Review article **Open Access**

Joe Touch*, Yinwen Cao, Morteza Ziyadi, Ahmed Almaiman, Amirhossein Mohajerin-Ariaei and Alan E. Willner

Digital optical processing of optical communications: towards an Optical Turing Machine

DOI 10.1515/nanoph-2016-0145
Received August 15, 2016; revised November 1, 2016; accepted November 5, 2016

Abstract: Optical computing is needed to support Tb/s in-network processing in a way that unifies communication and computation using a single data representation that supports in-transit network packet processing, security, and big data filtering. Support for optical computation of this sort requires leveraging the native properties of optical wave mixing to enable computation and switching for programmability. As a consequence, data must be encoded digitally as phase (M-PSK), semantics-preserving regeneration is the key to high-order computation, and data processing at Tb/s rates requires mixing. Experiments have demonstrated viable approaches to phase squeezing and power restoration. This work led our team to develop the first serial, optical Internet hop-count decrement, and to design and simulate optical circuits for calculating the Internet checksum and multiplexing Internet packets. The current exploration focuses on limited-lookback computational models to reduce the need for permanent storage and hybrid nanophotonic circuits that combine phase-aligned comb sources, non-linear mixing, and switching on the same substrate to avoid the macroscopic effects that hamper benchtop prototypes.

Keywords: digital optics; non-linear processing; optical computing; wave mixing.

***Corresponding author: Joe Touch,** USC/ISI, 4676 Admiralty Way, Marina del Rey, CA 90292, USA, e-mail: touch@isi.edu
Yinwen Cao, Morteza Ziyadi, Ahmed Almaiman, Amirhossein Mohajerin-Ariaei and Alan E. Willner: EE/Systems Dept., USC, Los Angeles, CA 90089, USA

1 Introduction

Optical communication is becoming more prevalent because it supports higher-bandwidth data transmission over longer distances than electronics. Increased bandwidth is not always enough, though – often the data needs to be computationally processed in-transit. This results in the need to support network functions at communication rates, preferably in the optical domain as well. In particular, in-transit computation benefits most from a means to compute and communicate using the same optical data format.

1.1 Use cases of optical processing

It can be very useful to unify communication and computation using a single data encoding to efficiently support both long-distance transmission and in-transit processing [1, 2]. Networks can transfer data opaquely, but increasingly need to support in-network computation involving data meta-information (headers) or the data itself. Common examples include network packet processing, bulk network security, and big data filtering (Figure 1).

Networks transfer information using either circuits or packets; circuits are more efficient when traffic patterns are predictable; however, packets (whether fixed or variable length) are more efficient for unpredictable uses and are thus preferred, where possible [3]. Packet processing can focus on simple switching, address indexing, and label/header rewriting [4, 5], or include more complex hop-count update, managing checksums, and collision resolution [6] – all necessarily occurring inside the network where traffic from different sources combines to share resources. This processing can include encapsulation (and decapsulation) or direct translation, supporting virtual networks, software-defined networking, and network address translations (NAT) [7, 8].

Figure 1: Use cases for digital optical computing.

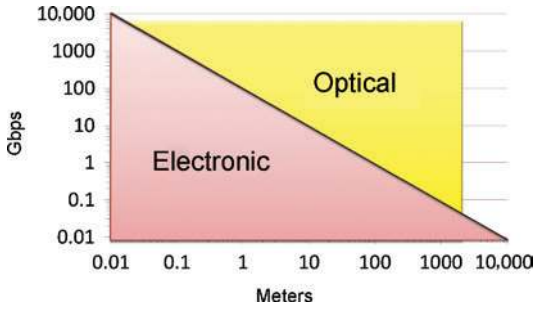

Figure 2: Bandwidth-distance limits of electronic communication.

Data transiting a network is vulnerable to tampering and copying, and thus is often processed to include integrity protection or encryption [7]. Such security is often applied between the communicating endpoints, but additional layers of security can ensure protection between enterprises or when transiting an untrusted area. This additional protection can only be implemented inside the network, at the boundaries between trusted network components. Network security also protects the network itself from both control-plane attacks and data-plane denial-of-service attacks.

Big data filtering involves collecting and exploring large amounts of information, some collected in advance and others collected on-the-fly. It can be impractical to store these data sets for off-line processing; instead, an on-line approach can digest large streams, either completely or as a preprocessing reduction step. In-network processing enables on-line filtering or coalescing of these streams. Note that filtering need not be perfect; a low occurrence of false positives can be tolerated if false negatives can be avoided.

1.2 The need for a new approach

It is currently common to compute using electronics and communicate using optics. Electronic switching can be fast (< 1.7 ps), integrated easily (4 G transistors per device), and support bit- and device-level parallelism (128 bit-wide processors, 3500 graphics cores, 384 bit-wide memory). However, high-bandwidth electronic signals do not propagate well; signals of 10 Gb/s over > 10 m are necessarily optical, as shown in Figure 2 [9].

Optics is more efficient for long-distance, high-bandwidth communication. Unamplified signals can propagate tens of kilometers, each symbol representing several bits, encoding 100 Gb/s on each of dozens of wavelengths. However, it is difficult to integrate the building blocks of optical systems, including laser pumps, comb sources (coherent multiwavelength sources), passive components (e.g. filters, splitters), and non-linear devices (for wave mixing) into a single optical circuit and optical switches are limited to operating in the ms-ns range.

The electronic and optical approaches conflict when considering functions that occur on data in transit. A hybrid approach using electronics for computation and optics for communication requires optical-electrical-optical (OEO) conversion, which is complex, expensive, and can waste energy. A unified approach would avoid these issues, but only if a single encoding were used for both communication and computation. High-bandwidth, long-distance communication is necessarily optical, so the encoding must be optical, too.

Merely replacing electronics with optics remains a significant challenge. Photons do not interact in ordinary environments; they interact only indirectly through matter. These interactions typically require either out-of-band electronic control (e.g. by inducing a voltage or current on the material) or high-power optical signals (e.g. for frequency mixing). As a result, we do not assume that a shift to optical computing will result in reduced energy consumption.

Optical wavelengths are 100× larger than commercial IC processes in each dimension (1500 nm vs. 14 nm), so optics requires 10,000× more area for an equivalent device [10]; this difference grows by another 50 : 1 in each dimension considering the 200 fm size of a single atom (Figure 3) [11]. As a result, each nanoscale optical device occupies the space of up to 25 million transistors.

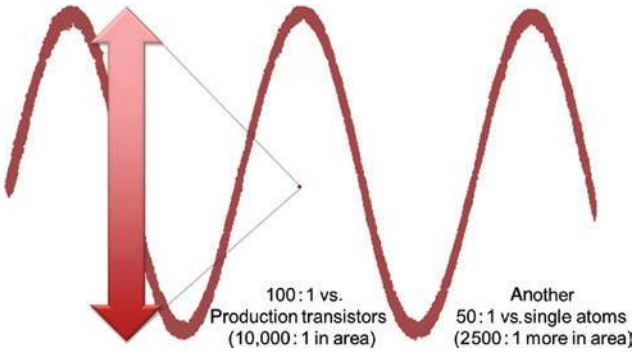

Figure 3: Device scale of optics vs. electronics.

The overall challenge is that electronics and optics have evolved different approaches to improving performance. As shown in Table 1, electronics is typically limited to 1 b/symbol over comparatively short distances and low channel capacity. It achieves high performance using very fast switching, extraordinarily high-density integration, and large parallelism. Optics more easily supports higher-channel capacity over much longer distances, but suffers in size and slow switching speed, offset only by a slightly higher symbol bit density. The table uses values for long-haul optical communication; distances decrease to 10–1000 m and wavelengths to 850–1310 nm for short and medium haul, but do not greatly impact the consequence.

The table also uses values for existing commercial transistors; at the limit of a single atom, feature size decreases to 200 fm (0.2 nm). The consequence of these differences is the need for optical processing that can support multibit encodings without relying on switching in the data plane.

1.3 Digital optical processing

A single multibit optical data encoding should suffice for both long-distance communication and digital computation. Processing needs to leverage the native properties of optical wave mixing to support Tb/s data rates and thus avoid the limitations of slow optical switching. It needs to focus on computation needed to support in-transit data processing, including developing new optics-friendly algorithms that reduce the need for arbitrary persistent state, lookup tables, indefinite recirculation, and other techniques that have no current optical equivalent. We call this approach an "Optical Turing Machine" (OTM) [1, 2].

Table 1: Domain impact on computation and communication.

	Electronics	Optics
Bits/symbol	1	4–8+
Bits/path	64–256+	1
Switching	<1.7 ps	>1 ns
Feature size	<14 nm	>1500 nm
Distance	<1 m	10 km+
Bandwidth	<600 GHz	40 THz+

2 Background

Before considering optical computation, it is important to understand the context of computation more generally. Much current work in optical computation is limited to very simplistic combinatorial functions or finite-state machines, which can severely limit the types of problems that can be supported.

This section addresses these issues, as well as the limitations of some previous approaches and what makes optical processing uniquely challenging. It also addresses our focus on bandwidth rather than energy conservation.

2.1 Principles of computation

Computation is a very well understood and precisely defined concept in computer science. The most basic computational hierarchy is based on the amount of state and a program's ability to access that state (Table 2). All variants assume combinatorial logic based on a mathematical field, i.e. an algebraic structure composed of a set of symbols and a pair of operations that satisfy a set of conditions (Figure 4, leftmost machine). Most electronic computations use Boolean logic, the smallest finite field.

Beyond that field, different levels of computational complexity assume more state and more complex state access, and support increasing levels of capability. Combinatorial logic has no internal state and is limited to generating non-feedback mappings, i.e. tables that map inputs to outputs with no context from previous lookups. A finite-state machine (FSM) includes exactly one state, which provides limited context of previous behavior and can generate so-called regular grammars, simple

Table 2: Hierarchy of computational models.

Machine	Functions	Optical computer
Combinatorial logic	Non-feedback maps	Most signal proc.
Finite-state machine	Regular grammars (cannot count)	Hop-count decrement, neural nets
Pushdown automata	Context-free grammars	Backtracking (AI search)
Linearly bounded TM	Context-sensitive grammars	Bounded feedback functions
TM	Recursive languages ("computable")	Any feedback function, incl. checksum

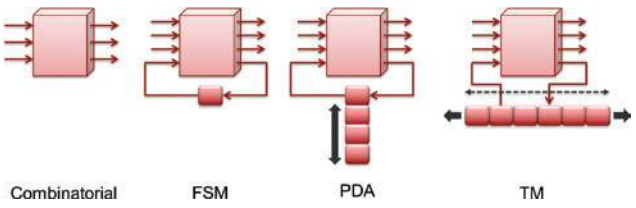

Figure 4: State and various machines.

expressions used in some search systems (Figure 4, second machine). FSMs cannot count but can perform limited arithmetic (e.g. hop-count decrement), and they describe the basic neuron, the building block of neural networks.

The next more complex level of computing is the pushdown automata (PDA), in which memory operates as an infinite stack of values in which one can access only the topmost value (Figure 4, third machine). PDAs support context-free grammars, including conversion from reverse Polish notation (RPN) to infix, and are important in artificial intelligence (AI) backtracking algorithms.

The two most complex levels define what is computable, i.e. the recursively enumerable languages. They are both based on the Turing Machine (TM), a generalized abstract computer that can (by definition) compute anything that is computable, including checksums, encryption algorithms, and big data filtering using arbitrary context (Figure 4, rightmost machine). Like a PDA, a TM has a large number (or infinite) set of values, but a TM can access them in any order.

Both TMs and PDAs are defined as having infinite storage; however, this is often misunderstood as precluding their real existence. The amount of memory required for a TM or PDA is always finite; otherwise, the machine could not stop (or "halt") – a requirement of computation. A linearly bounded TM is a limited form of TM whose state is bounded by a linear function of the size of the input, i.e. it never needs more than $K \times N$, where K is a (potentially large) constant and N is the size of the input.

This hierarchy thus defines what a device can compute, even if that device is replicated in parallel (e.g. a neural computer composed of a network of neurons or a grid of FSMs). Combinatorial logic and FSMs are not computationally capable of identifying strings of A/B symbols in which the number of A's matches the number of B's. PDAs and LB-TMs cannot compute cryptographic hash functions. This is why we call our project the OTM – because in-network processing may need the full power of a TM. This also highlights the computational limitation of approaches based on computational logic, FSMs, or even PDAs.

