Field Trial of a Flexible Real-Time Software-Defined GPU-Based Optical Receiver

Sjoerd van der Heide, *Graduate Student Member, IEEE*, Ruben S. Luis, *Senior Member, IEEE*, Benjamin J. Puttnam, *Member, IEEE*, Georg Rademacher, *Senior Member, IEEE*, Ton Koonen, *Fellow, IEEE*, Satoshi Shinada, *Member, IEEE*, Yohinari Awaji, *Member, IEEE*, Hideaki Furukawa, *Member, IEEE*, and Chigo Okonkwo, *Senior Member, IEEE*

***Abstract*—We introduce a flexible, software-defined real-time multi-modulation format receiver implemented on an off-the-shelf general-purpose graphics processing unit (GPU). The flexible receiver is able to process 2 GBaud 2-, 4-, 8-, and 16-ary pulse-amplitude modulation (PAM) signals as well as 1 GBaud 4-, 16- and 64-ary quadrature amplitude modulation (QAM) signals, with the latter detected using a Kramers–Kronig (KK) coherent receiver. Experimental performance evaluation is shown for back-to-back. In addition, by using the JGN high speed R&D network testbed, performance is evaluated after transmission over 91 km field-deployed optical fiber and reconfigurable optical add-drop multiplexers (ROADMs).**

***Index Terms*—Field trial, GPU, Kramers–Kronig, real-time.**

I. Introduction

With the continual increase in demand for data-traffic at lower cost-per-bit, there is an increased interest in low-cost optical transceivers for data-center interconnects. Multi-vendor standards, e.g., [1], are key to the development and roll-out of these systems. Software-defined transceivers have supported and enhanced the widespread development of 5 G and other wireless communications standards [2]. These systems perform digital signal processing (DSP) wholly [3] or partially [4] using off-the-shelf general purpose hardware, leading to high flexibility, combined with low development effort and rapid turnaround. Therefore, software-defined transceivers are expected to play an increasing role in the rapid development, validation, and test of optical communication standards.

Manuscript received November 23, 2020; revised December 28, 2020; accepted January 5, 2021. Date of publication January 8, 2021; date of current version April 16, 2021. This work was supported by the Dutch NWO Gravitation Program on Research Center for Integrated Nanophotonics under Grant GA 024.002.033. *(Corresponding author: Sjoerd vanderHeide.)*

Sjoerd van der Heide, Ton Koonen, and Chigo Okonkwo are with the High Capacity Optical Transmission Laboratory, Electro-Optical Communications Group, Eindhoven University of Technology, 5600 MB Eindhoven, The Netherlands (e-mail: s.p.v.d.heide@tue.nl; a.m.j.koonen@tue.nl; c.m.okonkwo@tue.nl).

Ruben S. Luis, Benjamin J. Puttnam, Georg Rademacher, Satoshi Shinada, Yohinari Awaji, and Hideaki Furukawa are with the National Institute of Information and Communications Technology, Photonic Network System Laboratory, KOGANEI Corp., Tokyo 184-8795, Japan (e-mail: rluis@nict.go.jp; ben@nict.go.jp; georg.rademacher@nict.go.jp; sshinada@nict.go.jp; yossy@nict.go.jp; furukawa@nict.go.jp).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/JLT.2021.3050304.

Digital Object Identifier 10.1109/JLT.2021.3050304

Whilst commonplace for wireless systems, the development of software-defined transceivers for optical communications has been restricted by energy and computing power limitations. Recently, exploiting field-programmable gate arrays (FPGAs) for real-time DSP for optical communications has been investigated [5]–[7]. GPU-based systems for optical communications are restricted by energy and computing power to prototype and test. With 45% [8] year-on-year growth of computing power and 25% increase [9] in energy efficiency (FLOPS per Watt), general-purpose GPUs have the potential to meet demanding processing requirements. Note that, GPU power efficiency showed a 3-fold improvement over equivalent FPGA for simple highly-parallelized operations [10]. These exponential increases may facilitate GPU use beyond prototyping. Compared to GPUs, FPGAs require longer development times and more stringent resource management to achieve the specific functions required for DSP.

Recently, the use of general-purpose GPUs has been demonstrated for specific functions such as forward error correction (FEC) decoding [11], [12] and physical-layer functions for optical communications [13]–[15]. Additionally, real-time DSP for optical differential quaternary phase-shift-keying (DQPSK) has been implemented on a GPU [16]–[18]. In these papers, massive parallel processing capabilities of GPUs were exploited for processing single-polarization 5 Gbit/s DQPSK signals, correcting for intersymbol interference (ISI) using a finite impulse response (FIR) filter. This approach greatly increases flexibility of optical transceivers. However, there remains the potential to further improve on this concept, since single-polarization coherent systems require real-time polarization control and differential phase-shift keyed modulation does not provide high spectral efficiency.

In this work, we implement a flexible, software-defined real-time multi-modulation format receiver. A full real-time DSP chain is implemented on a commercial, off-the-shelf general-purpose GPU and validated experimentally. The receiver DSP uses massive parallelization to receive PAM-2, -4, -8, and -16 signals at 2 GBaud as well as 4-, 16-, and 64-QAM signals at 1 GBaud, with the latter detected using a KK coherent receiver [19]. All measurements employ identical transmitter and receiver hardware without polarization control. The GPU software is able to switch between modulation formats. To the authors' knowledge, this is the first demonstration of a

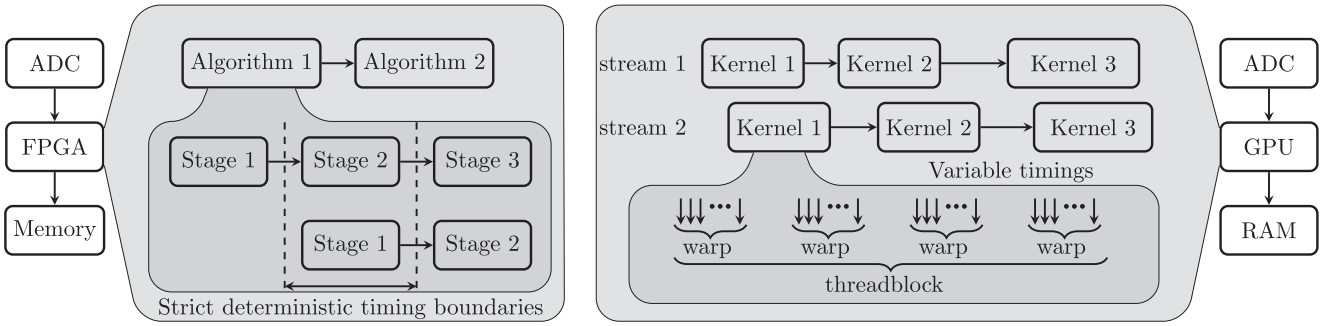


Fig. 1. Comparison between parallel processing on an FPGA (left) and a GPU (right). GPU processing algorithms are implemented in kernels which are executed in order in streams. Many threads in parallel perform the operations defined in the kernel. Threads are grouped in warps and threadblocks. Execution times on FPGAs are deterministic and strictly controlled, whilst timings on GPUs are non-deterministic.

multi-modulation format software-defined GPU-based receiver and the first real-time demonstration of coherent KK detection.

Furthermore, we validate the performance in a 91 km optical fiber link over a field-deployed metropolitan network. The fiber ring is part of the Japan Gigabit Network (JGN) high speed R&D network testbed [20] consisting of 3 commercial ROADMs in 2 separate Tokyo locations. These results demonstrate the potential of software-defined receivers for low-cost optical links, exploiting the exponentially growing computing power of GPUs.

