A low-latency real-time PAM-4 receiver enabled by deep-parallel technique

Optics Communications 508 (2022) 127836

Contents lists available at ScienceDirect

Optics Communications

journal homepage: www.elsevier.com/locate/optcom

A low-latency real-time PAM-4 receiver enabled by deep-parallel technique ☆

Liuyan Chen *, Chao Li, Chin Wan Oh, A.M.J. (Ton) Koonen

Department of Electrical Engineering, Eindhoven University of Technology, 5600 MB, Eindhoven, The Netherlands

ARTICLE INFO

Keywords:
Low-latency
Deep-parallel
Real-time
IM/DD
PAM-4

ABSTRACT

High-speed photonic networks using digital signal processing (DSP) techniques are flourishing nowadays to meet the high-bandwidth requirements of modern bandwidth-thirsty applications in a cost-effective manner. However, the additional latency introduced by DSP is hindering the latency-critical applications. In this paper, a FPGA-based real-time low-latency four-level pulse amplitude modulation (PAM-4) receiver including digital adaptive equalization (DAE) is designed and implemented by using a latency-reducing parallel architecture. The DSP-introduced latency in the receiver end is analyzed in detail. As for DAE parallel implementation, a novel re-allocation scheme is proposed to cope with the issue of the dependency of the output on the successive input samples, and a look-ahead computation technique is introduced to improve the adaptive update efficiency. A real-time PAM-4 receiver is demonstrated in an experimental fiber link with 2.5 Gbit/s data rate for the performance evaluation. Compared with offline processing with MATLAB, the BER performance has little deterioration at 7% FEC limit of 1×10^{-3}. With the help of the proposed deep-parallel technique, the DSP-introduced latency is reduced to 0.4 μs on average, which better meets the requirements of latency-sensitive user cases in 5G networks. Furthermore, the real-time PAM-4 receiver could be flexibly reconfigured for various scenarios with low-latency requirements, and the latency-efficient parallel technique as well as the latency analysis method can also be extended to high-speed hardware implementation for data rates up to 100 Gbit/s or more.

1. Introduction

Plenty of emerging bandwidth-hungry applications such as cloud computing, virtual/augmented reality (VR/AR), and three-dimensional (3D) holographic display are greatly challenging the prospective transmission network. For example, transmission of high-performance digital holographic 3D videos over a network occupies a data bandwidth of above 100 Gbit/s [1], which predicts the required bandwidth would breakthrough Tb/s in the near future. To satisfy the enormous bandwidth demand, optical link becomes a promising solution for the high-speed short-reach transmission system, benefiting from its immense bandwidth. Short-reach optical link solutions have been widely investigated in mobile front-haul, data center interconnect, metro, access, and indoor networks. The requirement on low-cost and low power consumption is leading to a revival of intensity modulation with direct detection (IM/DD) scheme. Meanwhile, advanced modulation format in conjunction with digital signal processing (DSP) techniques can improve the spectrum efficiency and signal quality, which will further boost the transmission data rate with required system bandwidth [2]. Four-level pulse amplitude modulation (PAM-4) is one of the most attractive modulation formats in short-reach optical links [3], which has already been ratified by IEEE 802.3 bs for 400 Gbit/s Ethernet transmission [4]. Together with digital equalization techniques, higher data rate could be achieved in a cost-effective manner. X. Tang et al. [5] demonstrated a 50 Gbit/s PAM-4 transmission system using a 10-GHz class optical transmitter and efficient equalization techniques, with no need for expensive ultra-wideband components, optical amplifier or compensation module. Nevertheless, DSP will introduce additional latency for the real-time transmission link, which may be inadequate for the latency-critical applications. For example, in the user case of robotics and telepresence which requires remote-control with real-time synchronous visual-haptic feedback, the system response time should be less than a few milliseconds including network delays [6]. To support such ultra-low latency scenarios, the fifth-generation (5G) network is required to provide an end-to-end latency in the order of 1 ms. However, this latency budget is mainly allocated to transport and switching [7]. A large DSP-introduced latency compresses the latency headroom of other parts during design of the network infrastructure.