2.2 The limits of previous approaches

Most previous approaches can be grouped into two categories that we call "Field of Dreams" and the "Aluminum Feather". The former (from the movie of the same name) explores the variety of potential device designs and the latter focuses on developing an "optical transistor".

Field of Dreams explores the design space by building new components in the anticipation that they will become useful to others in the future ("build it and they will come", again from the movie). Many of these approaches directly interfere with computation, component composition, or communication using a single format.

This includes lenses (which perform Fourier transforms), spatial light modulators, electro-optical and mechanical switches, gratings, etc. [12–15]. These approaches compared favorably to older technology, where even a small number of analog processing of combinatorial logic in two dimensions was sufficient to outperform 1980s uniprocessor architectures [16]; however, they do not compete as well with GPUs with thousands of cores. Noise accumulated by cascaded stages or feedback prohibits their use for all but the simplest functions [17].

Aluminum Feather refers to the following approach to airplane design: birds fly using feathers; thus, building an aluminum bird (an airplane) clearly requires aluminum feathers. Transistors support electronic computation, and this has been used as the primary justification that optical computation requires optical transistors [18–21]. However, most optical devices are very different from their electronic counterparts (Figure 5). Optical switching in microelectromechanical systems (MEMS) is very slow (1 kHz) and electro-optical photonic switches are limited by the radio frequency (RF) transmission effects of their high-voltage driving electronics.

Electro-optical switching at 1 GHz is often compared to silicon electronics at 3 GHz; however, such a comparison is unfair because EO devices require exotic materials. A more fair comparison would be to germanium and other exotic electronic devices, which are much faster (600 GHz). Our team's OTM project thus focuses on wave mixing, which processes light via interactions during transmission and is capable of 40 THz.

2.3 Optical computation is difficult

Computation is a complex process by which an output is generated based on a set of inputs and (usually) state. These functions are often cascaded, where the output of one function is the input of the next, and which may also

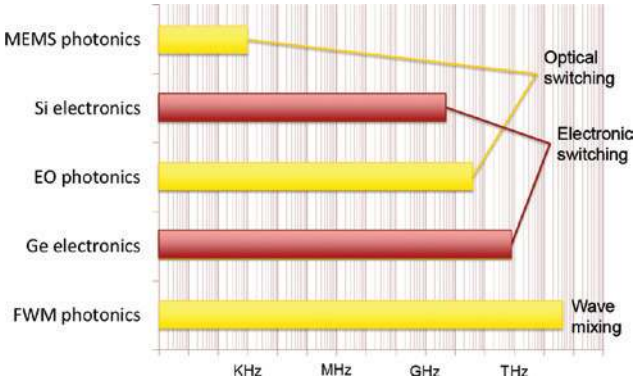

Figure 5: Domain impact on speed.

involve feedback (where the output returns to the input). State is a kind of delayed feedback of this sort, where it is written as output and read as input later.

These factors make computation difficult. Cascading and feedback both require noise dampening, which requires non-linear signal processing. Some of the combinatorial functions themselves are non-linear as well. The key to both issues is non-linear processing.

Non-linearities come in several forms, either as continuous or discrete functions. A neuron is modeled as a non-linear transfer function and a transistor is a degenerate case of a neuron with a highly discretized function. Non-linearities can be supported by switches or transformational processing.

Most electronic computation uses switching, in which an input signal is used to control whether the output is connected to one of two or more other signal sources. These sources are typically the direct-current anode or cathode, so the output either sources or sinks electrons via the power supply. The input signal stops at the switch; the output is controlled by, but not a direct derivation of the input (Figure 6, left).

Optics also supports switching, but often far more slowly than electronics (15 ps for 22-nm CMOS [22], and roughly 10× faster for germanium devices), typically because optical switching often relies on mechanical mirrors (1 ms), bulk thermal (1 μs), or electro-optic properties (1 ns). High-bandwidth optical processing more typically relies on transforms in which the input waveform is converted to an output waveform (Figure 6, right), such as wave mixing – which are limited by frequency bandwidth rather than symbol rates. We call this "transformational" processing, and it is critical to optical computing [1].

It can be tempting to try to support optical computing using complex native optical capabilities, notably Fourier transforms and filters. These are linear transforms, and thus are insufficient to support computation [15].

It can similarly be tempting to consider optical computation as potentially being more energy efficient than electronic computation. Although this may be possible for optical switching (which is inherently too slow), current wave mixing is inefficient, losing approximately 1 dB of power in highly non-linear fibers (HNLFs) and 5 dB in periodically poled lithium niobate (PPLN) devices. Mixing also requires 0.1–1 W pump signals, which are consumed in the mixing process. As a result, it may not be appropriate to assume that mixing-based optical computation is power efficient.

2.4 Wave mixing

Non-linear processes can be used to perform computation, leveraging Kerr non-linearities to combine optical signals well into the THz range. In wave mixing, multiple optical signals at different wavelengths interact with each other in a non-linear medium, as graphically depicted in Figure 7.

Mixing can occur as either second-order ($\chi(2)$) or third-order ($\chi(3)$) non-linear processes. $\chi(2)$ processes can generate harmonics and compute sums and differences by mixing two input waves and a continuous wave (CW) pump to yield a third wave, known as three wave mixing. $\chi(3)$ can perform four-wave mixing (FWM), combining three input waves and a pump to yield a fourth wave.

In both types of mixing, phase-aligned optical signals are sent through a non-linear medium of the corresponding type. Each such medium has a characteristic zero-dispersion wavelength, also known as the center frequency.

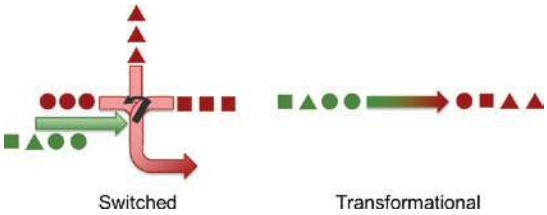

Figure 6: Switched vs. wave mixed processing.

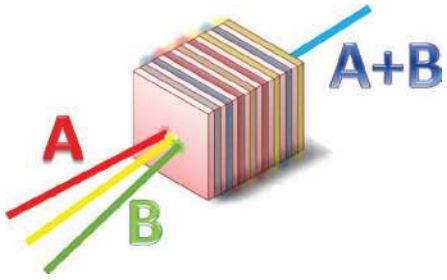

Figure 7: Graphical depiction of wave mixing.

Depending on the arrangement of the input signals and pump relative to this center frequency, a variety of results can be obtained. These include generation of conjugates, harmonics, and frequency-shifted copies of the input signals, phase summation, and phase difference. These devices are realized in a variety of ways. Periodically poled lithium niobate ($LiNbO_3$) is a common $\chi(2)$ medium, using a series of 5–50 μm layers of $LiNbO_3$ with varying polarization orientations (also depicted in Figure 7). HNLF is a $\chi(3)$ medium that can support FWM. PPLN can be difficult to fabricate as an integrated nanophotonic device because its layers can be difficult to orient in the natural direction of waveguides. HLNF typically requires tens of meters of fiber and is impractical as an integrated device. There are a variety of other $\chi(2)$ and $\chi(3)$ materials that can be integrated but are less commonly used in macroscopic laboratory systems, such as ours.

3 Requirements

An assessment of the basic requirements for high-bandwidth, long-distance communication and those for high-level computation can together be used to determine a viable unified approach. Optical communication is fundamentally driven by the need for dense symbol encodings (many bits per symbol), to overcome the serial nature of most long-distance optical media, as well as by the need to accurately regenerate symbols for relaying over global distances. Computation is driven by the requirements of processing, which include the need to support a field, the need to support uniform transforms, and the need to support serial algorithms.

3.1 Communication

Optics is necessary and already widely used for high-bandwidth, long-distance communication. It is efficient only if the encoding is dense, supporting many bits per symbol, and that it can be efficiently restored at relays, the latter because unamplified signals can travel only a few tens of kilometers in fiber and amplification typically decreases the optical signal-to-noise ratio (OSNR).

3.1.1 Density

Communication over long distances requires symbols with a high density, i.e. where each symbol is one of a large set; thus, each symbol encodes many bits [9, 23]. Commercial modulating electronics typically achieves rates up to 40 G symbols/s, and a binary encoding would limit an optical channel to 40 Gb/s. Parallel channels (WDM over one waveguide or one wavelength over multiple waveguides) can be used over short distances, but variation of dispersion, loss, and latency between different channels or fibers typically limits their use to datacenter scales or low data rates (10 Gb/s per channel).

Germanium electronics can modulate optics at higher frequencies, up to the serial electronic limit of 600 GHz. Unfortunately, use of such signals at the necessary voltage ranges (e.g. to drive a Pockels cell) often results in the driver circuit becoming a (very inefficient) transmission line because the drive signal wavelength approaches the length of the interconnect. At 600 GHz, the 1/10 wavelength rule of thumb indicates that traces longer than 30 μm become transmission lines. As a result, there are only two ways to achieve high density – either natively convert parallel electronics into multibit symbols (e.g. quadrature phase-shift keying – QPSK) or optically multiplex binary-modulated streams. OTM assumes the former may be possible, but explores the latter in Section 5.2.3.

3.1.2 Regeneration

Regeneration is needed to support communication over large geographic distances. It enables long-distance transmission by recovering symbols before they become ambiguous by both increasing the OSNR and restoring power. This is already a very active area of current optical processing research [24–27].

3.2 Computation

Optics is less natively compatible with computation. Computation requires processing to support the function operations of a mathematical field. It requires accurate composition, to cascade values through a series of these function operations. For all but the most basic level of the hierarchy, computation requires state to enable feedback from the output to the input, which enables context-sensitive processing. Finally, computation is often most useful when it is flexible, so that it can be "reprogrammed" or reconfigured for a variety of different tasks.

3.2.1 Processing

Computational processing requires functions that support the processing operations required for a field [2]. Although

simple operations support basic counting, the full power of a field is required for error detecting/correcting checksums, encryption, and authentication.

In discrete mathematics, a field is a set of symbols and two binary operators (i.e. two-input functions). The symbol set is closed under both operators, each operator has an identity element ($a \cdot I = a$), and every element of the set an inverse under each operator ($a \cdot a^{-1} = I$). Both operators are commutative ($a \cdot b = b \cdot a$), and one operator is distributive over the other [$a \cdot (b \circ c) = (a \cdot b) \circ (b \cdot c)$].

A field defines common arithmetic. It is also the foundation of most mechanical and electronic computing, whether in high-density symbols (e.g. decimal for the Eniac) or binary (Boolean logic). Because these are all based on the same abstract concept, they all easily support the same common arithmetic and logic operations. Other, more obscure mathematics have been considered, including residue arithmetic [28] and directed logic (which replaces scalars with vectors) [29]; however, neither provides a sufficient advantage for optics over that of a basic field.

Support for a field requires considering how its symbols are represented and how they are processed by its binary operations. Information is represented as a set of discrete symbols and a value mapping (a value assigned to each symbol). Candidate encodings include optical phase (e.g. PSK; Figure 8, right) and {phase, amplitude} pairs (e.g. QAM; Figure 8, left). Value maps assign specific information to each symbol (e.g. numbers in Figure 8).

Binary operations can be performed by switching or transformation, and optics can support high symbol rates only through the use of wave mixing transformations. Transformation processing is simpler with continuous, uniform transitions between its states (Hamiltonian paths). Such processing strictly requires unambiguous Hamiltonian paths; otherwise, on the way between transitions, the transformation device would enter a state with multiple outcomes, and the result of the computation would be ambiguous.

Different encodings can enable or impede these binary operations [30]. Consider the "+1" Hamiltonian path whose values increase by 1 (shown as the dashed paths in Figure 8). The PSK path is continuous, uniform, and unambiguous; the transitions are in the same relative direction, each transition is the same relative phase shift, and the path never crosses itself. The QAM path has discontinuities – short jumps (3–4, 7–8, 11–12), and a large jump (15–0). None of the transitions can be represented as a single transform of either absolute or relative amplitude-phase pairs. The QAM path also crosses itself in five different places. This is why M-PSK encodings can support field operations, whereas QAM cannot.