This paper is an extension to the work presented at the European Conference on Optical Communications (ECOC) 2020 [21]. Additional results and a detailed description of the structure of the real-time receiver architecture and the algorithms implemented on the GPU are presented. Clock-recovery for intensity-modulation direct-detection (IMDD) PAM-N signals is shown to tolerate rapid changes in clock-frequency offset and static clock-frequency offsets of up to 30.5 ppm. Using a noise-loading optical setup, 2 GBaud PAM-2, 4, 8 signals are shown to reach the 8.4 dB optical signal-to-noise ratio (OSNR) Q-factor threshold for 6.7% overhead hard decision forward error correction (HDFEC) [22] at 5.6 dB, 14.0 dB, and 22.2 dB, respectively. After transmission through the field trial network, an OSNR penalty of 0.4 dB and 1.5 dB is observed for PAM-2 and PAM-4, respectively. For PAM-8, a 20% overhead HDFEC was necessary since it cannot reach the 6.7% threshold. PAM-16 can be decoded in real time both in back-to-back and after transmission using the GPU DSP, but signal quality is not sufficient to reach either HDFEC threshold. For 1 GBaud KK N-QAM signals, carrier-to-signal power ratio (CSPR) optimization was performed and a CSPR of 6 dB was chosen for 4-QAM and 11 dB for 16- and 64-QAM. 4- and 16-QAM signals reach the 6.7% overhead HDFEC threshold at 5.5 dB and 5.9 dB OSNR, whilst 16-QAM requires an OSNR of 17.6 dB and 19.1 dB for back-to-back and transmission, respectively. 64-QAM signals were processed in real time, but performance was not sufficient to reach either HDFEC threshold. Six second continuous real-time transmission of all modulation formats show stable short-term average Q-factors despite the varying environment of installed fiber.

This paper is structured as follows: Section II introduces GPU processing and the general structure of the real-time GPU receiver architecture. Section III describes the DSP algorithms employed for IMDD PAM-N signals in detail and with performance evaluation the in a back-to-back scenario. Section IV discusses the implementation and back-to-back evaluation for KK N-QAM signals. Section V discusses the evaluation of the real-time receiver evaluated using the experimental field trial network. Finally, Section VI concludes this paper.

II. Real-Time GPU Receiver Architecture

A. Comparison Between FPGA and GPU Processing

Fig. 1 shows the similarities and differences between FPGA and GPU parallel processing architectures. Data are digitized by an ADC, copied to the processing device, and after processing the results are stored in memory. Timings between parallel stages of processing in FPGAs processing are deterministic and strictly controlled. The FPGA operates at a certain clock rate and every stage of processing should fit within the timing parameters imposed by this central clock. Also, each stage of processing is assigned a fixed portion of physical computing hardware. In contrast, execution times on the GPU are not deterministic. Computing hardware is shared for all kernels and a central scheduler assigns computing resources to kernels running in parallel.

B. GPU Processing Terminology

Kernels are highly parallel routines that act upon data in the GPU memory. The *GPU code* of the kernel is performed by *threads* running in parallel. A thread is executed on a GPU core and efficient implementations can use millions of threads. A group of 32 threads is called a warp and is guaranteed to execute simultaneously, which allows for very efficient data exchange between these threads through *warp-level shuffles*, used in this work for certain reduction kernels. A group of warps, called a *threadblock*, is executed on the same *streaming multiprocessor*, which is a group of GPU cores. Threads in a threadblock share physical computing hardware and memory, leading to caching benefits. Multiple threadblocks are not necessarily performed in parallel. This depends on the scheduling by the GPU driver.

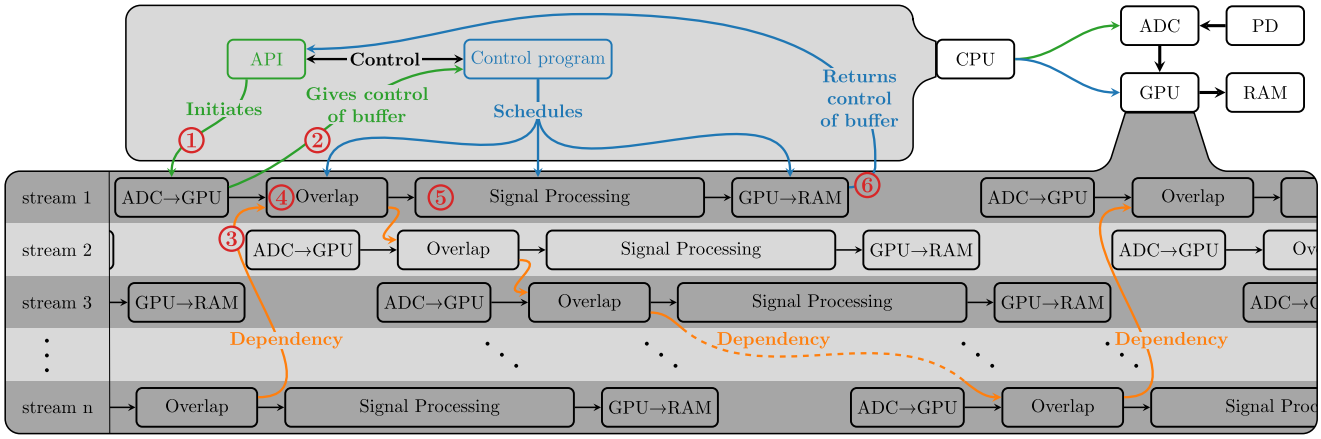


Fig. 2. Real-time GPU Receiver Architecture. The output of the photodiode (PD) is digitized by the analog-to-digital converter (ADC). Samples are processed per buffer in a block-wise fashion in parallel streams on the GPU controlled by the central processing unit (CPU). Detailed descriptions of the *Signal Processing* block can be found in Section III and IV.

Dependencies in the signal processing chain need to be handled appropriately. Kernels in the same processing *stream* are performed in order. Therefore, splitting an algorithm into separate kernels in the same stream can address the dependency. Alternatively, a single threadblock can be employed to perform a certain algorithm, synchronization within a threadblock is possible since it runs the same piece of physical hardware. Kernels in different streams run parallel to each other. In this case, *events* can be used to halt one stream until a certain kernel in another stream has finished processing.

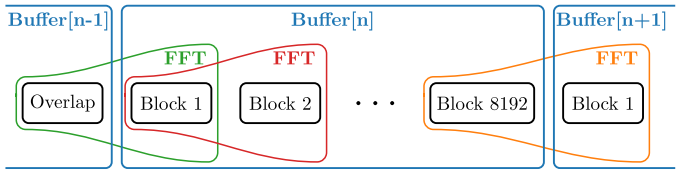


Fig. 3. Relation between buffers and blocks for block-wise processing. Each stream processes one buffer at a time, each consisting of 8192 blocks. Overlap-save Fourier transforms are used to ensure data integrity.

C. Continuous Real-Time Processing Requirements

The real-time GPU receiver consists of a 1GHz photodiode connected to a 12 bit 4 GSa/s ADC. Digitized samples are copied in buffers from the ADC to the GPU where they are processed in a highly parallel manner. Each buffer contains 2^{22} samples, which takes 1.049 ms at 4 GSa/s. In our implementation, each buffer is assigned its own processing stream and any dependencies to ensure data continuity are handled by events. For real-time processing, the buffers need to be processed as fast or faster than they are created by the ADC in order to avoid data loss. As such, the average buffer processing time needs to be lower than 1.049 ms times the number of streams employed. Therefore, buffer processing times can be relaxed by increasing the number of parallel streams at the expense of increased latency.

D. GPU Signal Processing Structure

Figure 2 shows the structure of receiver, the tasks performed by the GPU, and how those are controlled by the CPU. The ADC is controlled by an application programming interface (API), provided by the manufacturer which also manages the data transfer to the GPU. A second program, written by the authors, controls the API and launches signal processing kernels.