Some advanced DSP algorithms have been investigated for PAM4-based short reach fiber communication systems with off-line experimental demonstrations [5,8–11]. The BER performance of these systems is well studied. However, latency is not taken into consideration, since off-line DSP algorithms are utilized based on the assumption of

☆ This work is financially supported by ERC Proof-of-Concept project BROWSE+, Netherlands and Guangzhou Elite Project, China.
* Corresponding author.
E-mail address: L.Chen.1@tue.nl (L. Chen).

https://doi.org/10.1016/j.optcom.2021.127836
Received 21 July 2021; Received in revised form 7 December 2021; Accepted 18 December 2021
Available online 24 December 2021

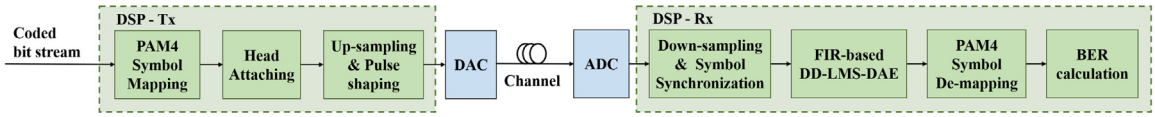

Fig. 1. Block diagram of a digital PAM-4 transmission system. DAC: digital-to-analog converter, ADC: analog-to-digital converter.

zero processing latency and one-symbol feedback loop latency. This approach is not practical for hardware implementation. The achieved system performance including BER and latency could only be considered together as an upper bound for the real implementation. Hence, hardware realization of the DSP algorithms is necessary in the algorithm feasibility verification and system performance evaluation. Advanced DSP algorithms implemented on hardware have been presented to improve the system performance in the real-time transmission experiments [12–18]. However, few literature have analyzed the latency of DSP algorithms implementation in detail. Here, we conduct a detailed analysis for the latency of hardware-realized DSP algorithms and further propose a deep-parallel technique to reduce the total DSP-introduced latency.

For hardware realization of DSP algorithms, field programmable gate array (FPGA) is employed since the dynamic reconfiguration property of FPGA can offer advanced estimation for the DSP algorithms. However, the operation frequency of FPGA is intrinsically limited to a few hundred MHz, which is severely below the required data rate of Gbit/s scale. Thus, parallel implementation of the DSP algorithms is necessary. Meanwhile, for the bandwidth-limited IM/DD systems, digital adaptive equalization (DAE) is usually employed for real-time channel estimation and inter-symbol-interference (ISI) cancellation. It is also challenging for parallel implementation on FPGA because of its successive input samples and iterative adaptive update scheme. FPGA-based architectures of least mean square (LMS), QR decomposition-based recursive least squares (QRD-RLS) algorithms have been proposed for real-time DAE implementation [19,20]. Regarding the parallel implementation of DAE, a fundamental digital static equalizer based on the well-known finite impulse response (FIR) structure is considered. A common parallel technique for the FIR filter is the polyphase decomposition [21]. The key point of this technique is to decompose a N-taps FIR filter into a set of P sub-filters with N/P-taps for parallel processing. In contrast to this theoretical approach, K. Maragos et al. proposed a high-speed Feed-Forward Equalizer (FFE) based on a custom multi-level parallel approach [22]. This method provides a practical and flexible solution for higher parallelism to achieve high data throughput. A coefficient-based strategy is introduced to cope with the problem of successive input samples of FFE. However, the coefficients of the FFE are static, which needs to be pre-estimated in an off-line way.

In this paper, we propose a latency-efficient parallel architecture to develop and implement a low-latency high-throughput PAM-4 receiver on a FPGA platform. A novel data re-allocation scheme is also proposed for deep-parallel processing. Different from the coefficient-based strategy in [22], the proposed data re-allocation scheme enables the utilization of identical parallel equalization blocks to be reused easily. Moreover, a look-ahead computation method [23,24] is employed to efficiently update the coefficients of DAE in parallel. Two parallel architectures with different parallel depths are compared, and the DSP-introduced latency is analyzed in detail. With the help of our proposed deep-parallel technique, the DSP-introduced latency is reduced to 0.13% of the classical straightforward parallel architecture, achieving the average 0.4 μs. The implemented real-time PAM-4 receiver is demonstrated in an optical fiber transmission experiment with a data rate of 2.5 Gbit/s, which reveals that the BER performance has little deterioration compared with offline processing. Our proposed method is also feasible to implement on real-time high-speed communication systems with data rate of up to 100 Gbit/s aiming to pursue low-latency potentially.