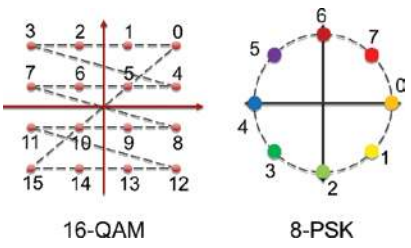

Figure 8: Encoding impact on state transition properties.

3.2.2 Composition

A field also relies on the ability to chain its operations, where the output of one operation is cascaded into others. This enables creating complex functions through the composition of simpler ones [17]. Composition requires that information be regenerated, to avoid the accumulation of noise during this cascade.

There are a variety of ways to regenerate symbols, i.e. to restore power levels and OSNR, so the information can be distinguished from background noise. Some methods rely on pilot signals that experience the same noise as the data; others remove noise by taking the differential of the signal (the difference between symbols). In the latter case, relatively stationary noise accumulated through transmission or processing can be removed because it cancels out between adjacent symbols.

3.2.3 State

All computations beyond combinatorial logic require a state that persists, even if temporarily [1]. A single state is required for an FSM, many states accessed in a limited way (the value on the top of a "stack") for a PDA, or arbitrary access to a "tape" of many states for TMs. In all cases where the state exists, that state depends on previous input data and affects the interpretation of subsequent input – i.e. it is effectively a time-delayed recirculation of information. Without a persistent state, only combinatorial functions can be generated; all feedback would be prohibited. This feedback enables recursion, a key requirement for general-purpose computation.

There are various forms of storage that can be used for optical data. Permanent storage, with arbitrary persistence, requires conversion of the optical signal to another form (e.g. electronic) because photons cannot be "stopped". Ephemeral storage, such as fiber loops, can

be used instead, as long as the data can be recirculated if needed. There is a large body of work in using ephemeral storage, beginning with acoustic waves in mercury delay lines in early electronic computers and in recent reservoir computing [31].

It may be important to also assume a complete reconsideration of computation architecture based on these capabilities combined with the serial nature of communication. Current architectures focus on increased parallelism and synchronous, phased operation due to the use of switched, highly integrated transistor systems. Digital processing of optical communication streams is better matched to asynchronous serial processing.

3.2.4 Control/programmability

A computational device can be fixed, supporting only a single algorithm, or can be reconfigurable to support the entire variety of algorithms available for a given level of complexity. For in-transit computation, reconfigurable devices are much more useful because the specific forwarding, data filtering, or cryptographic operation is very likely to vary. This reconfiguration can be accomplished either by changing the signal path directly (known as "rewiring", even for optical systems) or by using some portion of the data (known as a "program") to control the signal path. The latter falls into two general classes: using program data that is distinct from the processed data (called the "Harvard" architecture) and treating program and processed data as co-mingled (called the "von Neumann" architecture). It is important to appreciate that all three of these variants – rewiring, Harvard, and von Neumann – have equivalent capabilities.

Rewiring is easy to achieve in an optical system using any type of deflection switch controlled by a configurable state [1]. The state need not be represented optically, as it applies only to reprogramming. The deflection switch need not be fast or lossless (i.e. it can interrupt the signal flow), as it can be used only when reconfiguring an entire system. This allows the use of a variety of optical switches for reconfigurability, including MEMS.

4 Implications of requirements

The communication and computational requirements discussed in the previous section result in a set of implied design requirements. These requirements suggest a preferred data-encoding format used to support both communication and computation, and a preferred approach to supporting the field operations needed for computation. They also help indicate requirements for regeneration methods and the overall system implementation.

4.1 Encoding

Support for in-transit processing of optical communication requires that a single format be used for both communication and computation. These data must be encoded with a high density and in a digital format to support both efficient high-bandwidth communication and complex computation using composition and recirculating state.

Candidate multibit encodings modulate phase (M-PSK), power (PAM), or both (QAM). Other encodings are limited to binary values (e.g. OOK, polarization) or create independent channels rather than codes [e.g. polarization, OAM, WDM, code division multiple access (CDMA)]. The need for transformational processing favors one-dimensional encodings, e.g. M-PSK or PAM, rather than those that vary multiple properties, e.g. QAM and CDMA.

Of the candidate encodings, only M-PSK supports the required Hamiltonian properties (continuous, uniform, and unambiguous). M-PSK additionally is desirable because its symbols have uniform power. Receiver processing of phase encodings requires coherent detection, which depends on carrier recovery; however, this is already mature and efficient.

4.2 Computation

Support for in-transit computation requires processing that can support data rates of 1 Tb/s and beyond. For optically encoded data, only wave mixing supports these rates and is already known to support many of the operations needed to emulate a field. Additionally, serial algorithms are preferable to parallel, because optics can support native serial Tb/s symbol rates more easily than it can support the complexity of parallelization.

Wave mixing includes a variety of processes available in non-linear optical materials, including harmonics generation, sum frequency generation (SFG), difference frequency generation (DFG), and optical parametric amplification and oscillation. The key feature is that optical signals affect how other optical signals are combined; thus, the input and output symbols are all encoded in the same domain (a requirement of a field). They differ from

processes that vary optics based on other input domains, e.g. electro-optics (as in Pockels cells), which may be useful for device configuration and reprogramming but cannot support all-optical computation.

Wave mixing is supported in both $\chi(2)$ and $\chi(3)$ materials. SFG/DFG is supported in $\chi(2)$ materials via three-wave mixing, whereas $\chi(3)$ materials are required for FWM. For Tb/s computation, $\chi(2)$ materials currently appear to be sufficient to support the optical processes needed for field operations.

Optical wave mixing is inherently serial, supporting the data rates needed in a single, linear sequence of symbols over a single wavelength. By contrast, most electronic computation relies on a significant parallelism – 64b is common for primary processors, with 256b and above used in graphics and network processors [1]. In electronics, parallelism is used to overcome the limited symbol rate of individual components – e.g. typically 3–6 Gb/s.

The use of parallelism to overcome limited electronic symbol rates comes with a significant cost. Some functions that are the most complex when parallel become nearly trivial when serialized; addition has O(NlogN) elements and O(logN) delays for N-bit summands (Figure 9, left), because each bit of the output is computed based on whether the bits to its right of both the summands would generate a carry (known as "carry look-ahead"). The same operation in serial optics requires only one element, using delay and feedback instead (Figure 9, right), because the carry is propagated through the summation as each output bit is computed.

Many functions important for in-network processing support efficient serial implementations, including statistics (sums, averages, standard deviations), pattern matching (correlators), error processing (checksums, CRCs), and cryptographic processing (authentication, integrity protection, encryption). In some cases, there are simple serial equivalents of conventional algorithms (e.g. hop-count decrement), and in other cases there may be algorithms that compute different values with similar properties, such as alternate checksum algorithms. Cryptography is notable among the latter because many current algorithms are designed for parallel architectures and rely on indexing (lookup) functions, both of which are difficult in optics. Alternate algorithms exist that avoid these issues and may be better suited to use with optical signals.

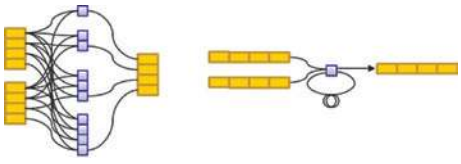

Figure 9: Parallel vs. serial 4-bit addition.

There are many issues in developing a serial, optics-friendly algorithm. In some cases, there are known equivalent serial circuits or algorithms for parallel methods – in some cases, the serial variant comes first, such as for addition and multiplication. In other cases, there are no dependencies between adjacent bits, so serialization is trivial – as for bitwise logical operations (AND, OR, etc.). In an abstract sense, a programmer or circuit designer needs to consider the dependencies between the bit positions of the input and output bit fields and the operations that combine them as a graph, and seek a Hamiltonian path through that graph. This remains somewhat of an art.

Additionally, an optics-friendly serial algorithm needs to minimize the amount of state recirculation, to reduce the amount of regeneration needed. In some cases, this can be accomplished through redesigning the circuit to compute the same results. More often, such feedback reduction may require using a different algorithm, e.g. one that computes a different but similarly useful result. A good example is encryption, in which the typical method of cypher-block chaining uses part of the result of encrypting one block of data to initialize the encryption for the next block. This chaining accumulates data through signal feedback that can degrade and require regeneration. An alternate approach is known as "counter mode", which uses a simple counting index as the seed instead, avoiding the continued use of recirculated data.

4.3 Regeneration

The ability to cascade computation, to include state as context, and to transmit information over very long distances require that symbols be regenerated. Regeneration includes removing noise that increases symbol ambiguity and increasing power to improve detection, and is sometimes referred to as "redigitization".

For M-PSK symbols, power should be constant across all symbol values and is thus relatively simple to restore using simple saturation amplification. The challenge is that amplification increases phase noise and phase noise reduction often decreases power. A solution is needed that can increase both power and OSNR simultaneously.

It is important that signal regeneration retain symbol representation and meaning. There are efficient methods that reduce noise by converting from one symbol format to another or from one symbol semantics to another. For example, if noise is relatively stationary, it can be possible

to cancel noise by taking the signal difference [27]. A QPSK stream enters as A, B, C, D values and is output as the sequence (A–B), (B–C), (C–D), etc. Noise is cancelled out and the output may retain QPSK encoding; however, the symbol semantics have changed. Such changes interfere with computation because advanced levels of complexity rely on recirculation of symbols as state (Figure 4) [32].

It is also important that signal regeneration supports a format that is compatible with high-bandwidth transmission and computation. Efficient regeneration exists for OOK, which may support computation; however, this binary encoding is inefficient for transmission [33]. Efficient regeneration also exists for differential PSK, where the symbol sequence represents the difference between phases of successive information values [34]; however, that format does not support computation on individual optical symbols.

It is also important that regeneration relies only on the data that has experienced computation or communication. Some methods rely on the data symbols being accompanied by a pilot tone or reference symbols, either on a different frequency or interleaved with the data symbol sequence. These supplemental signals will not experience the same computational path as the data should, so they will not accumulate correlated properties (noise, power degradation, etc.). As a consequence, they need to be avoided because they, too, do not preserve the semantics of the computationally processed data stream.

4.4 Implementation

The requirements, thus far, strongly suggest that M-PSK symbols are preferred and should be processed using wave mixing. The resulting computational system is highly sensitive to phase noise and variation. Benchtop experiments are notoriously difficult to phase stabilize; they often create interferometers as a side effect of selective signal processing, which become very sensitive instruments that unintentionally measure the laboratory environment. Mixing can require phase-aligned sources that are similarly difficult to stabilize in a macroscopic configuration.

As a result, it is critical to developing optical computation for in-transit processing using integrated nanophotonic components on a common substrate. This requires combining non-linear $\chi(2)$ elements, linear elements (filters), and optical vias on a single chip. It also requires including phase-aligned sources, e.g. frequency combs, with relatively high power (1 W) on the same substrate.

5 Experiment results

The following is a summary of experiments to support computation and regeneration on M-PSK symbols. Computation is needed to support the two binary operations of a field, and regeneration is needed to support composing these operations, using state for high-level computational complexity and communication.

For computation, methods are shown to support key field operations. Serial algorithms compatible with optical processing are also shown. The key remaining challenge, besides implementation complexity (requiring hybrid integration), is the computational need for regeneration, notably to support carry generation.

Regeneration of M-PSK symbols requires phase squeezing, often via phase-sensitive amplification (PSA), to restore the digital nature of the stream. It requires amplification to restore power levels. Finally, because of some of the limitations of phase squeezing to support higher-density encodings, it may be important to support optical aggregation and deaggregation of lower-density symbols that can be more efficiently regenerated.

5.1 Computation results

Our team has explored the potential for optical computation by developing increasingly complex algorithms with various encodings. A circuit was developed to decrement an OOK-encoded hop-count field, one of the simplest operations performed in a packet switch. A corresponding M-PSK variant was designed, as was a more complex packet checksum calculator. The checksum relies on components that support modular arithmetic and carry generation. In addition, the team explored the potential to support recirculating state, as is needed for most high levels of computational complexity. Note that in all cases, we assume that packet boundaries are indicated, e.g. using an out-of-band signal or by correlation detection of a preamble pattern, as in our previous work on packet header pattern matching [35].