- *Step 1:* The ADC digitizes the analog signal into 12-bit digital samples at either 2 (IMDD) or 4 (KK) samples per symbol and temporarily stores them in ADC memory. The API initiates the transfer of a buffer containing 2^{22} samples from ADC memory to GPU memory using direct memory access (DMA), provided a free GPU buffer is available for the API to use. This is marked as Step 1 in Fig. 2. Each buffer is assigned its own stream and DSP kernels are added to that stream to process the data.
- *Step 2:* Control over the GPU buffer which now contains the digitized signal is handed over to the control program written by the authors.
- *Step 3:* For continuous real-time data processing certain overlap between buffers is required, an overlap kernel is used for this. These overlap kernels need to be executed in order and events ensure an overlap kernel cannot start processing until its predecessor is finished. This is shown in Fig. 2 as Step 3 and marked as *Dependency*.
- *Step 4:* The 2^{22} samples in a buffer are subdivided into 8192 blocks of 512 samples for frequency domain (FD) processing as depicted in Fig. 3. FD processing requires one block of overlap between buffers for data continuity. An overlap kernel handles this by prepending a block to the current buffer which was stored elsewhere in memory. Afterwards, it copies the last block of its buffer to memory for the next overlap kernel to use. Also, the overlap kernel converts the data from 12-bit unsigned integers to 32-bit floats.
- *Step 5:* This step contains the actual DSP chain which uses both time domain (TD) and 100% overlap-save FD

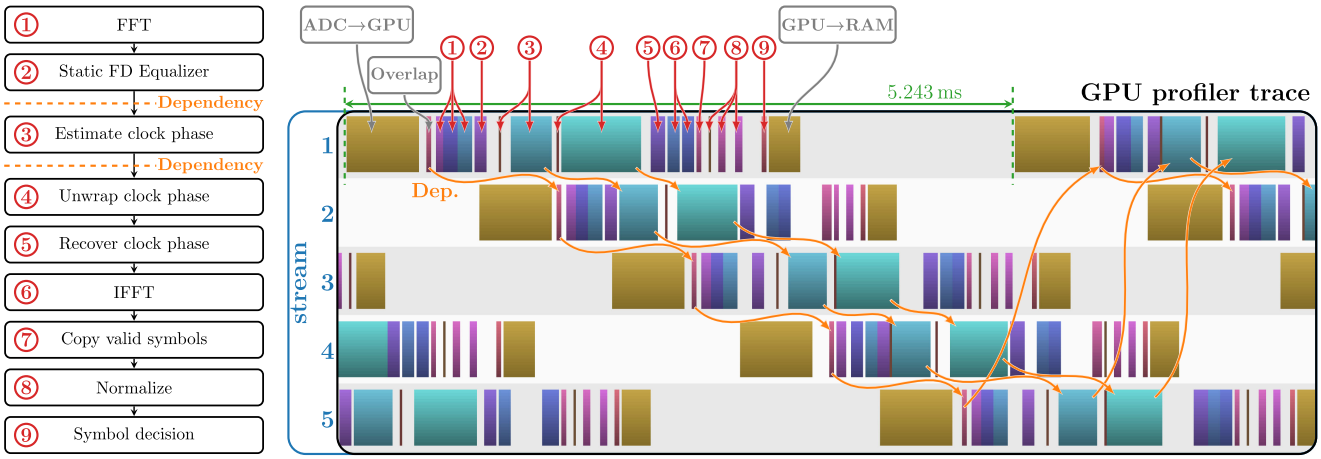


Fig. 4. IMDD GPU signal processing chain including an annotated GPU profiler trace, detailing the *Signal Processing* block of Fig. 2 for PAM-N signals.

processing. This block uses floating point samples as input and produces decoded bits as output. A detailed description can be found in Section III and IV for PAM-N and N-QAM signals, respectively.

- *Step 6:* After processing, the decoded bits are copied to random-access memory (RAM) and control over the buffer is handed back to the API.

III. IMDD GPU Signal Processing Chain

Figure 4 lists the DSP chain for IMDD PAM-N signals. It consists of 9 steps, each of which executed by one or more kernels. To support real-time operation, five parallel streams are used as shown in the annotated profiler trace in Fig. 4. Dependencies between algorithms running in parallel streams are handled through events which are marked as *Dependency* in the DSP chain and shown by the orange arrows in the profiler trace.

A. Step 1 and 2: fast Fourier transform (FFT) and Static Equalization

The IMDD signal processing chain starts after overlap copying. A 100% overlap-save 1024-point FFT at 2 samples-per-symbol is performed using a readily-available highly-parallel GPU FFT implementation. This splits the 2^{22} samples in the buffer into 8192 blocks of 1024 samples, of which 512 are *valid* due to 100% overlap-save. Secondly, static FD equalization is performed to compensate for receiver bandwidth impairments using a pre-computed FIR filter. This filter optimized offline in TD using 503 taps, converted to a 1024-point FD version, and uploaded to the GPU. The number of taps was limited to 503 to prevent introduction of ISI through the cyclic nature of the 1024-point FFT.

To fully appreciate the parallel nature of this processing, we need to look at the number of independent threads working in this one kernel alone. FD equalization requires 512 complex multiplications to be performed for each of the 8192 blocks, Hermitian symmetry allows for the omission of half of the spectrum. To this end, 2^{21} threads are launched, each operating on 2 complex samples (4 32-bit floats) at a time. 128-bit vector loads/stores allow for the 4 floats to be loaded/stored using just a single instruction, increasing memory throughput. These 2^{21} threads can be performed in parallel, exploiting the massive parallel capabilities of the GPU. Fig. 4 shows that during the execution of this kernel in stream 1, marked as step 2, is performed in parallel with a ADC-to-GPU copy in stream 2 and other stages of the signal processing of other buffers in streams 4 and 5. Therefore, parallelization is not only exploited *within* kernels acting on a buffer, but also *between* streams operating on different buffers.

B. Step 3 and 4: Clock-Phase Estimation and Unwrapping

Clock-phase estimation is performed block-wise in FD after static equalization using a technique introduced in [23]. This provides an estimate clock-phase for each block of samples. To improve noise tolerance, these estimates are averaged over 105 blocks. This requires the 52 previous and 52 *future* clock-phase estimates to be known as well. The causality issue of the future estimates is resolved through increased buffering in the *overlap* kernel before actual signal processing starts. The dependency on previous estimates requires the clock-phase estimation of the previous buffer to be completed before the averaging and unwrapping step of the current buffer can be allowed to start. To this end, *events* are used to signal when clock-phase estimation is completed, allowing for the current processing to wait until the previous has completed. Note that only the estimation step has this dependency, the remainder of the signal processing can occur in parallel. The events resolving these dependencies are shown by orange arrows in Fig. 4.

The clock-phase estimates are restricted to 2π. Hence, averaging is performed through vector addition in complex space and subsequent phase unwrapping is required. It is denoted as step 4 in Fig. 4. The phase unwrapping kernel checks whether the current averaged clock-phase differs more than π from the previous. This sequential algorithm is hard to parallelize. To some extent this is done through inter-thread communication using warp-level shuffles. This requires some significant processing time. However, the unwrapping algorithm uses a single warp of 32

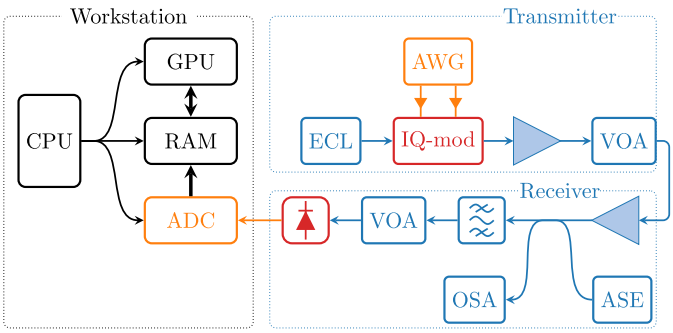


Fig. 5. Experimental noise-loading setup for back-to-back evaluation of the real-time receiver.