The remainder of this paper is organized as follows. In Section 2, DSP algorithms for parallel implementation and their potential problems are stated. Section 3 describes the detailed hardware implementation and analyzes the latency of two parallel architectures with different parallel depths. In Section 4, the experimental demonstration of the implemented low-latency real-time PAM-4 receiver in an optical fiber transmission link is presented. The conclusion is drawn in Section 5.

2. DSP algorithms for parallelization

The general block diagram of a digital PAM-4 transceiver with digital equalization is showed in Fig. 1. At the transmitter side, the coded bit stream is firstly mapped to PAM-4 symbol. Then symbol synchronization header and training sequence are attached at the beginning of the symbol stream. After up-sampling and pulse shaping, the data stream is converted to analog signal using a digital to analog converter (DAC). As for the receiver side, an analog to digital converter (ADC) is firstly used to sample and quantize the received analog signal to digital signal. In order to improve acquisition accuracy of the symbol stream, down-sampling and symbol synchronization are utilized. Then digital equalization is used to track the variance of the channel and reduce ISI in real time. Finally, the equalized and recovered symbol stream is de-mapped to bit stream for error counting.

DAE algorithm, which is usually used for channel estimation, is the main bottleneck for parallel hardware implementation. In general, DAE in frequency domain (FD) is less latency-efficient than that in time domain (TD) especially when the Fast Fourier Transform (FFT) block size of FD-DAE is large. So, for latency-critical application, TD-DAE is preferred. Moreover, for the convenience of parallel processing, an algorithm should provide the possibility to make the processing modules work independently from each other. FIR filter structure is a more appropriate choice than the infinite impulse response (IIR) filter solution. This is because each output of IIR filter depends on the calculated results at the same time in other parallel modules, which is a barrier for independent processing module design. Therefore, FIR-based TD-DAE is chosen for developing the low-latency receiver in this work. The operation of FIR-based DAE is described as

$$y(n) = \sum_{i=0}^{N-1} c_i \cdot x(n-i), 0 \le i \le N-1 \tag{1}$$

where $y(n)$ is the output signal. $x(n)$ is the input signal. N and c_i are the filter length and the filter coefficient of the ith tap. The filter coefficients are updated adaptively by LMS algorithm benefiting from its simplicity, computational efficiency, and good performance under a variety of operating conditions [25]. The adaptive updating scheme of LMS is expressed as

$$c(n+1) = c(n) + 2 \cdot \mu \cdot e(n) \cdot x^*(n) \tag{2}$$

$$e(n) = d(n) - y(n) \tag{3}$$

where μ is the updating step, which affects the updating speed and converging accuracy. $e(n)$ is the error between the equalized output signal and the reference signal $d(n)$. From another perspective, the error signal indicates the distant vector from the current result towards the desired optimal point. In the error calculation procedure, the reference signal $d(n)$ is set as the training sequence initially and then switched to the recovered symbol stream when the error is smaller than a designed threshold. A training sequence (TS) is used to assist the pre-convergence of algorithm at startup, and then a decision-directed LMS (DD-LMS) method is employed to track the remaining error. To implement the DD-LMS-DAE algorithm in parallel, two challenges need to be addressed.

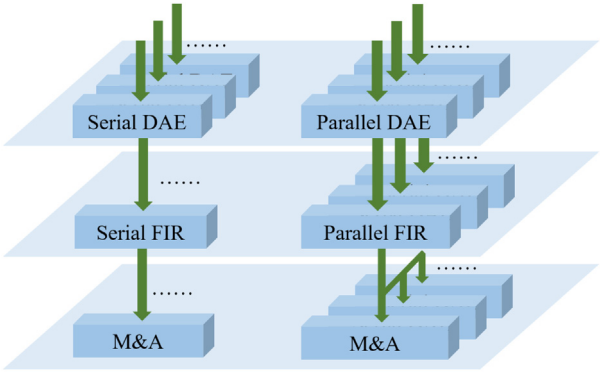

Fig. 2. Parallelism of algorithms with different depths.

