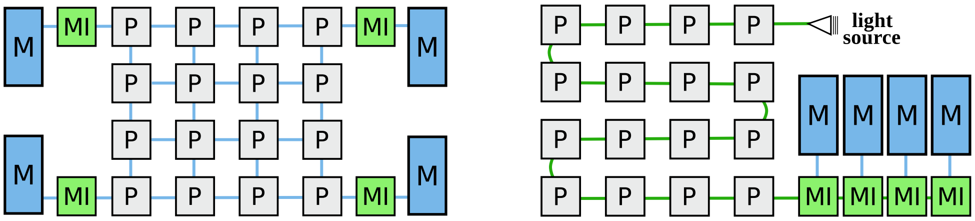
For the purposes of simulating the 2D FFT operation, the Lincoln Laboratory Mapping and Optimization Runtime Environment (LLMORE) is used. LLMORE is a framework for optimizing the mapping of parallel data objects in parallel applications, simulating and optimizing new (and existing) architectures, generating performance data (runtime, power, etc.), and generating/executing optimized code on target architectures. LLMORE can be used to improve the performance of parallel applications and as an important tool for analyzing new hardware architectures. LLMORE takes as input user code, a model of the system architecture, and a set of LLMORE specific parameters that guide its execution. LLMORE's output consists of one or more of the following five items: a complete set of optimized maps (describing the data distribution for all parallel objects in the user code), performance data, a set of optimized architectures for the user code, optimized generated code, and results from a run on target architectures.

LLMORE is used to generate performance data of the 2D FFT operation on the two architectures shown in Figure 12: an electronic mesh and P-sync architecture (using PSCAN). Both architectures assume fast local memory and four shared external memory banks located in the corners for the electronic mesh and at the end of the waveguide for P-scan. When scaling the number of cores the architectures assume a square topology (e.g., a 2x2 mesh). The link bandwidths and latencies are equivalent across architectures in order to achieve a conservative, fair comparison. This results in a bisection bandwidth for the electronic mesh architecture which is significantly larger than that of the P-sync.

In this analysis, both the electronic and P-sync architectures execute two parallel FFT phases interspersed with a transpose. The P-sync architecture achieves this with an SCA, while the electronic mesh executes a *block-wise* transpose. The block-wise transpose loads the data in small pieces to local memory to perform the reorganization prior to writeback.

B. LLMORE Simulation Results

The simulated performance of the 2D FFT on the electronic mesh (blue) and P-sync (green) architectures in gigaflops as



(a) Electronic Mesh Architecture

(b) P-sync Photonic Architecture

Fig. 12. Both architectures contain processing (P), memory (M) and memory interface (MI) elements.

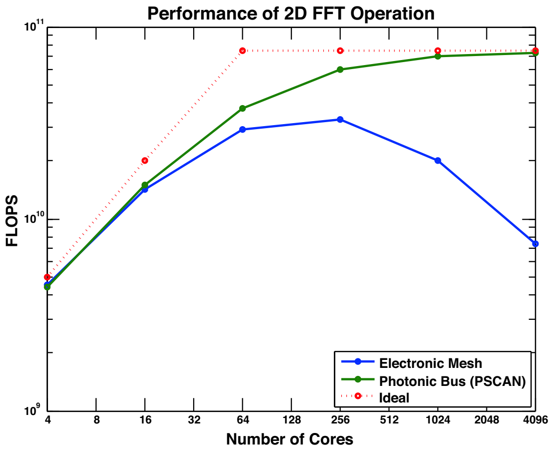


Fig. 13. Simulated performance in floating-point operations per second of the electronic mesh architecture (blue) and P-synch architecture using PSCAN (green) for 2D FFT. Ideal performance shown by red curve.

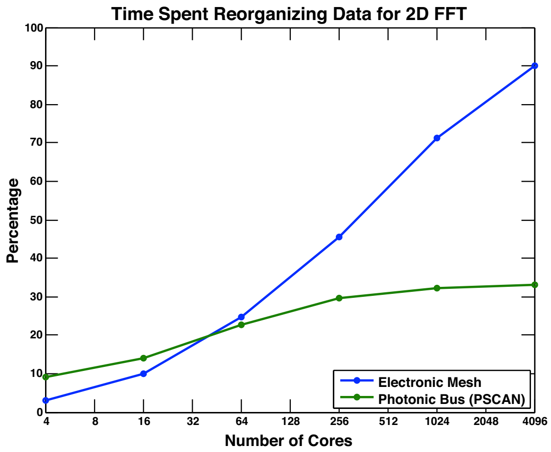


Fig. 14. Percentage of runtime spent in reorganizing data for the 2D FFT on the electronic mesh (blue) and P-sync (green) architectures.

the number of cores increases from 4 to 4096 (mesh dimension from 2 to 64) is shown in Figure 13. It is important to note that the ideal performance for these types of architectures (shown in red) does not scale linearly with the number of cores due to limited memory parallelism (4 memory controllers) and bandwidth. As the number of cores is increased, the performance of the P-sync architecture converges to ideal performance. However, the performance of the electronic mesh architecture peaks around 256 cores and decreases for larger numbers of cores. The performance for the P-sync architecture for $P > 256$ is two to ten times better than the electronic mesh architecture, which is consistent with the previous analysis (Table III). It is also important to note that these simulations use a Model I delivery mode. It is likely that the performance would improve further under P-sync if a Model II delivery mode was used.