5.1.1 Hop-count decrement and checksums

The simplest network function is hop-count processing. Packets contain a hop count (a.k.a. “time to live”) field to ensure that forwarding loops and misdirected traffic does not overload the network [36]. Each packet is given an initial count (e.g. 255), which is decremented at each hop visited. When the count reaches zero, the packet is

silently discarded. This avoids the need for a coordinated mechanism to flush out stale packets.

Hop-count processing is currently implemented using either a general-purpose CPU or complex, dedicated bit-parallel arithmetic hardware. Subtraction is typically implemented as addition of "– 1", a complex operation cascading "carries" across the bit field in parallel, similar to the approach shown for parallel addition in Figure 9 (left).

A simpler approach is possible when processing the data serially, using only three active components (Figure 10). Subtraction of an unsigned bit field involves inverting the "0" values (starting at the low-order bit) until the first "1" value is encountered. That value is inverted, and all subsequent inputs are copied (Figure 10, left). The entire system can be implemented using only three active elements (Figure 10, right) – an inverter, a 2×2 switch, and a set-reset (S/R) flip-flop (FF). The switch is set to select the inverted input until the first "1" bit arrives. That bit sets the flip-flop, whose output is delayed by one symbol to change the switch just before the next symbol. After the change, the input is copied to the output without inversion.

Our team implemented this mechanism for a binary OOK encoding, using an electronic S/R FF (Figure 11) [36]. All-optical OOK micro-ring S/R FFs have been demonstrated [37], but were too costly to replicate.

The mechanism has been extended to support M-PSK multibit encoding, replacing inversion by a "– 1" transformation, and all of the optical components already exist for M-PSK encodings. For example, "– 1" transform for 8PSK encoding is a $-\pi/4$ phase shift that can be accomplished using a fixed difference in signal path, e.g. of 187.5 nm for a 1500 nm wavelength signal. The OOK micro-ring flip-flop can be extended to support M-PSK encoding using phase interference with a reference signal as input, and a Mach-Zehnder modulator can serve as an optical switch that can be controlled using an independent M-PSK input (i.e. the flip-flop output).

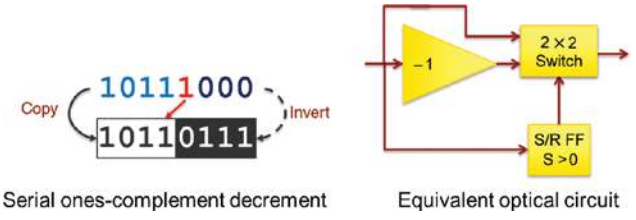

Figure 10: Serial ones-complement subtraction.

Internet packet routers support hop-count processing, but also rely on other, more complicated functions. One such function is a checksum, a mathematical function that validates that the contents of a message have not been corrupted in transit. There are a wide variety of such checksums used in various protocol layers, including cyclic redundancy checks (CRCs, as for Ethernet and Bluetooth), parity (a 1-bit CRC, as for memory checks), and very complex hash functions (e.g. MD5, SHA-1) that also detect deliberate tampering. Parity and CRCs are very simple operations using position-based symbol feedback and serial modular arithmetic (shown as an XOR symbol), and are already well suited to all-optical implementation (Figure 12). The key challenge to their computation is maintaining signal integrity through the feedback of state, i.e. regeneration. Note that Figure 12 depicts a binary implementation using XOR gates; multibit encoding would use optical adders for modular arithmetic [38].

The Internet checksum is of particular interest as it is used for IP version 4 packet header error detection and as a check on the entire end-to-end message at higher protocol layers, e.g. TCP and UDP [6]. This checksum is a 16-bit one's complement sum of the message, i.e. the entire message is added as adjacent byte pairs of 16 bits. The summation function is one's complement, which differs from the more familiar two's complement arithmetic by looping its high-order (most significant bit) carry-out back

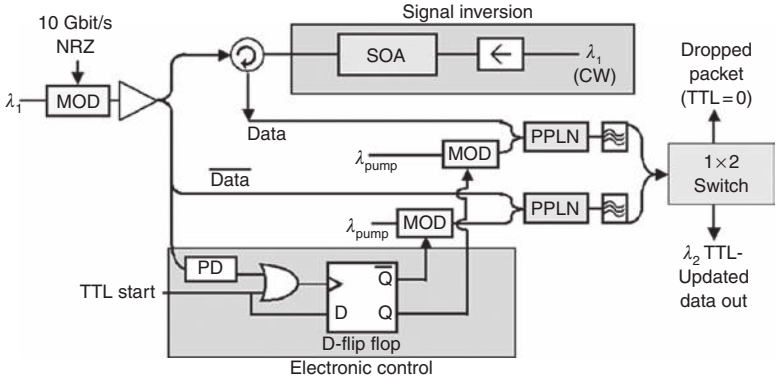

Figure 11: Optical packet hop-count decrement.

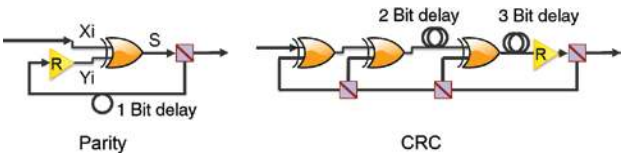

Figure 12: Parity and CRC via simple serial feedback circuits.

around as a low-order (least significant bit) carry-in. This feature of one's complement summation enables a simple serial design.

When computed on two's complement hardware, a one's complement sum is typically computed using a larger accumulation register (e.g. 32 bits). As values are added into the accumulator (in bit-parallel), the high-order carry-out bits accumulate in the top half of the register. When the computation is complete, the carries are "folded over" and added back into the bottom half of the register. This operation is simple in typical two's complement hardware; however, it is even easier in one's complement, where the entire checksum can be computed using a single "full adder", i.e. one device that adds two symbols and a carry-in, and generates the sum and a carry-out (Figure 13). This serial circuit relies on delay and recirculation to overcome the need for one adder per bit position and complex carry-look-ahead, as is needed in the corresponding parallel circuit (Figure 9). Because one's complement accumulates carries through wrap around, there is no need to reset the circuit as bits are summed in series; instead, the carries wrap around (from Co to Ci) the same way as the partial sum (from S to Yi).

The primary challenge in implementing a CRC or the Internet checksum using serial optics is the need to recirculate state, which requires regeneration. An additional challenge for the Internet checksum is the need for a full adder, which is significantly complex in optics.

5.1.2 Modular arithmetic

Support for the mathematical field that is the basis of computation typically requires support for modular arithmetic, as this is the basis of the operations in the most familiar fields. In binary logic, the corresponding field is Boolean algebra, which includes addition (OR), multiplication (AND), and complementation (NOT). For fields with more than two symbols, modular operators are needed, whose outputs "wrap around".

Members of our group previously developed modular arithmetic for M-PSK symbols [39], as depicted in Figure 14. Three symbols encoded as QPSK were added and subtracted in a non-degenerative FWM process in an HNLF (Figure 15). Three-input operations are often more useful than two-input operations, as they natively support the combination of two new values as well as a carry-in from either a previous or adjacent stage. A similar mechanism was used to extend this QPSK method to 8PSK and 16PSK encodings [38].

5.1.3 Extension to a full adder

Although modular arithmetic is useful on its own, e.g. for CRC calculations, most computing relies on the ability to extend the symbol space of a field using positional notation, so that a fixed set of symbols can represent an indefinitely large number of values. Support for positional notation requires the extension of modular arithmetic to carry-based arithmetic. This evolution typically requires the design of a half-adder and full-adder, the former combining two symbols to generate a sum and carry-out symbols and the latter combining three symbols: two input symbols and a carry-in symbol. It is a basic exercise to extend binary half-adders to create a full adder, using two half-adders and an OR to merge the carry-out values (Figure 16, left). A less typical design can combine multivalued (i.e. so-called "multibit") half-adders to create a corresponding full-adder; however, this design requires

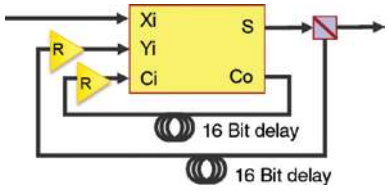

Figure 13: Serial, optics-friendly Internet checksum.

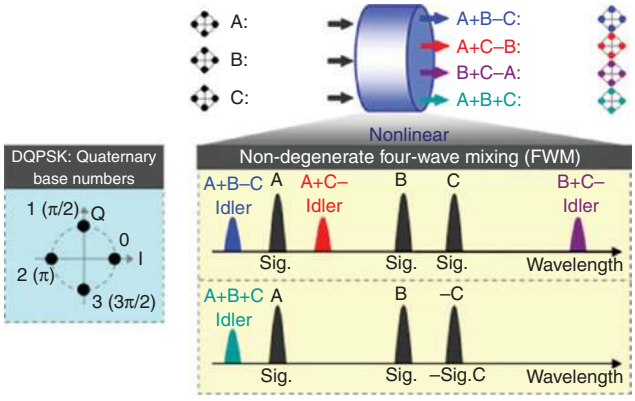

Figure 14: Modular arithmetic via wave mixing concept.

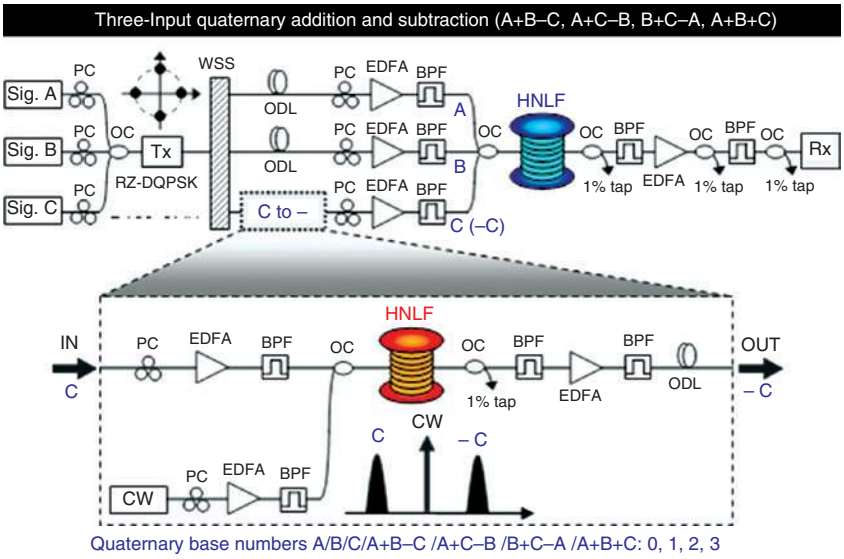

Figure 15: Modular arithmetic system design.

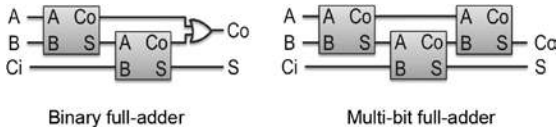

Figure 16: Full adders via half-adder composition.

three half-adders (Figure 16, right). The final carry-out (Co) of the last half-adder is never active and can be ignored.

Half-adder and full-adder designs rely on both modular arithmetic (for the sum) and carry generation. Modular addition loses the information needed to indicate whether a carry is needed or not, e.g. for 8PSK, $1+2=3$ but also $6+5=3$ (Figure 17). In M-PSK modular addition, no part of the FWM "remembers" whether the phase simply rotates as needed ($1+3$) or "wraps around" ($6+5$).

Our team has proposed a method to determine whether a carry is generated or not. The goal is to keep track of whether the accumulation of phase would have "wrapped" or not. This approach first halves the original values, so that instead of adding $1+3$ or $5+6$, the values of $0.5+1.5$ and $2.5+3$ are computed (as shown in Figure 18).

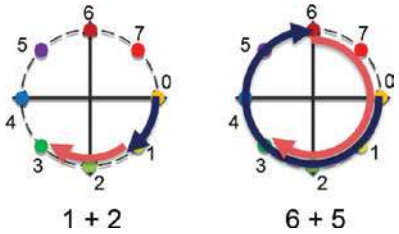

Figure 17: Loss of carry context during modular add.

The resulting value cannot wrap around because the inputs are at most half the maximum symbol value. The half-sum ends up either on the bottom or top half of the phase diagram (Figure 19, left). The bottom half indicates no carry and the top half indicates carry. The key is to then squeeze the phase of the half-sum to represent two distinct values (Figure 19, middle) and then shift those values to represent the appropriate carry symbols (Figure 19, right).