2.1. Problem of output dependency resulting from the successive input

Based on Eq. (1), each output signal is calculated based on N successive input signals. It means that every output signal depends on the information of other parallel modules in direct parallel implementation. For example, assuming that 4 lanes are paralleled and 3 taps are used for equalization, then $y(n)$ relies on not only the signal $x(n)$, but also the signals $x(n-1)$ and $x(n-2)$ from neighboring parallel modules. Consequently, in order to maintain the calculation continuity, the successive input signals should not be directly separated and fed to the independent parallel processing modules. Hence, the first challenge for parallel implementation of the DD-LMS-DAE is how to make the processing modules work independently for parallel processing without breaking the calculation continuity.

2.2. Problem of iterative adaptive updating scheme

According to Eqs. (2) and (3), the filter coefficients are updated step by step. Each new set of coefficients at time $n+1$ relies on the information of signal $e(n)$ at time n. However, for direct parallel implementation, the updating interval between the new set of coefficients and the old ones could not maintain 1, since several parallel error signals $e(n)$ are generated at one moment and predicted several different distant vectors towards the optimal point simultaneously. A large updating interval makes the updating procedure less efficient. For example, if 4 parallel error signals $e(n)$ are generated at time n and only one could be used to predict the new set of coefficients at time $n+1$. Then, the updating interval is 4, and the valid information of the other 3 error signals are neglected, which could lead to a slower convergence. Therefore, the second challenge for parallel implementation of the DD-LMS-DAE is how to update the coefficients efficiently in parallel.

3. Hardware implementation and latency analysis

In terms of hardware implementation, the algorithms parallelism can be realized on different levels, as shown in Fig. 2. For a classical straightforward parallel architecture, parallelism is only implemented on top level, while parallelism from top to down is performed in the proposed deeply parallel architecture. In this section, these two implemented parallel architectures are compared. A step-by-step description of the hardware implementation about the DD-LMS-DAE-based receiver is provided in detail.

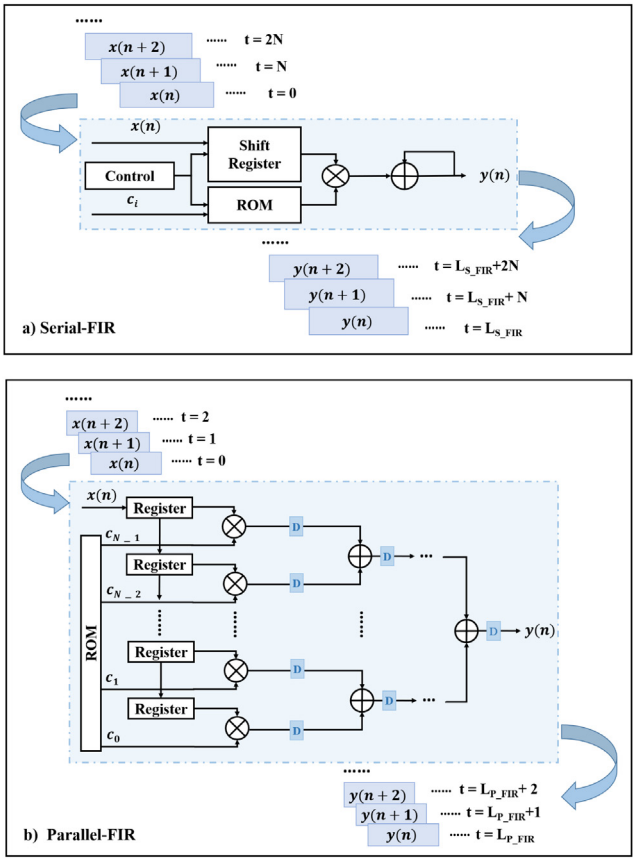

Fig. 3. FIR Architecture for hardware implementation: (a) Serial-FIR and (b) Parallel-FIR.

3.1. Latency-costly parallel architecture/classical straightforward parallel architecture

A latency-costly parallel architecture is based on serial-FIR (S-FIR), which is a straightforward and hardware-efficient method for bottom-level hardware implementation. In this architecture, parallelism is only addressed on the top level.

3.1.1. Serial FIR (S-FIR)

A fundamental S-FIR architecture is shown in Fig. 3. It includes a shift register for caching N input signals, a ROM for storing all filter coefficients, a control logic block for synchronizing the shift register and ROM, a multiplier, and an adder for arithmetic calculation. The input signals are flowed into the S-FIR sequentially at a speed of one sample per N clock cycles, since N multiplication and addition (M&A) operations are required for each convolution. With controller, the data stream flows and caches in the shift register, and are calculated together with the corresponding filter coefficient simultaneously. The output signals are generated sequentially cycling one time at every N clocks.