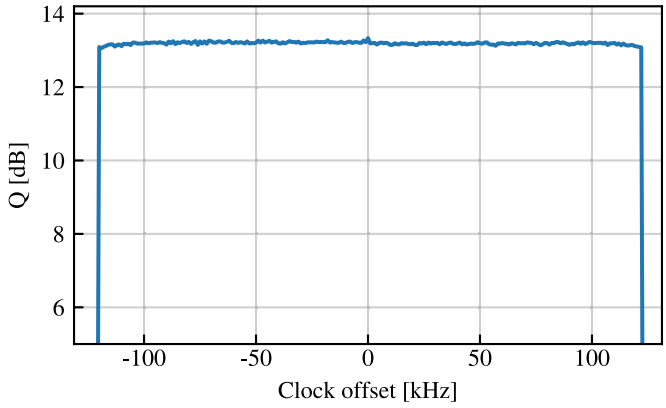


Fig. 6. Q-factor versus clock-frequency offset. The clock-recovery algorithms allows for stable performance across a wide range of clock-frequency offsets.

threads and leaves much of the GPU processing power unused, which can be used by other kernels running in different streams. For example, during the phase unwrapping in stream 1, stream 2 performs an FFT, FD equalization, and clock-phase estimation, stream 3 performs a ADC-to-GPU memory copy, stream 4 is idle, and stream 5 performs normalization, symbol decision, and a GPU-to-RAM copy. Therefore, phase unwrapping does not take up significant amount of *resources*, even though it takes up significant amount of *time*.

C. Step 5-9: Clock Recovery, IFFT, Normalization, and Symbol Decision

Clock recovery is performed by correcting for the unwrapped clock-phase in FD. After the 1024-point inverse fast Fourier transform (IFFT), 256 *valid* symbols need to be extracted for further processing. In the presence of clock-frequency offset, every now and then, either more or fewer symbols may need to be extracted from a block to keep the unwrapped clock-phase within bounds. This is performed in step 7 of Fig. 4, which converts the fixed rate sample input to a variable rate symbol output. Then, buffer-wise normalization is performed using three kernels: initialization, estimation of the DC-offset, and estimation of the amplitude. In the symbol decision kernel, the DC-offset and amplitude are corrected for and PAM-N symbols are decoded into bits. Decision thresholds are optimized offline beforehand and uploaded to the GPU.

D. Experimental Setup for Back-to-Back Evaluation of PAM-N

Figure 5 shows a diagram of the experimental setup for back-to-back characterization of the real-time receiver. At the transmitter, the lightwave from a 500 kHz linewidth external cavity laser (ECL) centered at 1542.92 nm is modulated using a single-polarization in-phase and quadrature modulator (IQM). Electrical driving signals for the IQM are provided by a 2-channel arbitrary-waveform generator (AWG) operating at 12Gs/s amplified by RF-amplifiers, whilst bias-tees and voltage sources control the bias of the modulator arms. PAM-N signals are modulated by biasing one of the IQM-arms to mid-point and driving it with a baseband 2 Gbaud 50% roll-off root-raised-cosine (RRC) pulse-shaped signal.

The receiver consists of an erbium-doped fiber amplifier (EDFA) pre-amplifier followed by a 0.04 nm bandpass filter (BPF). In addition, a noise-loading setup is included with an amplified spontaneous emission (ASE) source and an optical spectrum analyzer (OSA) through a 2×2 coupler. A variable optical attenuator (VOA) is used to control the power at the input of a PD with a 3 dB cut-off frequency of 1GHz. The electrical PD output is directed to the ADC for processing.

E. Experimental Results

The performance of clock-recovery is evaluated using PAM-4 and PAM-8 signals in back-to-back transmission. Fig. 6 shows the Q-factor versus the clock-frequency offset between transmitter and receiver clock when transmitting 2 GBaud PAM-4 signals. An attenuator limited the power into the photodiode to -10 dBm to introduce enough noise and thus bit errors to properly evaluate performance when changing the clock-frequency offset. Performance is stable for a wide range of offsets, showing the resiliance of the implemented algorithms. Performance drops off very rapidly when an offset of 122 kHz (30.5 ppm) or more is applied, which can be attributed an implementation choice to use an 8-bit integer to keep track of number of symbols added or removed throughout the buffer. A change to a 16- or even 32-bit number would greatly increase clock-frequency offset tolerance, but was deemed unnecessary.

Fig. 7 shows clock-frequency offset and Q-factor over time for 2 GBaud PAM-8 signals at 0 dBm input power when using free-running clocks. The transmitter digital-to-analog converter (DAC) is driven by a laboratory-grade tone-generator whilst the ADC uses its own internal clock source. Even when the clock-frequency offset experiences rapid changes as shown in Fig. 7, the Q-factor remains constant, demonstrating that the clock-recovery algorithm is able to cope with these rapid transitions. Since the ADC manufacturer advises against the use of the internal clock, the authors consider this a worst-case test. For the remainder of this work, the ADC received a high-quality clock-signal from a laboratory-grade tone-generator.

Fig. 8 shows the Q-factor as a function of OSNR for PAM-2, PAM-4, PAM-8, and PAM-16. In back-to-back, performance

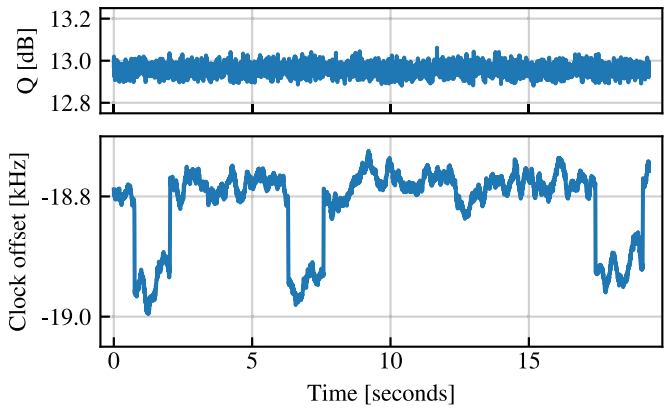


Fig. 7. Clock-frequency offset and Q-factor versus time when using free-running clocks. The clock-recovery algorithm allows for stable performance even when clock-frequency offset changes rapidly.

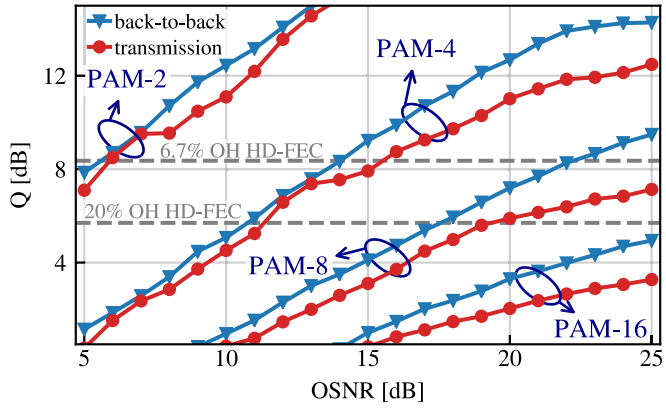


Fig. 8. Q-factor versus OSNR for PAM-N signals, both back-to-back and after transmission over the field trial network.

reaches the 8.4 dB Q-factor threshold for 6.7% overhead HD-FEC [22] at 5.6 dB, 14.0 dB, and 22.2 dB for PAM-2, PAM-4, and PAM-8, respectively. PAM-16 can be decoded in real-time using the GPU DSP, but signal quality is not sufficient to reach the threshold for either 6.7% or 20% [22] overhead HDFEC. Most likely this is due to severe low-pass filtering of the signal by the receiver components. The 2 GBaud signal with 50% RRC roll-off uses 1.5GHz of electrical bandwidth, whilst the 3 dB bandwidth of the photodiode and ADC are both 1GHz. The static equalizer (see Section III and Fig. 4, step 2) can boost the attenuated higher frequencies, but only at the cost of amplifying noise. Future ADCs (PCIe Gen 4) offer greater bandwidth and sampling rate, facilitating greater baud and data rates. Proprietary interfaces such as NVIDIA NVLink [24] can support a further tenfold increase.