The circuit for accomplishing this method is based on shifting the input symbols using a CW pump, summing the result using modular addition, and then squeezing and transforming the result (Figure 20). It demonstrates the way in which carry generation relies on phase squeezing. This circuit has been proposed and analyzed, but it

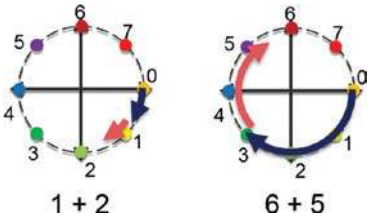

Figure 18: Carry context via adding half-phase.

Figure 19: Half-phase add transformed into encoded carry out.

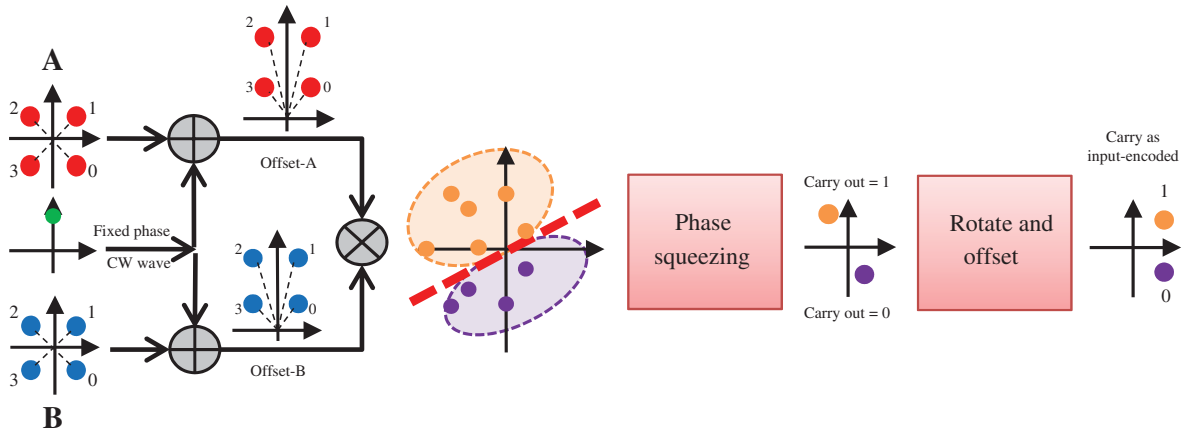

Figure 20: Generating the carry of the sum of QPSK symbols (A) and (B) via half-sum and phase squeezing.

is currently difficult to implement using benchtop equipment due to the number of cascaded components and the constraint that signals cannot be regenerated between the intermediate stages (as this would defeat the overall goal).

5.1.4 Need for recirculating state

Computation beyond simple combinatorial logic requires both retaining state and recirculating that state back into the combinatorial logic together with the input. As noted earlier, various levels of computational complexity hold state in different forms (a single value for an FSM, a stack for a PDA, or an arbitrary-access "tape" for a TM) and for different lengths of time. Even some of the simplest functions, including hop-count decrement, CRCs, and Internet checksums, all require this type of state recirculation.

This state either needs to be continually recirculated (as with an FSM) or held in a way that emulates persistent storage, which may itself require recirculation (e.g. in a delay loop). Optical signals can be delayed off-chip using fiber loops and on-chip using convoluted waveguide paths. On-chip delays are limited in length to a few centimeters; however, the capacity of that delay increases as baud rate increases, e.g. at 1 T baud, each 1 cm of waveguide can store 33 symbols.

There are many forms of "limited lookback" computation that restrict the amount of time a state is held, notably linearly bounded TMs that can leverage such limited storage. These approaches assume that state is recirculated only a limited number of times or expires after a fixed period (whether recirculated or not), which can limit the kinds of computation that they can perform.

Regeneration can be coupled with delay-based storage to extend its lifetime and/or enable increased recirculation. This is critical because most examples of computation assume arbitrary recirculation of state through the combinatorial logic and/or indefinite storage of state. A simple 8-bit binary hop count requires seven such cycles; the Internet checksum, over a typical 1500B Internet packet, could require as many as 750 such recirculation cycles. Optical signals tolerate traversing only a very small number of devices before signal integrity cannot be recovered, so regeneration is critical to supporting these advanced levels of computation for even the most basic operations. Similarly, the limitations of on-chip delays require regeneration to recycle state over longer periods.

5.2 Regeneration results

Regeneration of optical signals is a very active area of investigation, as it is critical to relaying signals over long distances through fiber that consumes power and distorts symbols. There are two distinct aspects to this regeneration: increasing the resolution of different symbols and increasing the difference between a symbol and background noise. The former is known as symbol restoration and the latter amplification. For M-PSK symbols, which are the only symbols known to support computation and high-bandwidth communication, restoration is known as phase squeezing. This section explores recent results in phase squeezing and amplifying M-PSK symbols. Some of the limits of phase squeezing suggest that it may be necessary to consider aggregating and deaggregating medium-density encodings (e.g. QPSK to 16PSK).

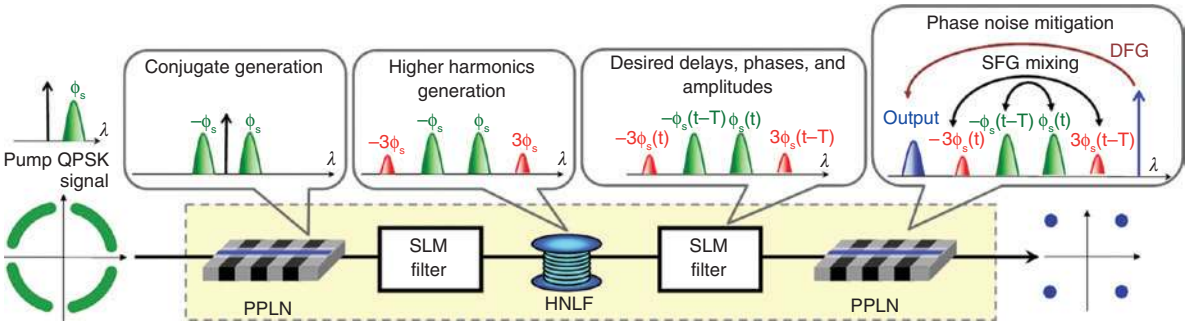

Figure 21: Non-PSA phase squeezing system design.

5.2.1 Phase squeezing

Signal regeneration begins with restoring the distinctness of the symbol representations. There are a variety of methods that have been used to restore M-PSK symbols, including differential regeneration and PSA. Note that phase restoration focuses on reducing accumulated phase noise and does not address other noise sources (e.g. amplitude noise).

One of the most effective methods is differential regeneration, often referred to as "non-PSA" squeezing, in which a symbol stream is processed with a one-symbol delayed copy of itself [40] (Figure 21). This method assumes that phase noise is mostly stationary compared to the symbol stream, i.e. that adjacent symbols experience similar noise, irrespective of their symbol values. This might occur if a symbol stream is noisily amplified or if it experiences phase-insensitive transmission loss.

In non-PSA squeezing, harmonics are generated in a non-linear medium (a PPLN or HNLF) using wave mixing injected pumps. In two different stages, the signal is first copied as a conjugate and then the third harmonics are generated. The conjugate and third harmonic of the original signal are delayed by one symbol, and combined using SFG/DFG processes in a final non-linear device. The resulting output is the differential of the input, i.e. it represents the difference of adjacent symbols. Because the phase noise is relatively stationary, it cancels out of this difference and OSNR is improved, as shown in Figure 22.

The challenge with using non-PSA squeezing is that the output is semantically different from the input. Even though both are represented using the same symbols, the meaning has changed. Computation requires that each symbol be independent, so it can be manipulated without needing adjacent context. Differential output lacks this property, and cannot support computation. Further, it is impossible to recover a non-differential equivalent, e.g. by "integrating" the output signal [32]. Doing so requires inserting reference symbols in the data stream (to provide

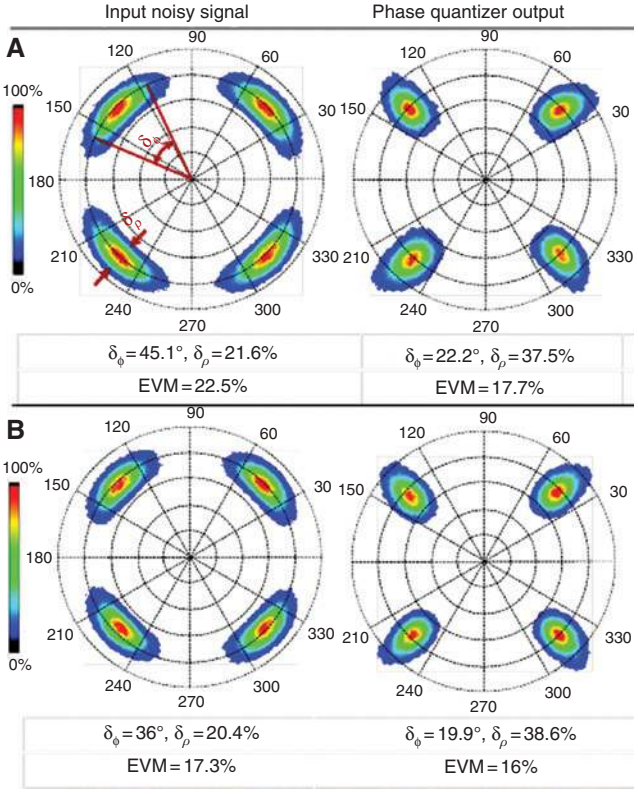

Figure 22: Non-PSA phase squeezing results.

the "constant" needed to perform integration); however, those reference values would have needed to be kept separate from the data and not participate in computation.

A different approach is to amplify signals that are close to valid symbol values and attenuate signals that are not. This is known as PSA; it does not actually correct any phase variation directly, but rather it tries to discard the portion of the signal that is slightly phase misaligned and to copy or reinforce the portion that is aligned. The particular variant explored by our team is non-degenerate PSA, which uses a distinct idler to amplify and select the desired portion of the signal [27] (Figure 23). Degenerate

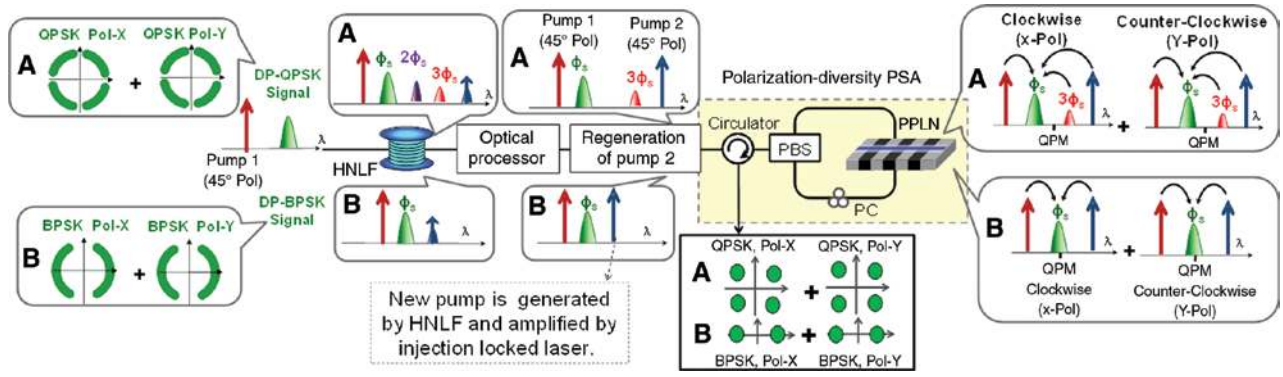

Figure 23: Non-degenerate PSA phase squeeze system design.

PSA overlaps the idler with the signal, but is more difficult to control.

This system uses pumps to generate harmonics in a non-linear medium (an HNLF) and combines those pumps and signals in a SFG/DFG process in another non-linear medium (a PPLN). There are two complications to this approach – the system requires generating the harmonic in direct proportion to the number of distinct symbols – e.g. QPSK requires the fourth harmonic, 8PSK requires the eighth, and the system implementation includes an interferometer as a side effect, which is very difficult to stabilize.