3.1.2. Classical serial DD-LMS-DAE (S-DD-LMS-DAE)

In terms of the FIR-based DD-LMS-DAE, the coefficients of FIR filter are not learned in advanced and updated adaptively. A feedback loop is employed to calculate the inverse response of channel step by step. As shown in Fig. 4, a DD-LMS-DAE includes a FIR filter for equalization, an adder (subtraction) for calculating distance vector between the equalized output signal and the reference signal, a symbol recovery block for direct decision, a ROM for storing the training sequence, a control logic block for selecting the reference signal and synchronizing the calculation, a shift register for caching the input signals, a delay

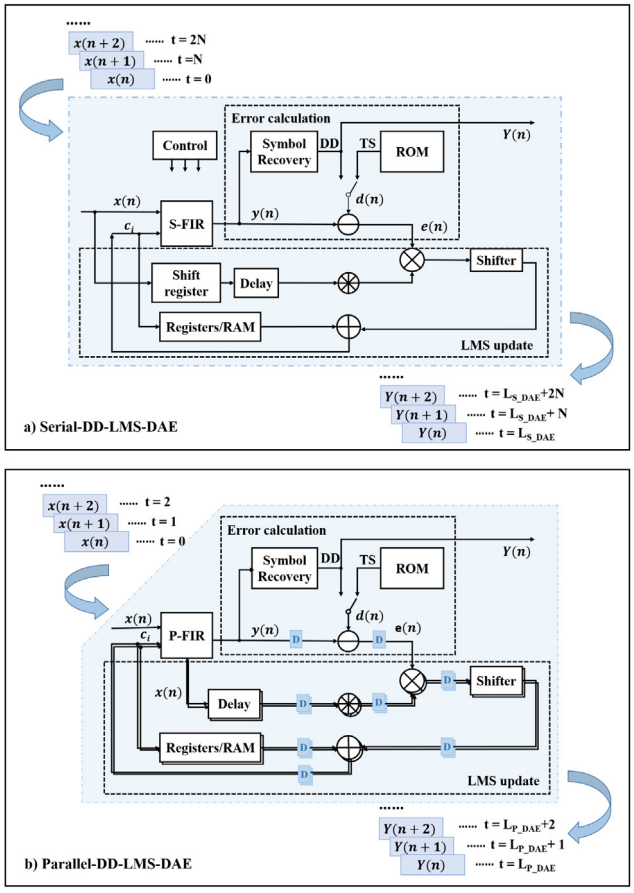

Fig. 4. DD-LMS-DAE Architecture for hardware implementation: (a) Serial-DD-LMS-DAE and (b) Parallel-DD-LMS-DAE.

block for balancing the path latency, registers for saving the old filter coefficients, a conjunction block, a multiplier, a shifter, and an adder for LMS updating. The input signals are fed into the DAE at the speed of one sample per N clock cycles. The output signals of DAE are generated at one sample per N clock cycles.

3.1.3. Top-level parallel of algorithms

As mentioned in Section 1, in order to balance the transmission data rate and FPGA-based signal processing rate, the algorithms should be implemented in parallel. However, algorithms like DD-LMS-DAE require consecutive calculation due to the output dependency caused by the successive input samples. To maintain the calculation continuity, the incoming high-speed data stream should be written into a large memory for caching and then read out for processing frame by frame. Fig. 5 shows the top-level parallel architecture and the data flow of parallelism with parallel S-DD-LMS-DAEs. In this architecture, memory is an important element to transform the parallel data stream into a serial data stream. Moreover, to cope with the discontinuities at the start of each individual S-DD-LMS-DAE, a data overlap between adjacent processing modules and a training sequence at the beginning of each frame is required.

At the receiver, symbol synchronization based on auto-correlation also needs sequential calculation implemented with FIR structure. Hence, the throughput of symbol synchronization is one sample every N clock cycles as well. Finally, the receiver can process data at a throughput of P samples every N clock cycles.