Figure 14 shows the percentage of total runtime that each architecture spends in the reorganization of the data between the two 1D FFT operations. This illustrates the difficulty that the electronic mesh architecture has with scaling the 2D FFT up to a large number of cores. The block transpose operation used for the data reorganization on the electronic mesh architecture requires an increasingly larger percentage of the total runtime as the number of cores is increased. However, the percentage of total runtime for the SCA operation used by the P-sync architecture for data reorganization levels off to a significantly more reasonable percentage.

VII. Conclusion

Due to the expense of long-distance on-chip communication, many electrically interconnected CMPs have adopted mesh interconnect that linearizes delay at the expense of additional hardware and increased energy consumption. To mitigate the latency of communication across this hop-based network, architects include hierarchical caches to maximize data locality near processing cores. These hardware resources result in redundant storage of data but ultimately increase aggregate performance. Unfortunately, maintaining cache coherence and re-distributing data during parallel operations presents a challenge to algorithm developers trying to achieve high performance.

In this paper, an architecture called P-sync is described. P-sync utilizes chip-scale photonics in a novel manner, greatly accelerating parallel applications and application kernels that exhibit non-local data accesses, such as scatter/gather patterns. These patterns are common in linear algebra computations in the form of the matrix transpose as well as signal and image processing kernels, such as the Fast Fourier Transform.

It was shown that the very common scatter data pattern can be optimized to maximize the utilization of compute hardware. Achieving this efficiency in a typical electrical mesh-connected network was analytically shown to be extremely difficult due to network effects, routing overhead, and the cost of reordering data at the memory port. In addition, the inverse gather process, in which distributed data must be accumulated and reordered at a single location, was explored and shown to benefit greatly from in-flight data reorganization in the proposed photonic interconnect.

The mechanisms that enable these performance gains are the Synchronous Coalesced Access (SCA) and the network to support it, the Photonic Synchronous Coalesced Access Network (PSCAN). The PSCAN allows burst transactions to a single destination to be synthesized in-flight from multiple spatially separate contributors for reorganization of distributed data. This operation is performed without any special buffering logic and can result in optimal use of channel bandwidth to off-chip storage. Through simulation, it is shown that a large number of cores can reorganize data in-flight at the maximum data rate to memory, achieving upwards of 6x in improved performance.

This exploration is a significant departure from previous thinking about the uses of chip-scale photonics. In this work, features enabled by the unique properties of photonic interconnect—easier global synchrony and distance independence—are explored as application performance enablers, rather than solely bandwidth density and lower power. By first focusing on the application, and then rethinking the interaction of the processors with the network, we have shown that it is possible to achieve large gains in performance and energy efficiency for challenging non-local data access patterns.

VIII. Future Work

The experiments presented in this paper are relatively simple, but they expose the power of the PSCAN interconnect. There is much work to be done to explore the characteristics, scalability, and utility of the SCA. Areas for further exploration include: generation of distributed communication programs from abstract programmer constructs, compatibility with other transfer modes and architectures, and utility in other applications and application domains.

References

[1] D. Wentzlaff, P. Griffin, H. Hoffmann, L. Bao, B. Edwards, C. Ramey, M. Mattina, C.-C. Miao, J. Brown, and A. Agarwal, "On-chip interconnection architecture of the Tile processor," *Micro, IEEE*, vol. 27, no. 5, pp. 15–31, Sept.–Oct. 2007.

[2] G. Hendry, E. Robinson, V. Gleyzer, J. Chan, L. Carloni, N. Bliss, and K. Bergman, "Circuit-switched memory access in photonic interconnection networks for high-performance embedded computing," in *Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 1–12.

[3] G. Hendry, E. Robinson, V. Gleyzer, J. Chan, L. P. Carloni, N. Bliss, and K. Bergman, "Enabling high performance embedded computing through memory access via photonic interconnects," in *High Performance Embedded Computing (HPEC)*, Sep 2010.

[4] B. S. Nordquist and S. D. Lew, "Apparatus, system, and method for coalescing parallel memory requests," U.S. Patent 7,4923,68B1, Feb. 17, 2009.

[5] H. Izumi, K. Sasaki, K. Nakajima, and H. Sato, "An efficient technique for corner-turn in SAR image reconstruction by improving cache access," in *Parallel and Distributed Processing Symposium., Proceedings International, IPDPS 2002, Abstracts and CD-ROM*, 2002, pp. 3–8.

[6] J. Owens, M. Houston, D. Luebke, S. Green, J. Stone, and J. Phillips, "Gpu computing," *Proceedings of the IEEE*, vol. 96, no. 5, pp. 879 –899, may 2008.

[7] D. H. Bailey, "FFTs in external or hierarchical memory," *Journal of Supercomputing*, vol. 4, pp. 23–35, 1990.