The harmonic required depends on the number of symbols. Our experiments showed a 3.6 dB OSNR gain for BPSK using the second harmonic, but only a 0.4 dB OSNR gain for QPSK using the fourth harmonic (Figure 24). As symbol density increases, the harmonics become more difficult to generate and are generated less efficiently, rapidly reducing the system impact.

The system design includes an interferometer created because of how the signals are processed. The first non-linear process converts the signal and pump to generate the needed harmonic. An injection-locked pump is added to amplify the harmonic; however, the harmonic has a much lower power, so the harmonic needs to be separated from the other signals for the injection-locked laser to synchronize to the desired signal. This creates an interferometer – one arm where the harmonic is coupled to the injection-locked laser and another where other signals bypass that laser. This device is very difficult to implement macroscopically on an optical bench using fibers, because the system is sensitive to vibration and thermal disturbances, and the result requires active phase stabilization.

In an attempt to avoid the need for this stabilization, our group explored PSA based on Brillouin amplification [41] (Figure 25). It replaced the interferometer and injection-locked laser with a Brillouin amplifier, which has an inherently narrow band and is able to selectively amplify the harmonic only. This method achieves 11 dB OSNR gain at 1E-5 BER for 10 Gb/s BPSK, 9.1 dB OSNR for 20 Gb/s BPSK, as shown in Figure 26. Brillouin PSA remains compatible with computation because it neither changes the signal semantics nor uses reference signals.

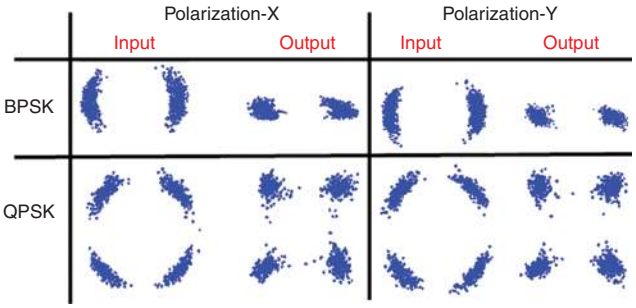

Figure 24: Non-degenerate PSA phase squeezing results.

5.2.2 Amplification

In addition to differentiating the symbols, it is important that each symbol be detectable. Improving the power level of the symbol is known as amplification. For M-PSK symbols, each symbol should have the same power level, so nearly any efficient amplification should suffice and will also reduce amplitude noise. However, it is important that amplification should not decrease intersymbol OSNR, i.e. it should not "spread" the phase of the symbols at the same time.

Our team explored using saturation amplification in an HNLF [40]. The results show improved signal power at the expense of an increase in phase noise. Figure 27 shows the noisy input signal, where the noisy input (left) is phase squeezed (right) using differential noise reduction (although already indicated as not viable for computation,

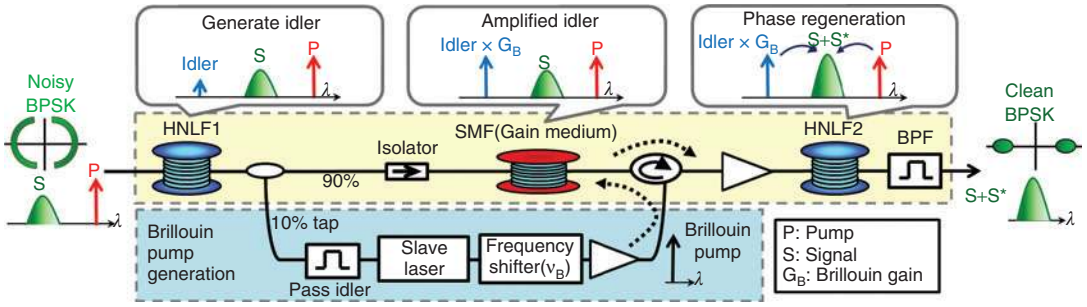

Figure 25: Brillouin phase squeezing system design.

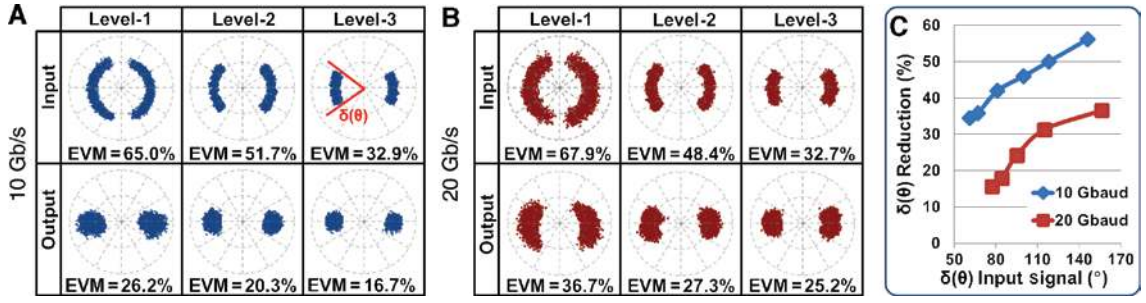

Figure 26: Brillion phase squeezing results.

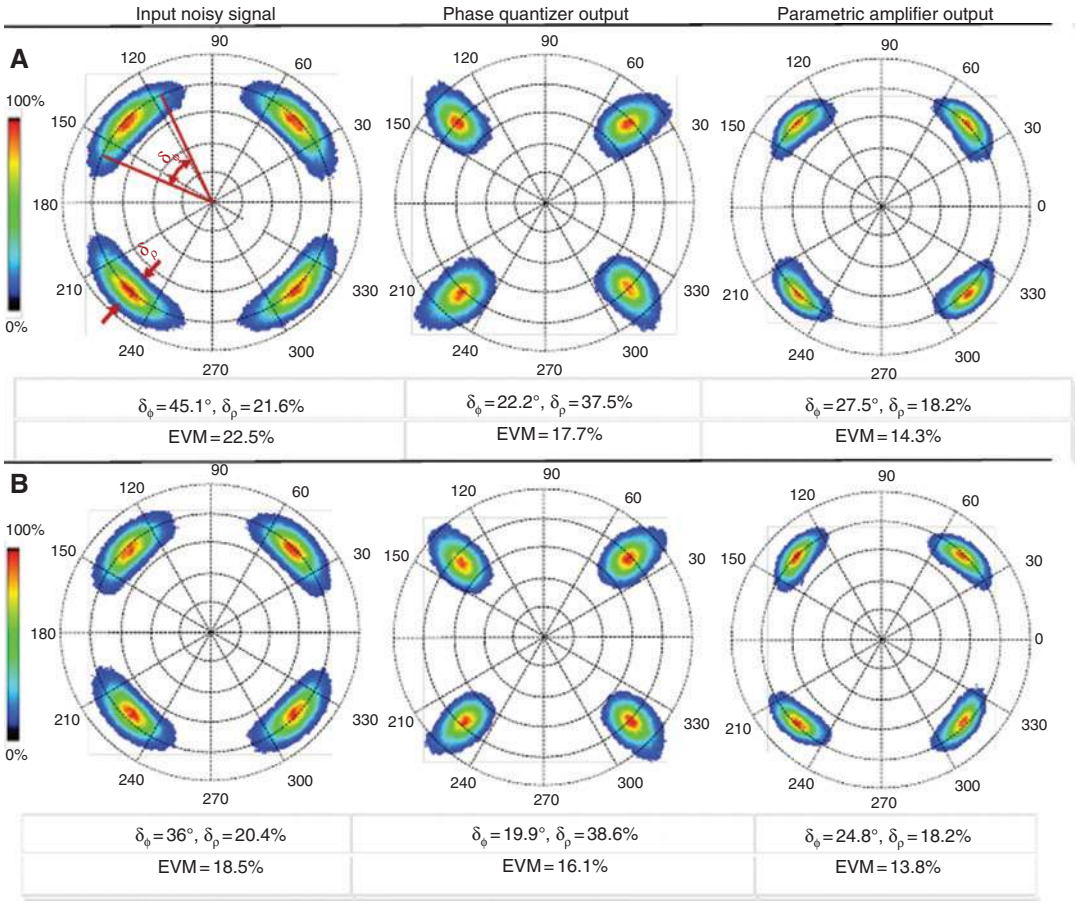

Figure 27: Saturation amplification results.

the phase squeezing method used in this experiment is not critical to the next step), and then saturation amplified (right). This type of amplification was able to compensate for power lost during the squeezing operation; however, it also introduced phase dispersion (the right signal is "wider" than the middle). The phase noise introduced by amplification was lower than in the noisy input signal, but the overall result did not squeeze enough.

Our team also explored using Raman amplification in an attempt to increase the signal power without introducing phase dispersion [42]. The concept is similar to the saturation amplifier, replacing the HNLF with a Raman amplifier (shown in Figure 28). The result had similar performance, resulting in a 19.3 dB gain for a −26 dBm input signal but had no significant impact on phase noise.

Our team explored aggregating and deaggregating (multiplexing and demultiplexing) M-PSK signals. The aggregation method (Figure 29) uses an apparatus based on phase-coherent addition (Figure 30) [43]. Both BPSK and QPSK signals were combined to generate QPSK (via two BPSKs). The approach depends on the availability of phase-locked comb sources.

Deaggregation of an M-PSK was achieved by separating the I and Q planes of the source signal, resulting in BPSK for QPSK inputs and PAM for higher-density inputs [44] (Figure 31). This system uses the apparatus shown in Figure 32, with results as shown in Figure 33. For computation, only the QPSK to BPSK conversion would be useful; the 8PSK to 4PAM output would require subsequent conversion from 4PAM to QPSK.

5.2.3 Aggregation and deaggregation

Because phase squeezing requires harmonics linearly proportional to the number of symbols, it may not be feasible to assume symbol regeneration for very dense encodings. It may be useful to consider long-distance transmission using aggregates of less dense encodings, where signals are deaggregated for computation. This is similar to the OEO/OOO conversion that should generally be avoided, but might be necessary and useful if limited to conversions that are feasible in all-optical devices. Note that this refers to combining or splitting multiple lower-baud-rate streams (on separate waveguides or different wavelengths in a single waveguide) into a single higher-baud-rate stream on a single wavelength.

6 Discussion

The viable ways in which digital optical computation can support network functions is influenced by deductions from basic principles, the results of experiments, and past experience. The basic principles dictate the need for phase-based encodings and wave mixing as processing, as well as the need for state. Experimental results indicate that this approach may be possible, but also indicate that it can be very challenging to implement. This entire set of investigation suggests that the most useful next steps toward all-optical in-transit computation will need to focus on hybrid integration, efficient regeneration, and possibly the need to support aggregation/deaggregation.

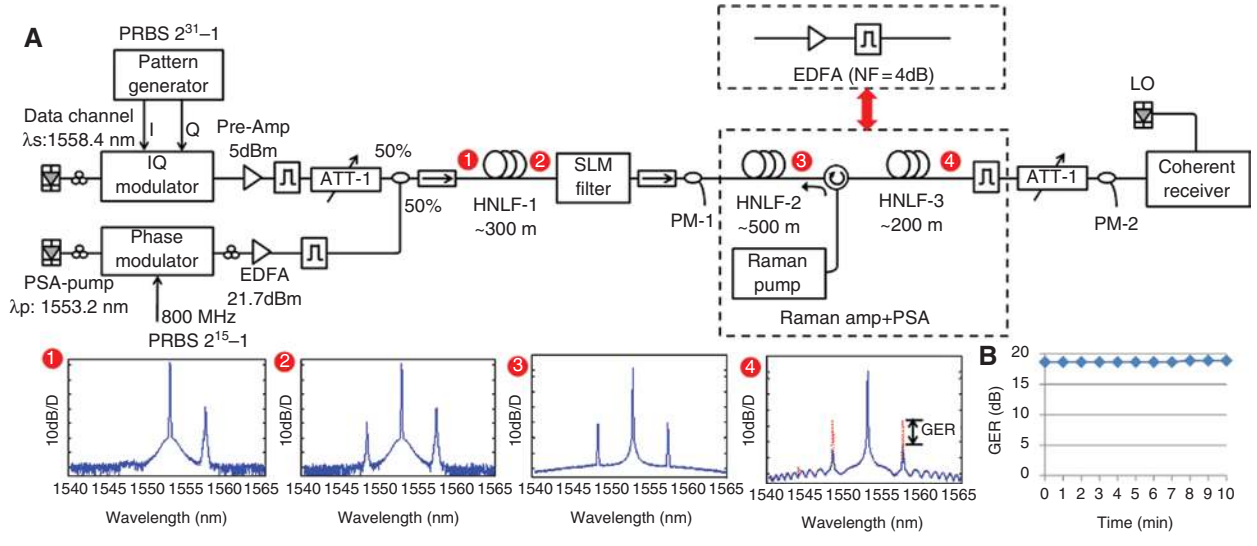

Figure 28: Raman-assisted PSA system design.