IV. KK GPU Implementation and Evaluation

We chose to implement KK field reconstruction to showcase GPU excellence in handling large FFTs and exploiting its enhanced capability for frequency-domain signal processing. Fig. 9 shows the DSP chain for KK N-QAM signals subdivided in 9 steps, each of which an algorithm performed by one or more *kernels* as described in this section. Five parallel streams are used as shown in the profiler trace. Dependencies between streams are marked as *Dependency* and annotated with orange arrows in the profiler trace.

A. Step 1: Overlap and KK Front-End

The KK Front-end containing the square root and logarithm operations are incorporated into the overlap kernel to limit GPU memory access and thus improve performance. The overlap part of this kernel, including the dependency handling via *events*, see Section II-D and Fig. 2.

Since the digitizer used in this experiment was AC-coupled, no DC-terms are measured, hampering KK field reconstruction. Therefore, an offline-optimized static DC-offset is added to the signal [25] after the data are converted from 12-bit unsigned integers to 32-bit floats. Subsequently, a conventional KK front-end [19] performs the square root, to retrieve the signal amplitude, and logarithm, required for phase reconstruction, operations at 4 samples-per-symbol.

B. Step 2-5: Hilbert Transform and KK Field Reconstruction

A 100% overlap-save 1024-point real-to-complex FFT is used to convert the samples pre-processed for phase-retrieval by the KK front-end to frequency domain, dividing the 2^{22} samples in the buffer in 8192 blocks of 1024 samples of which, because of 100% overlap-save, 512 are *valid*. The Hilbert transform is performed in FD before a complex-to-complex IFFT converts back to TD. Now, the KK field reconstruction [19] combines the previously retrieved signal amplitude with the phase recovered through the logarithm and Hilbert transform. The recovered optical field is downshifted to DC for further processing.

C. Step 6-8: FD Static Equalization

After a 1024-point complex-to-complex FFT, the recovered signal is filtered in FD by a static 203-tap FIR filter, which is optimized offline beforehand and uploaded to GPU memory. This static equalizer compensates for receiver bandwidth impairments and performs matched filtering for the RRC N-QAM signals. A 512-point IFFT both converts the signal to TD and downsamples it to 2 samples-per-symbol.

D. Step 9: TD Adaptive Equalization and Symbol Decision

Clock-phase and symbol-phase recovery, transmitter IQ-imbalance compensation, and symbol decision and demapping are performed by a 4-tap adaptive widely-linear [26] TD decision-directed least mean square (DDLMS) equalizer. Note that in contrast to the PAM-N signals of Section III, a 10MHz reference clock was shared by transmitter and receiver, so the equalizer only needs to handle relatively small clock-phase and symbol-phase fluctuations, for example due to changing conditions in the field-deployed fiber. During equalization, the decisions made by the equalizer are demapped and stored in GPU memory to be sent to RAM after this kernel is finished.

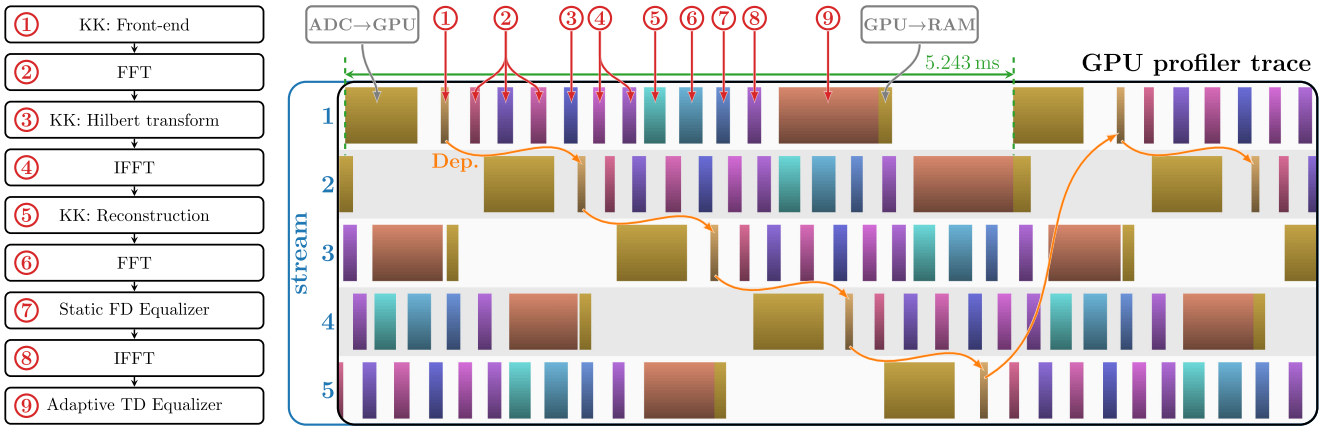


Fig. 9. KK GPU Signal Processing Chain including an annotated GPU profiler trace, detailing the *Signal Processing* block of Fig. 2 for N-QAM signals.

Four taps was deemed sufficient and has the benefit of exploiting 128-bit parallel data access through vector load/store instructions as explained in Section III-A. Furthermore, warp-level shuffles are used to further optimize this TD adaptive equalizer kernel which is serial in nature. One might conclude based on the GPU profiler trace in Fig. 9 that this kernel uses a lot of resources since it uses a lot of time. However, this is not correct. A relatively low amount of GPU parallel processing units are used for execution of this kernel. Therefore, this kernel does not take up significant amount of *resources* even though it takes up significant amount of *time*, similar to the clock-phase unwrapping kernel discussed in Section III-B. The unused parallel processing units can be used by other parallel processing streams, see Fig. 9.

E. Back-to-Back Evaluation of N-QAM Signals

KK N-QAM signals are generated using the same setup as PAM-N signals explained in Section III-D, Fig. 5. However, the IQM is operated at the minimum optical output bias point, whilst the AWG produces baseband 1 GBaud N-QAM signals with 1% roll-off RRC pulse shaping combined with a digitally-introduced carrier tone at 0.547 GHz. The tone power can be chosen to produce the desired CSPR. Note that the N-QAM required optical bandwidth is half of PAM-N, but the required electrical bandwidth is identical.