[8] W. Dally and B. Towles, "Route packets, not wires: on-chip interconnection networks," in *Design Automation Conference, 2001. Proceedings*, 2001, pp. 684–689.

[9] R. Ho, K. Mai, and M. Horowitz, "The future of wires," *Proceedings of the IEEE*, vol. 89, no. 4, pp. 490–504, Apr 2001.

[10] M. Taylor, J. Kim, J. Miller, D. Wentzlaff, F. Ghodrat, B. Greenwald, H. Hoffman, P. Johnson, J.-W. Lee, W. Lee, A. Ma, A. Saraf, M. Seneski, N. Shnidman, V. Strumpen, M. Frank, S. Amarasinghe, and A. Agarwal, "The Raw microprocessor: a computational fabric for software circuits and general-purpose programs," *Micro, IEEE*, vol. 22, no. 2, pp. 25–35, Mar./Apr. 2002.

[11] S. Vangal, J. Howard, G. Ruhl, S. Dighe, H. Wilson, J. Tschanz, D. Finan, A. Singh, T. Jacob, S. Jain, V. Erraguntla, C. Roberts, Y. Hoskote, N. Borkar, and S. Borkar, "An 80-tile sub-100-W TeraFLOPS processor in 65-nm CMOS," *Solid-State Circuits, IEEE Journal of*, vol. 43, no. 1, pp. 29–41, Jan. 2008.

[12] C. Gunn, "CMOS photonics—SOI learns a new trick," in *SOI Conference, 2005. Proceedings. 2005 IEEE International*, Oct. 2005, pp. 7–13.

[13] M. Salib, L. Liao, M. Morse, A. Liu, and D. Samara-Rubio, "Silicon photonics," *Intel Technology Journal*, vol. 08, May 2004.

[14] M. Lipson, "Guiding, modulating, and emitting light on silicon-challenges and opportunities," *Lightwave Technology, Journal of*, vol. 23, no. 12, pp. 4222–4238, Dec. 2005.

[15] N. Sherwood-Droz and M. Lipson, "Scalable 3D dense integration of photonics on bulk silicon," *Opt. Express*, vol. 19, no. 18, pp. 17 758–17 765, Aug. 2011.

[16] N. Kirman, M. Kirman, R. Dokania, J. Martinez, A. Apsel, M. Watkins, and D. Albonesi, "Leveraging optical technology in future bus-based chip multiprocessors," in *Microarchitecture, 2006. MICRO-39. 39th Annual IEEE/ACM International Symposium on*, Dec. 2006, pp. 492–503.

[17] D. Vantrease, R. Schreiber, M. Monchiero, M. McLaren, N. P. Jouppi, M. Fiorentino, A. Davis, N. Binkert, R. G. Beausoleil, and J. H. Ahn, "Corona: System implications of emerging nanophotonic technology," in *Proceedings of the 35th Annual International Symposium on Computer Architecture*, ser. ISCA, 2008, pp. 153–164.

[18] A. Krishnamoorthy, R. Ho, X. Zheng, H. Schwetman, J. Lexau, P. Koka, G. Li, I. Shubin, and J. Cunningham, "Computer systems based on silicon photonic interconnects," *Proceedings of the IEEE*, vol. 97, no. 7, pp. 1337–1361, July 2009.

[19] A. Shacham, B. Lee, A. Biberman, K. Bergman, and L. Carloni, "Photonic noc for dma communications in chip multiprocessors," in *High-Performance Interconnects, 2007. HOTI 2007. 15th Annual IEEE Symposium on*, aug. 2007, pp. 29 –38.

[20] A. Joshi, C. Batten, Y.-J. Kwon, S. Beamer, I. Shamim, K. Asanovic, and V. Stojanovic, "Silicon-photonic Clos networks for global on-chip communication," in *Networks-on-Chip, 2009. NoCS 2009. 3rd ACM/IEEE International Symposium on*, May 2009, pp. 124–133.

[21] G. Hendry, J. Chan, S. Kamil, L. Oliker, J. Shalf, L. Carloni, and K. Bergman, "Silicon nanophotonic network-on-chip using TDM arbitration," in *High Performance Interconnects (HOTI), 2010 IEEE 18th Annual Symposium on*, Aug. 2010, pp. 88–95.

[22] O. Liboiron-Ladouceur, H. Wang, and K. Bergman, "An all-optical PCI-Express network interface for optical packet switched networks," in *OFC/NFOEC 2007*, Mar. 2007, pp. 1–3.

[23] J. Chan, G. Hendry, A. Biberman, K. Bergman, and L. Carloni, "Phoenixsim: A simulator for physical-layer analysis of chip-scale photonic interconnection networks," in *Design, Automation Test in Europe Conference Exhibition (DATE), 2010*, Mar. 2010, pp. 691–696.

[24] H.-S. Wang, X. Zhu, L.-S. Peh, and S. Malik, "Orion: a power-performance simulator for interconnection networks," in *Microarchitecture, 2002. (MICRO-35). Proceedings. 35th Annual IEEE/ACM International Symposium on*, 2002, pp. 294–305.