3.1.4. Latency analysis

Normally, a complex multiplication introduces a latency of 3 clock cycles while a real multiplication has a latency of 1 clock cycle on FPGA when the pipeline registers are irrespective. Other arithmetic operations like complex addition/real addition result in a latency of 1 clock cycle, while logic operation introduces zero latency. Each data registration occupies a latency of 1 clock cycle. For example, in DAE, the input signals flowing into a serial FIR are supposed to pass through a shift register (1 clock cycle), a multiplier (1 clock cycle), and an adder (1 clock cycle). N arithmetic calculations are required to generate one output signal. Thus, the latency of a real-valued S-FIR filter is at least $3N$ clock cycles. Similarly, a real-valued S-DD-LMS-DAE results in a latency of $3N + 2$ clock cycles conservatively. The latency caused by the receiver DSP algorithms are analyzed and listed in Table 1.

The latency of the whole digital receiver is accumulated by all the algorithms and memories, and is expressed as

$$L_{Total} = L_{\mathrm{Pr}\,oces} + L_{Cach} \tag{4}$$

where $L_{\mathrm{Pr}\,oces}$ is the latency of all the algorithms for signal processing. L_{Cach} represents the latency caused by caching. The significant latency contribution of this architecture comes from the caching for parallel-to-serial transformation. The incoming parallel data stream is stored in a memory and read out in a serial way. A latency of $\lceil (P-1) \times F/P \rceil$ is introduced for caching, where P is the number of parallel lanes and F is the frame length. Generally, the value of F is usually larger than 10 thousand of symbols, leading to a large caching latency in comparison to the processing latency. Regardless of pipeline registering, the total DSP-introduced latency in this case is $(9 + 3S + 3N + \lceil \log_2(P) \rceil + 3 \times \lceil (P-1) \times F/P \rceil)$ clock cycles, where S is the length of synchronization header. For example, F is 10000 symbols, S is 128 symbols, N is 4 taps and P is 8 lanes. The generated latency is 26658 clock cycles, which approximately equals to 133.29 μs when FPGA operates at 200 MHz. It is composed of signal processing latency (2.04 μs) and caching latency (131.25 μs). So, the caching latency is the main contributor of the total latency.

3.2. Latency-efficient parallel architecture/proposed deep-parallel architecture

Different from the latency-costly parallel architecture as discussed in Section 3.1, all the algorithms are implemented in parallel from top to bottom level in the latency-efficient parallel architecture. It is based on the parallel FIR (P-FIR) structure, which unfolds the convolution operation and calculates the M&As in parallel within one clock cycle using dedicated hardware rather than shared hardware. It reduces the latency and improves the throughput at the expense of more hardware resource. In this part, a deep-parallel (parallelism from top to down) receiver architecture based on a thorough P-FIR is discussed.

3.2.1. Parallel FIR (P-FIR)

In contrast to a S-FIR architecture, a thorough P-FIR architecture uses N multipliers and a $log_2(N)$ depth adder tree to conduct the N arithmetic calculations simultaneously, as shown in Fig. 3. In addition, pipeline registers are inserted to cache the intermediate results of each step for throughput optimization. Consequently, it can receive, process and output the data stream at a throughput of one sample per clock cycle.

3.2.2. Enhanced parallel DD-LMS-DAE (P-DD-LMS-DAE)

The P-DD-LMS-DAE is developed based on the S-DD-LMS-DAE. As displayed in Fig. 4, P-FIR is used instead of S-FIR, and N times of hardware is employed for LMS updating. It is worth noting that the delay compensation should be re-designed to adjust the timing of feedback loop in order to guarantee the calculation synchronization. Moreover, fully-pipeline technique is employed, enabling the DAE to achieve the maximum throughput at one sample per clock cycle.