CSPR optimization is important for KK N-QAM signals since it directly influences the accuracy of signal reconstruction and OSNR performance. When employing high carrier power, signal-signal beat interference (SSBI) is lower and signal reconstruction through the KK algorithm is better, thus improving signal quality after receiver DSP. However, higher carrier power leads to lower signal power for the same combined power. Therefore, signal quality degrades in the higher CSPR region, as can be seen in Fig. 10. The choice of CSPR is essentially a trade-off between increased reconstruction error at lower CSPRs versus increased noise at higher CSPRs. Moreover, the optimal choice also depends on modulation cardinality, since high-cardinality modulation formats such as 64-QAM suffer more from reconstruction errors than 4-QAM. For simplicity

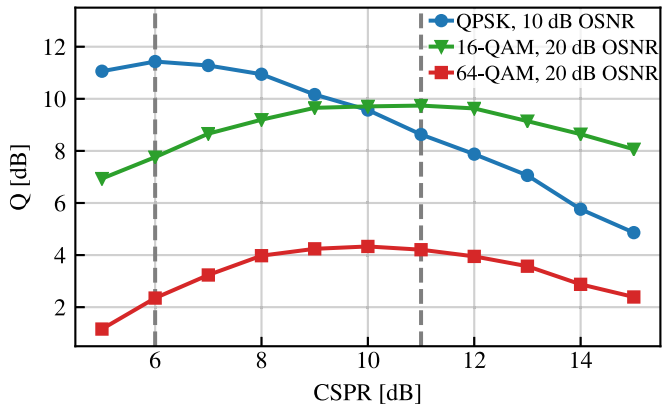


Fig. 10. CSPR optimization shows a trade-off between reconstruction errors at low CSPR and increased noise at high CSPR. 6 dB is chosen for 4-QAM and 11 dB for 16/64-QAM.

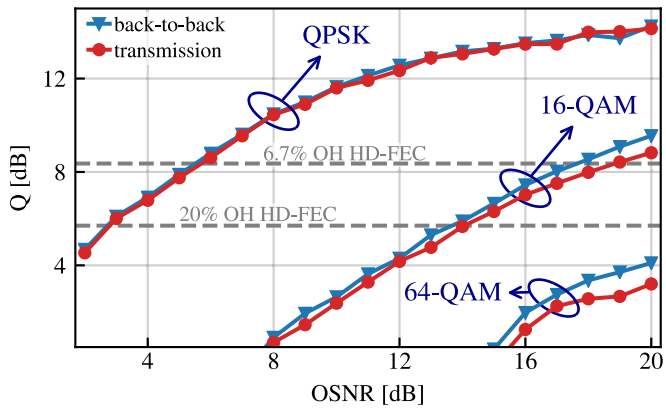


Fig. 11. Q-factor versus OSNR for Kramers-Kronig N-QAM signals, both back-to-back and after field trial transmission.

of measurement, the CSPR is optimized at only one specific value for OSNR, 10 dB for 4-QAM and 20 dB for 16-QAM and 64-QAM. A CSPR of 6 dB is chosen for 4-QAM whilst 11 dB is employed for 16-QAM and 64-QAM throughout this work.

Fig. 11 shows the Q-factor as a function of OSNR for 4, 16, and 64-QAM. 4-QAM reaches the 6.7% overhead HDFEC

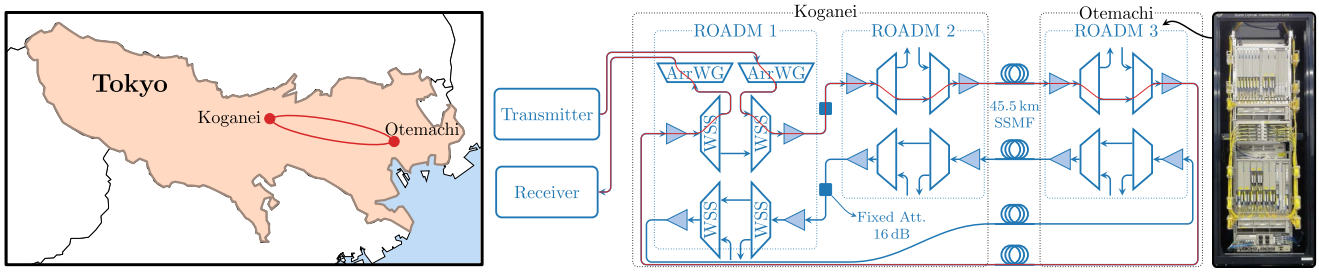


Fig. 12. Experimental setup using field-deployed fiber between Koganei and Otemachi, Tokyo. Transmitter and receiver structure are detailed in Fig. 5.

threshold [22] at 5.5 dB OSNR, whilst 16-QAM requires an OSNR of 17.6 dB. 64-QAM signals were received and processed in real time, however, performance was not sufficient to reach either the 6.7% of 20% overhead HDFEC threshold [22].

V. Experimental Field Trial

A. Experimental Setup

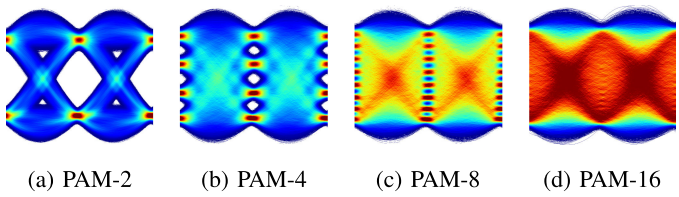
The same transmitter and receiver architecture used for back-to-back characterization, see Fig. 5, is also used to generate and receive the signals in the field trial scenario. The signal launch power is set by an EDFA followed by a VOA. The transmission network shown in Fig. 12 consists of a bidirectional ring with 3 commercial ROADMs. Two ROADMs are installed in the same location in Koganei, Tokyo. The link between these ROADMs is relatively short and its loss was set to 16 dB using fixed attenuators. Both ROADMs are connected to a commercial ROADM in Otemachi, Tokyo by a 45.5 km, 4-fiber link. The transmission loss, including optical distribution frames, is 16.5 dB. 56% of the fiber is installed in underground ducts and the remainder on areal paths and in the surface along railway tracks. The red line in Fig. 12) shows the signal path along the network, with a total transmission distance of 91 km. Each ROADM has two line sides, each consisting of WSSs and optical amplifiers for add/drop and express connections. In addition, (ArrWGs) were used for add and drop. Fig. 12 shows a photograph of one of the commercial ROADMs.

B. Transmission Results of PAM-N Signals

Fig. 8 shows the Q-factor as a function of OSNR for PAM-2, PAM-4, PAM-8, and PAM-16 for back-to-back and after transmission through the field trial network. An OSNR penalty increasing with modulation cardinality, is observed. The penalty at the 6.7% overhead HDFEC threshold is 0.4 dB and 1.5 dB for PAM-2 and PAM-4, respectively. After transmission through the field trial network, PAM-8 cannot be recovered using a 6.7% overhead HDFEC, but can when a 20% overhead HDFEC with a Q-factor threshold of 5.7 dB is employed [22]. Eye diagrams for PAM-N transmission over the field trial network without noise loading are plotted in Fig. 13.

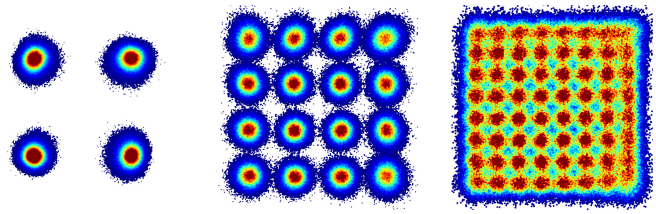
C. Transmission Results of N-QAM Signals

Fig. 11 shows the Q-factor as a function of OSNR for 4-QAM, 16-QAM, 64-QAM for back-to-back and after transmission through the field trial network. A 0.2 dB and 1.2 dB OSNR penalty is observed for 4-QAM and, respectively. 64-QAM signals were received and processed in real time, however, performance was not sufficient to reach either the 6.7% of 20% overhead HDFEC threshold [22]. Constellation diagrams for these modulation formats at maximum available OSNR after transmission over the field trial network are plotted in Fig. 14.



(a) PAM-2 (b) PAM-4 (c) PAM-8 (d) PAM-16

Fig. 13. Eye diagrams for PAM-N transmission over the field trial network without noise loading.