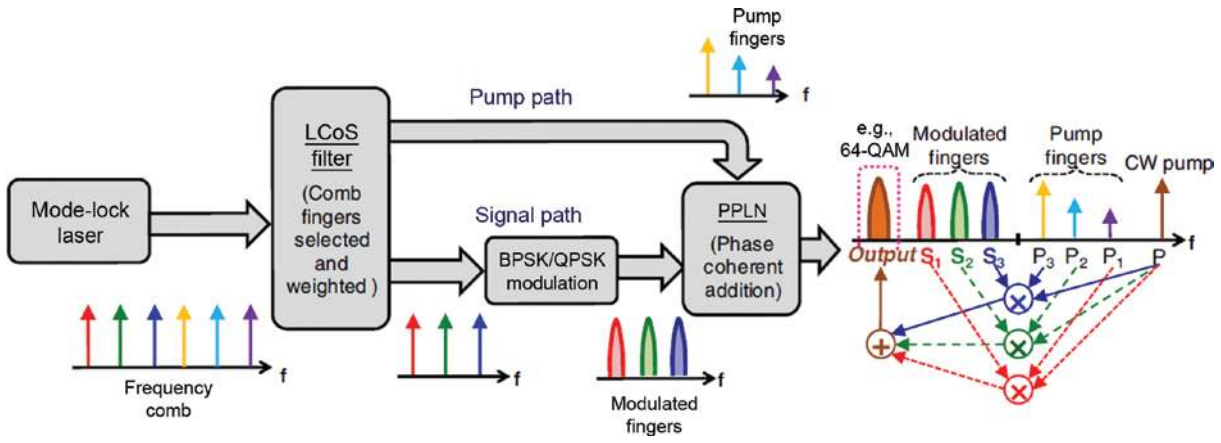

Figure 29: Aggregation concept.

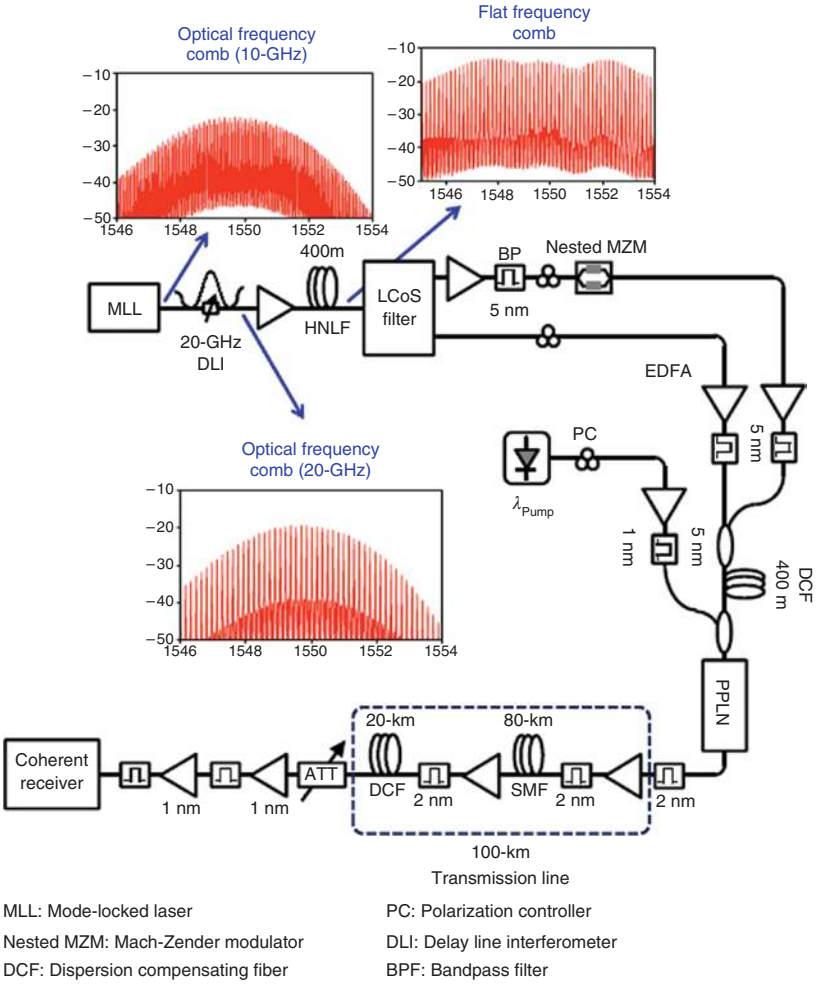

Figure 30: Aggregation system design.

6.1 Deductions

In seeking a way to support in-transit processing of optically encoded signals, only phase encodings support the needed field operations and wave mixing is required to process those operations at data rates that can exceed electronics. In addition, only digital encodings enable both operation composition and feedback to implement

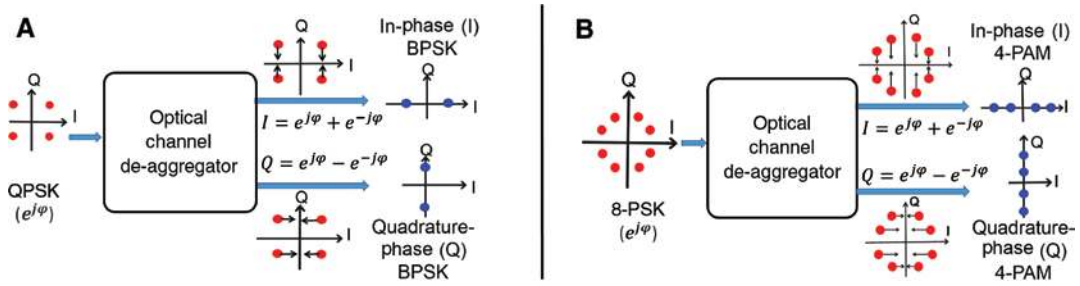

Figure 31: Deaggregation concept.

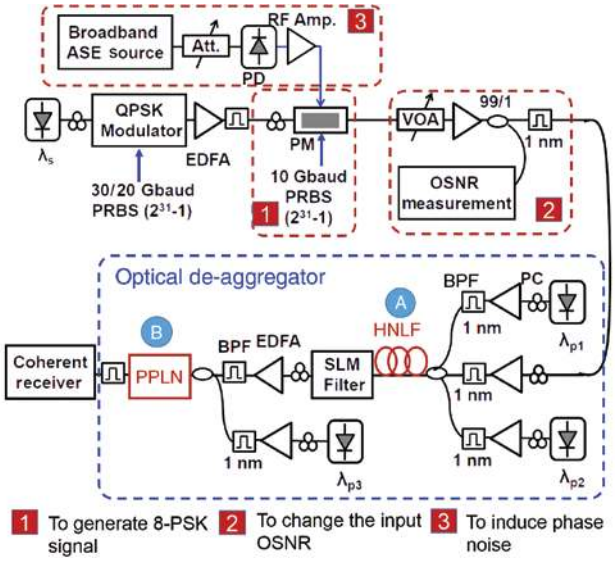

Figure 32: Deaggregation system design.

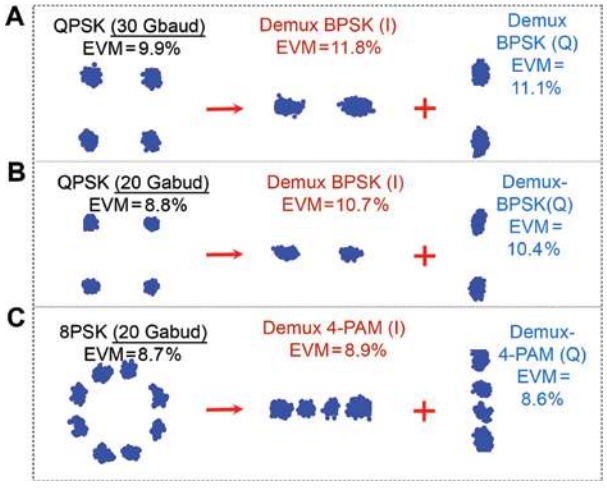

Figure 33: Deaggregation results.

state, where state is required to support the more complex levels of computation beyond basic combinatorial logic.

Finally, hybrid integrated devices are important to enable stable, efficient implementations.

6.1.1 Phase encodings

Phase-based encoding is the only multibit optical representation that supports the operations of a mathematical field with transformations that are continuous, uniform, and unambiguous. The lack of ambiguity is needed to allow operations to have deterministic consequences. Having continuous and uniform transforms means that a single device can process a single operation, without need for conditional (value-dependent) processing.

Phase encoding is attractive for other reasons. It allows use of other encoding dimensions (such as polarization) as independent channels, potentially enabling a single device to process multiple channels concurrently. It also simplifies amplification because all symbols are represented using the same power. Receivers require phase recovery, but this is already mature and efficient.

Binary encodings also support the operations needed to implement a field but are very inefficient for communication. Their use would necessitate the same very high level of aggregation and deaggregation that would be required for OEO conversion and would render an optical computing solution uncompetitive with one using conventional electronics.

6.1.2 Digital encodings

The need for digital encodings is driven by the desire to cascade processing functions and to support state feedback. Analog systems can cascade functions, but only through a limited number of levels due to signal degradation, which constrains the computational complexity and prevents any but the most basic computational model of combinatorial logic.

Regeneration is the key to supporting digital encodings and is the largest impediment to developing effective optical computation. Current methods are complex, consume significant power, and have limited impact. There are a wide variety of approaches to regeneration, but the more efficient and effective methods – such as differential processing – fail to preserve the semantics of the symbol stream, rendering them useless for computation.

6.1.3 Wave mixing

Although phase encodings and digitization are more obviously needed, the need to use wave mixing is less so. It is driven primarily by the need to support field operations on symbols at rates that exceed that of electronics.

Wave mixing has the benefit of being able to support transformational processing transparently to the symbol coding rate. Some functions (i.e. some field operations) are completely transparent, depending only on the frequency over which the symbols are encoded. Others, such as regeneration, can depend on the symbol density (i.e. the number of distinct phase values used for encoding), but remain transparent to the symbol rate.

6.1.4 Need for state

The desire to use digital phase encoding and mixing is not uncommon for optical computation, but there is less consideration for state. The amount of state and the complexity of its state access patterns determine the computational capabilities of the resulting processing system.

A combinatorial system can never be used to identify context-dependent patterns, as are needed for the most basic packet forwarding operations or even basic data filtering beyond exact match. An FSM cannot count or support AI backtracking, so it cannot be used for triggers based on threshold levels, compression, or encryption. A PDA cannot compute recursive functions, as are sometimes needed for hash functions used for authentication.

State is the key to supporting these higher levels of computation. Many optical signal processing systems are only combinatorial, using no state feedback at all. Others have a single state, either as direct feedback or encoded in the device itself; some systems use a single such device, others an array, but all are limited to the capability of an FSM. Most in-transit operations require more than just a single state, e.g. either PDA or TM capabilities. This is why our focus is on seeking a viable approach to an OTM.

The key to supporting state for such a TM is the use of optical delays. Implemented in on-chip waveguides, these may be limited to tens-hundreds of symbols, recirculated with phase restoration. That should be sufficient for many of the kinds of algorithms that are useful for in-transit processing. Additional storage may be possible off-chip, e.g. using board-level fiber "spools". The amount of recirculation is limited by the efficiency of symbol restoration and access to state needs to be coordinated with the symbol's position in the delay line. The latter may require splitters and configurable delays to align stored state with incoming data. These issues are under active investigation by our team.