(a) QPSK (b) 16-QAM (c) 64-QAM

Fig. 14. Constellation diagrams for KK N-QAM transmission over the field trial transmission without noise loading.

D. Continuous Real-Time Transmission

Fig. 15 shows the short-term average Q-factor for six second long traces for various modulation formats. Within these six seconds, all transmitted symbols were received, processed, and recorded continuously, using the real-time GPU algorithms detailed in the previous sections. During these six seconds, 6 billion symbols were received for N-QAM signals and 12 billion symbols for PAM-N signals. The Q-factor displayed in Fig. 15 is estimated from the bit error rate (BER) in sections of 21 ms. For all transmitted signals, we observe stable performance. No errors are observed while transmitting PAM-2. 4-QAM, PAM-4, and 16-QAM can be recovered using a 6.7% overhead HDFEC since the Q-factors are 14.1 dB, 13.4 dB, and 10.4 dB, respectively. With a Q-factor of 7.4 dB, PAM-8 cannot be recovered by the 6.7% overhead HDFEC but performance is sufficient for 20% overhead error coding. 64-QAM is successfully transmitted,

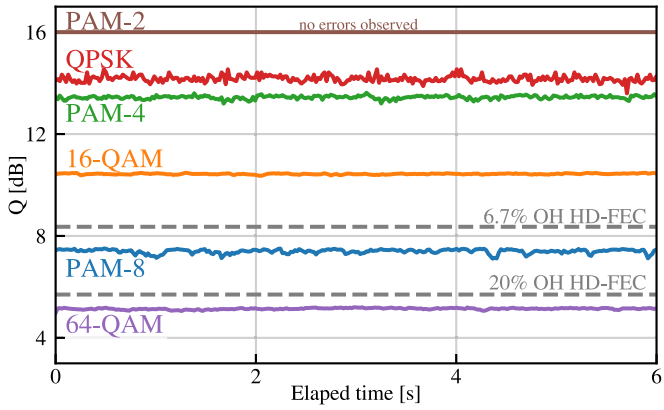


Fig. 15. Continuous real-time transmission traces for various modulation formats. Performance remains stable despite the varying environment of installed fiber.

received, processed using the GPU, and stored in RAM in real time, but performance is not sufficient for HDFEC algorithms considered.

VI. Conclusion

A real-time, software-defined, multi-modulation-format, GPU-based receiver achitecture is introduced, detailed, and demonstrated to achieve stable real-time operation over a field-deployed metropolitan network. We show the potential for massive parallel processing provided by a GPU to recover directly-detected PAM-N signals as well as N-QAM signals with Kramers-Kronig coherent detection. 2 GBaud optical signals using PAM-2, PAM-4, PAM-8, and PAM-16 modulation, 1 GBaud 4-QAM, 16-QAM, and 64-QAM modulation, are received and processed in real time by our flexible receiver architecture. PAM-2 and -4 and 4- and 16-QAM reach the Q-factor threshold for a 6.7% overhead HDFEC both in back-to-back and after transmission through the field-trial network. PAM-8 reaches this threshold in back-to-back, but no longer after transmission, although it can be received using a 20% overhead HDFEC. PAM-16 and 64-QAM are received and processed in real-time, but performance is not sufficient to reach either HDFEC threshold. Continuous real-time transmission reveals stable performance despite the varying environment of installed fiber. These results show the potential of massive parallel processing provided by GPUs for low-cost flexible optical links for a range of modulation formats.

References

[1] OIF, "OIF-400ZR Implementation Agreement." [Online]. Available: https://www.oiforum.com/technical-work/implementation-agreements-ias/

[2] G. Kakkavas *et al.*, "A software defined radio cross-layer resource allocation approach for cognitive radio networks: From theory to practice," *IEEE Trans. Cogn. Commun. Netw.*, vol. 6, no. 2, pp. 740–755, Jun. 2020.

[3] "GNU Radio," Accessed: Nov 23, 2020. [Online]. Available: https://www.gnuradio.org,

[4] T. Kazaz *et al.*, "Hardware Accelerated SDR Platform for Adaptive Air Interfaces," 2017.

[5] S. Randel *et al.*, "First real-time coherent MIMO-DSP for six coupled mode transmission," in *Proc. IEEE Photon. Conf.*, 2015, pp. 1–2, doi: 10.1109/IPCon.2015.7323761.

[6] S. Beppu *et al.*, "Real-time strongly-coupled 4-core fiber transmission," in *Proc. Opt. Fiber Commun. Conf.*, 2020, Paper Th 3H.2.

[7] S. Beppu *et al.*, "Weakly coupled 10-mode-division multiplexed transmission over 48-km few-mode fibers with real-time coherent MIMO receivers," *Opt. Express*, vol. 28, no. 13, pp. 19 655–19 668, Jun. 2020.

[8] P. J. Winzer and D. T. Neilson, "From scaling disparities to integrated parallelism: A decathlon for a decade," *J. Lightw. Technol.*, vol. 35, no. 5, pp. 1099–1115, Mar. 2017.

[9] Y. Sun *et al.*, "Summarizing CPU and GPU design trends with product data," 2019, *arXiv:1911.11313*.

[10] M. Qasaimeh *et al.*, "Comparing energy efficiency of CPU, GPU and FPGA implementations for vision kernels," in *Proc. IEEE Int. Conf. Embedded Softw. Syst.*, 2019, pp. 1–8, doi: 10.1109/ICESS.2019.8782524.

[11] R. Li, J. Zhou, Y. Dou, S. Guo, D. Zou, and S. Wang, "A multi-standard efficient column-layered LDPC decoder for software defined radio on GPUs," in *Proc. IEEE 14th Workshop Signal Process. Adv. Wireless Commun.*, 2013, pp. 724–728.

[12] T. Suzuki, S. Kim, J. Kani, T. Hanawa, K. Suzuki, and A. Otaka, "Demonstration of 10-Gbps real-time reed-solomon decoding using GPU direct transfer and kernel scheduling for flexible access systems," *J. Lightw. Technol.*, vol. 36, no. 10, pp. 1875–1881, 2018.

[13] T. Suzuki, S. Kim, J. Kani, A. Otaka, and T. Hanawa, "10-Gb/s software implementation of burst-frame synchronization using array-access bitshift and dual-stage detection for flexible access systems," *J. Lightw. Technol.*, vol. 36, no. 23, pp. 5656–5662, 2018.

[14] T. Suzuki *et al.*, "Software implementation of 10G-EPON upstream physical-layer processing for flexible access systems," *J. Lightw. Technol.*, vol. 37, no. 6, pp. 1631–1637, Mar. 2019.

[15] T. Suzuki, S. Kim, J. Kani, and J. Terada, "Demonstration of fully softwarized 10G-EPON PHY processing on a general-purpose server for flexible access systems," *J. Lightw. Technol.*, vol. 38, no. 4, pp. 777–783, 2020.

[16] S. Kim, T. Suzuki, J. Kani, A. Otaka, and T. Hanawa, "Coherent receiver DSP implemented on a general-purpose server for full software-defined optical access," in *Proc. Opt. Fiber Commun. Conf. Expo.*, 2018, pp. 1–3.

[17] S. Kim, T. Suzuki, J. Kani, and A. Otaka, "Coherent receiver DSP implemented on a general-purpose server for a full software-defined access system," *IEEE/OSA J. Opt. Commun. Netw.*, vol. 11, no. 1, pp. A 96–A102, Jan. 2019.

[18] T. Suzuki *et al.*, "Real-time implementation of coherent receiver DSP adopting stream split assignment on GPU for flexible optical access systems," *J. Lightw. Technol.*, vol. 38, no. 3, pp. 668–675, Feb. 2020.

[19] A. Mecozzi, C. Antonelli, and M. Shtaif, "Kramers kronig coherent receiver," *Optica*, vol. 3, no. 11, Nov. 2016, Art. no. 1220.

[20] "Japan Gigabit Netw. (JGN)," Accessed: Nov. 23, 2020. [Online]. Available: https://testbed.nict.go.jp/jgn/

[21] S. van der Heide *et al.*, "Real-time, software-defined, GPU-Based receiver field trial," 2020, *arXiv:2010.14333*.

[22] E. Agrell and M. Secondini, "Information-theoretic tools for optical communications engineers," in *Proc. IEEE Photon. Conf.*, 2018, pp. 1–5, doi: 10.1109/IPCon.2018.8527126.

[23] K.-T. Wu and H. Sun, "Frequency-domain clock phase detector for nyquist WDM systems," in *Proc. Opt. Fiber Commun. Conf.*, 2014, Paper Th 3E.2.

[24] NVIDIA, "*NVIDIA A100 Tensor Core GPU Architecture*," Accessed: Dec 28, 2020. [Online]. Available: https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/nvidia-ampere-architecture-whitepaper.pdf

[25] R. S. Luís, G. Rademacher, B. J. Puttnam, C. Antonelli, S. Shinada, and H. Furukawa, "Simple method for optimizing the DC bias of Kramers–Kronig receivers based on AC-coupled photodetectors," *Opt. Express*, vol. 28, no. 3, pp. 4067–4075, 2020.

[26] E. P. da Silva and D. Zibar, "Widely linear equalization for IQ imbalance and skew compensation in optical coherent receivers," *J. Lightw. Technol.*, vol. 34, no. 15, pp. 3577–3586, 2016.

Sjoerd van der Heide (Graduate Student Member, IEEE) was born in 's-Hertogenbosch, The Netherlands, in 1992. He received the B.Sc. and M.Sc. *(cum laude)* degrees in electrical engineering in 2015 and 2017, respectively, from the Eindhoven University of Technology, Eindhoven, The Netherlands, where he is currently working toward the Ph.D. degree with Electro-Optical Communications Group, High Capacity Optical Transmission Laboratory. His research interests include space-division multiplexing and digital signal processing. He was the recipient of the Student Paper Award at ECOC 2018 and the Best Paper Award at OECC 2019.

Ruben S. Luis (Senior Member, IEEE) received the B.Sc. and M.Sc. degrees in electrical and computer engineering from Instituto Superior Tecnico, Technical University of Lisbon, Portugal, in 2000 and 2003, respectively, and the Ph.D. degree in electrical engineering from the University of Aveiro, Portugal, in 2007. Since 2016, he has been with the Photonic Network System Laboratory, National Institute of Information and Communications Technology, Tokyo, Japan, where he researches advanced coherent transmission systems using spatial division multiplexing and optical packet switching. Dr. Luis is a Senior Member of the Photonics Society.

Benjamin J. Puttnam (Member, IEEE) received the M.Phys. degree in physics from the University of Manchester, Manchester, U.K., in 2000 and the Ph.D. degree from University College London, London, U.K., in 2008. He was a Switch Design Engineer with T-mobile, U.K., in between. He is currently a Senior Researcher with Photonic Network System Laboratory, National Institute of Information and Communications Technology (NICT), Tokyo, Japan. After short-term visits to NICT, he was supported by JSPS and the Photonics Group with Chalmers University, Gothenburg, Sweden, and the Ericsson Research Foundation. In 2010, he rejoined NICT. His research interests include space-division multiplexing for optical transmission and optical signal processing.

Georg Rademacher (Senior Member, IEEE) received the Dipl.-Ing. and Dr.-Ing. degrees in electrical engineering from Technische Universitt Berlin, Berlin, Germany, in 2011 and 2015, respectively. During his doctoral studies, he did internships with Bell Laboratories, Holmdel, NJ, USA, and the National Institute of Information and Communications Technology (NICT), Tokyo, Japan. In 2016, he joined Photonic Network System Laboratory, NICT, where he is currently engaged in the research on subsystems and systems for efficient high-capacity optical transmission.

Ton Koonen (Fellow IEEE) is a Full Professor with the Eindhoven University of Technology (TU/e), Eindhoven, The Netherlands, since 2001. Since 2004, he is the Chairman of the group Electro-Optical Communication Systems and from 2012 to 2020, he was the Vice-Dean of the Department of Electrical Engineering. From 2016 to 2019, he was also the Scientific Director of the Institute for Photonic Integration with TU/e. Before 2001, he worked for more than 20 years in applied research in industry, Bell Labs-Lucent Technologies being one of them. He is a Bell Labs Fellow (1998), OSA Fellow (2013), and Distinguished Guest Professor of Hunan University, Changsha, China (2014). In 2012, he received the prestigious Advanced Investigator Grant of the European Research Council on optical wireless communication. His current research interests are optical fiber-supported in-building networks (including optical wireless communication techniques, radio-over-fiber techniques, and high-capacity plastic optical fiber techniques), optical access networks, and spatial division multiplexed systems.

Satoshi Shinada (Member, IEEE) received the B.S. degree from the Science University of Tokyo, Tokyo, Japan, in 1998, and the M.E. and Ph.D. degrees from the Tokyo Institute of Technology, Tokyo, Japan, in 2000 and 2002, respectively. In 2002, he joined Precision and Intelligence Laboratory, Tokyo Institute of Technology, as a JSPS Postdoctoral Fellow. Since 2003, he has been with the National Institute of Information and Communications Technology (NICT), Tokyo. From 2015 to 2016, he was the Deputy Director at the Ministry of Internal Affairs and Communications, Japan. He has been engaged in the research and development on $LiNbO_3$ optical modulators, optical switches, optical interfaces for single flux quantum circuit, and optical packet switching systems. He was the recipient of the IEEE/LEOS Student Award in 2002 and the 2015 Ichimura Prize in Science for Excellent Achievement from the New Technology Development Foundation. He is a member of the IEEE Photonics Society, the Japan Society of Applied Physics, and the Institute of Electronics, Information and Communication Engineers of Japan.

Yohinari Awaji, biography not at the avilable time of publication

Hideaki Furukawa, biography not avilable at the time of publication

Chigo Okonkwo (Senior Member, IEEE) was born in Wakefield, U.K., in 1979. He received the Ph.D. degree in optical signal processing from the University of Essex, Colchester, U.K., in 2010. Between 2003 and 2009, he was a Senior Researcher with Photonic Networks Research Lab, University of Essex, U.K. After the Ph.D. degree, he was appointed as a Senior Researcher with the Electro-Optical Communications Group, working on digital signal processing techniques and the development of space-division multiplexed transmission (SDM) systems. He is currently an Associate Professor and leads the High-Capacity Optical Transmission Laboratory, Institute for Photonic Integration (former COBRA), Department of Electrical Engineering, Eindhoven University of Technology, Eindhoven, The Netherlands. He was instrumental to the delivery of the first major SDM project in the European Union-MODEGAP project. Since 2014, he has been tenured with the ECO Group, where he has built up a world-class laboratory collaborating with several industrial and academic partners. His research interests include optical and digital signal processing, space-division multiplexing techniques, and long-haul transmission techniques. In 2018, he was the TPC Chair for subcommittee three on digital signal handling. Between 2015 and 2017, he was on the TPC for the *OSA Conference on Signal Processing in Photonic Communications (SPPCom)*. In 2017 and 2018, he was the Program Chair and the Conference Chair at *SPPCom*. He was an Associate Editor for special edition of the IEEE JOURNAL ON LIGHTWAVE TECHNOLOGY. From 2020 to 2022, he has been retained on Technical Programme Subcommittee on Fiber-Optic and Waveguide Devices and Sensors (subcommittee D5) at Optical Fiber Communications Conference.