6.1.5 Need for hybrid integration

The desire to encode data using phase and to use wave mixing drives the need for hybrid nanophotonic integration. Benchtop systems are too difficult to stabilize because the optical paths often create multiple overlapping interferometers. Although a single interferometer can be stabilized, multiple overlapping interferometers cannot.

This extreme phase sensitivity can be avoided through nanoscale integration. Distances between components can be engineered as needed, often within $< 1\%$ of a wavelength (e.g. 14-nm fabrication resolution used to implement 1400-nm-wide devices).

Passive optics can be easily fabricated on a silicon substrate, including filters, couplers/splitters, and vias. Silicon can also be used to implement detectors and some forms of wave mixing. Full processing requires pumps (to support mixing), generation, and modulation, all of which require other substrates. Integration supports stability and efficiency only when all these devices are implemented on a single underlying substrate, to enable environment control (temperature, vibration) and to avoid the coupling loss of chip-to-chip interfaces.

6.1.6 Some additional caveats

This discussion began with an assumption that optical computation should focus on in-transit operations because that might be where it would be most useful. The corollary is that optical computation is likely not useful as an end system alternative to electronics.

Optical processing already requires the use of exotic materials to do anything beyond limited passive processing. It is unfair to compare it to silicon electronics, whose

frequencies have recently plateaued near 4 GHz. A more appropriate comparison would be to GaAs, germanium, or other more exotic materials, which already support switching upwards of 600 GHz.

Fast optical processing, at rates competitive with electronics, requires wave mixing, which, in turn, requires (and consumes) pumps. The power needed for these pumps suggests that optical computation should never be considered as power efficient. There is no optical equivalent of adiabatic (zero heat or friction loss) or reversible (zero energy) computing, as there is for electronics.

As a result, it seems less useful to consider optical processing as a replacement for electronics in the end system, or to seek an optical solution as a power-efficient alternative to an electronic approach.

6.2 Impact of the experiments

The analysis of the basic principles of computation, communication, and optics yields the previous conclusions regarding optical computation. The results of recent experiments have further impact on viable approaches, including the significant impact of carry generation and regeneration as the two most significant challenges.

6.2.1 Computation

Computation relies on two capabilities: the operations of a mathematical field (for combinatorial logic) and support for state recirculation (for all higher levels). Wave mixing already natively supports some of the functions that correspond to the needed field operations, including addition, subtraction, and multiplication. These functions require substantial signal manipulation, including frequency conversion, harmonics generation, conjugate generation, and the need for phase-aligned pump sources.

Carry generation is required to support positional representations. Current approaches to carry generation require a cascade of several wave mixing operations, including phase division (via phase-amplitude offset), symbol addition, phase squeezing, and output normalization. The challenge of these cascaded operations is that regeneration cannot be supported between the first three (division, addition, squeezing) operations.

6.2.2 Regeneration

Regeneration is known as the most significant challenge for optical computation. It is widely understood as needed to support function composition, to cascade processing operations, and to allow single-state feedback (to support the computational complexity of an FSM).

The need for regeneration to support state is also widely appreciated. State is a form of delayed signal recirculation and recirculation is easily seen as amplifying signal distortion. Given regeneration, state can be supported using delay, as was done in the earliest electronic computers.

Regeneration is also critical for carry generation. Carries are inherently a type of digitization, taking a set of symbols and returning either a 0 (no carry) or 1 (carry). In this case, regeneration needs to be even more powerful, collapsing previously distinguishable states.

Efficient regeneration is difficult to achieve. The power needed for phase-sensitive amplification or phase squeezing via wave mixing processes can be substantial. Denser encodings require correspondingly higher harmonics, which are increasingly inefficient to generate. This density-harmonic relationship may place an effective upper bound on the encoding density, requiring some aggregation/deaggregation to support efficient communication, but it may still retain enough symbol density to remain competitive as an alternative to binary electronics.

7 Conclusions

The desire to support in-transit processing of optically encoded information drives the need for digital optical processing. That processing requires computational complexity beyond simple field operations, e.g. combinatorial logic. The need to support state to enable higher levels of computational complexity thus further motivates the search for an OTM.

Overall, for optical computation to move forward, there need to be significant advances in both regeneration and hybrid integration. The need for regeneration is already well understood, but the need for regeneration to support carry generation for positional representations is less so. The need for nanophotonic integration, especially hybrid integration that supports pumps, mixing, and other active devices together with passive components on the same substrate, is widely appreciated; however, the critical need for this hybrid substrate is much less so as well. In particular, hybrid integrated nanophotonic devices with multiple interconnected components needs to become reliably available.

In addition, the limits of regeneration due to the relation between encoding density and the level of

harmonics needed may require all-optical aggregation and deaggregation. This would enable high-density communication for efficient long-distance communication while supporting digital optical computation for in-transit processing.

Ultimately, this work suggests that there is a path toward a feasible OTM that can exceed the limits of electronic computation. Optics may never be as efficient as electronics for end systems; however, if a signal is already optically encoded, the power and design requirements of optical processing may be worth the effort.

Acknowledgments: This work was partly supported by National Science Foundation (NSF) under contract (1344221). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.

References

[1] Touch J, Willner A. Native digital processing for optical networking. In: Proc. IEEE Third Int'l. Conference on Future Generation Communication Technologies (FGCT), 2014.
[2] Touch J, Cao Y, Ziyadi M, et al. A candidate approach for optical in-network computation. Invited paper, IEEE Summer Topicals, 2016.
[3] Green P. An all-optical computer network: lessons learned. IEEE Netw 1992;6:56–60.
[4] Harai H, Murata M. High-speed buffer management for 40 Gb/s-based photonic packet switches. IEEE/ACM Trans Netw 2006;14:191–204.
[5] Jeon M, Pan Z, Cao J, et al. Demonstration of all-optical packet switching routers with optical label swapping and 2R regeneration for scalable optical label switching network application. IEEE/OSA J Lightwave Technol 2003;21:2723.
[6] Touch J, Bannister J, Suryaputra S, Willner A. A design for an Internet router with a digital optical data plane. Invited paper, Photonics West, 2014.
[7] Chowdhury N, Boutaba R. A survey of network virtualization. Comput Netw 2010;54:862–76.
[8] McKeown N, Anderson T, Balakrishnan H, et al. OpenFlow: enabling innovation in campus networks. ACM SIGCOMM Comput Commun Rev 2008;28:38.
[9] Gringeri S, Basch E, Xia T. Technical considerations for supporting data rates beyond 100 Gb/s. IEEE Commun 2012;50:521–30.
[10] Athale R, Psaltis D. Optical computing: past and future. Optics Photon News 2016;27:29–39.
[11] Xu Q, Fattal D, Beausoleil R. Silicon microring resonators with 1.5-μm radius. Optics Express 2008;16:4309–15.
[12] Arrathoon R. Digital optical computing: possibilities and pitfalls. In: Proc. SPIE Real Time Signal Processing IV, V564, 1985, pp. 108–18.
[13] Caulfield H, Dolev S. Why future supercomputing requires optics. Nat Photon 2010;4:261–3.
[14] Jackson D. Photonic processors: a systems approach. Appl Optics 1994;33:5451–66.
[15] Miller D. Correspondence – the role of optics in computing. Nat Photon 2010;4:406.
[16] Feitelson D. Optical computing: a survey for computer scientists. MIT Press, Cambridge, MA, 1992.
[17] Tucker R. Correspondence – the role of optics in computing. Nat Photon 2010;4:405.
[18] Abraham W, Seaton C, Smith S. The optical computer. Sci Am 1983;85–93.
[19] Ambs P. Optical computing: a 60-year adventure. Adv Opt Tech 2010;2010:1–15.
[20] Jain K, Pratt GW Jr. Optical transistor. Appl Phys Lett 1976;28:719–21.
[21] Miller D. Are optical transistors the logical next step? Nat Photon 2010;4:3–4.
[22] Mistry K. Tri-gate transistors: enabling Moore's law at 22nm and beyond. In: Presentation at Semicon West 2014. Available at: http://www.semiconwest.org/sites/semiconwest.org/files/docs/Kaizad%20Mistry_Intel.pdf. Accessed June 2014.
[23] Chattopadhyay T, Roy JN. All-optical quaternary computing and information processing: a promising path. J Optics 2013;42:228–38.
[24] Kakande J, Slavík R, Parmigiani F, et al. Multilevel quantization of optical phase in a novel coherent parametric mixer architecture. Nat Photon 2011;5:748–52.
[25] Slavík R, Parmigiani F, Kakande J, et al. All-optical phase and amplitude regenerator for next-generation telecommunications systems. Nat Photon 2010;4:690–5.
[26] Zhu Z, Funabashi M, Pan Z, Xiang B, Paraschis L, Yoo S. Jitter and amplitude noise accumulations in cascaded all-optical regenerators. IEEE/OSA J Lightwave Technol 2008;26:1640–52.
[27] Yang JY, Akasaka Y, Ziyadi M, et al. PSA and PSA-based optical regeneration for extending the reach of spectrally efficient advanced modulation formats. Invited paper, IEEE Summer Topicals 2015.
[28] Sawchuk A, Strand T. Digital optical computing. Proc. IEEE 1984;72:758–79.
[29] Hardy J, Shamir J. Optics inspired logic architecture. Optics Express 2007;15:150–65.
[30] Huang A. Parallel algorithms for optical digital computers. In: Proc. SPIE Int'l. Optical Computing Conf., April 1983, pp. 13–17.
[31] Paquot Y, Duport F, Smerieri A, et al. Optoelectronic reservoir computing. Sci Rep 2012;2:1–6.
[32] Touch J, Mohajerin-Ariaei A, Chitgarha M, et al. The impact of errors on differential optical processing. USC/ISI Tech Report ISI-TR-690, 2014.
[33] Mamyshev P. All-optical data regeneration based on self-phase modulation effect. In: ECOC 1998.
[34] Striegler A, Schmauss B. All-optical DPSK signal regeneration based on cross-phase modulation. IEEE Photon Technol Lett 2004;16:1083–5.
[35] Hauer M, McGeehan J, Kumar S, et al. Optically-assisted Internet routing using arrays of novel dynamically reconfigurable FBG-based correlators. IEEE/OSA J Lightwave Technol Spec Issue Opt Netw 2003;21:2765–78.
[36] McGeehan J, Kumar S, Gurkan D, et al. All-optical decrementing of a packet's time-to-live (TTL) field and subsequent dropping

of a zero-TTL packet. IEEE/OSA J Lightwave Technol Spec Issue Opt Netw 2003;21:2746–52.
[37] Liu L, Kumar R, Huybrechts K, et al. An ultra-small, low-power, all-optical flip-flop memory on a silicon chip. Nat Photon 2010;4:182–7.
[38] Wang J, Yang JY, Wu X, Yilmaz O, Nuccio S, Willner A. 40-Gbaud/s (120-Gbit/s) octal and 10-Gbaud/s (40-Gbit/s) hexadecimal simultaneous addition and subtraction using 8PSK/16PSK and highly nonlinear fiber. In: OFC 2011.
[39] Wang J, Yang JY, Huang H, Willner A. Three-input optical addition and subtraction of quaternary base numbers. Optics Express 2013;21:488–99.
[40] Mohajerin-Ariaei A, Ziyadi M, Chitgarha M, et al. Phase noise mitigation of QPSK signal utilizing phase-locked multiplexing of signal harmonics and amplitude saturation. Optics Lett 2015;40:3328–31.
[41] Almaiman A, Cao Y, Ziyadi M, et al. Experimental demonstration of phase-sensitive regeneration of a 20–40 Gb/s QPSK channel without phase-locked loop using Brillouin amplification. In: ECOC 2016.
[42] Cao Y, Alishahi F, Akasaka Y, et al. Experimental investigation of quasi-periodic power spectrum in Raman-assisted phase sensitive amplifier for 10/20/50-Gbaud QPSK and 10-Gbaud 16QAM signals. In: ECOC 2016.
[43] Chitgarha MR, Khaleghi S, Ziyadi M, et al. Demonstration of tunable optical generation of higher-order modulation formats using nonlinearities and coherent frequency comb. Optics Lett 2014;39:4915–8.
[44] Ziyadi M, Chitgarha M, Mohajerin-Ariaei A, et al. Optical channel de-aggregator of 30-Gbaud QPSK and 20-Gbaud 8-PSK data using mapping onto constellation axes. Optics Lett 2015;40:4899–902.