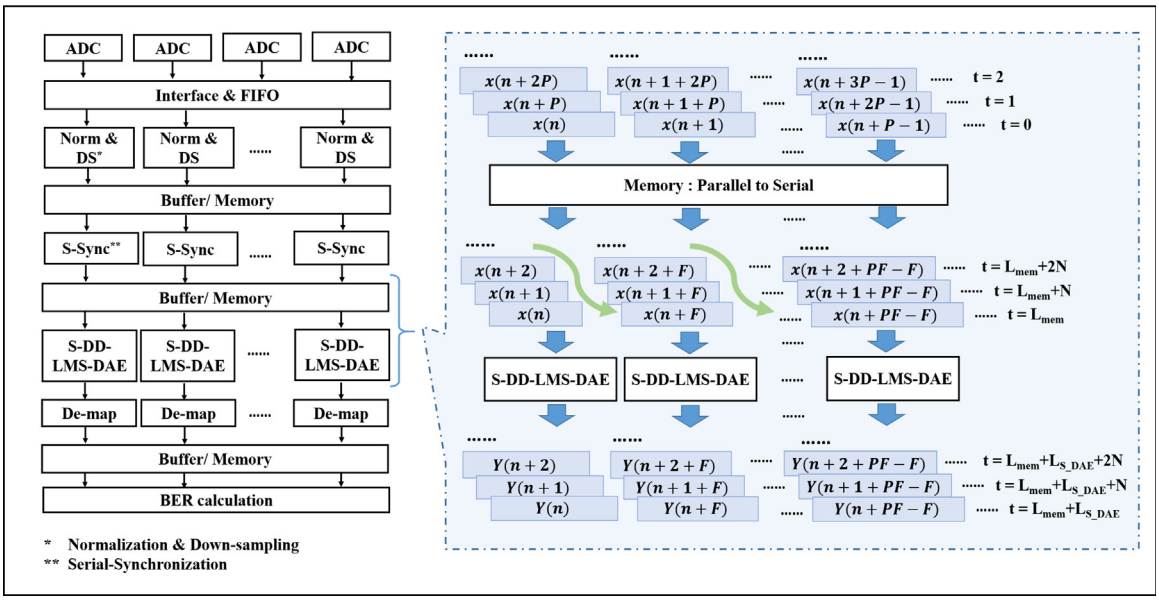

Fig. 5. Parallel of algorithms only on top level (Latency-costly parallel architecture).

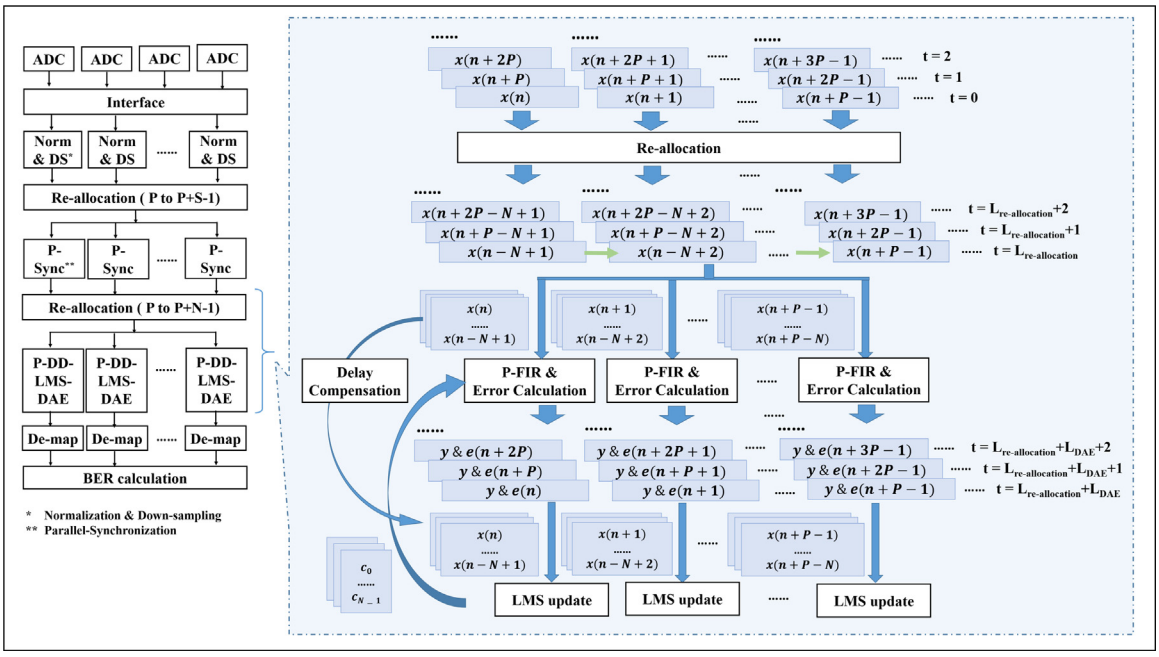

Fig. 6. Deep-parallel of algorithms (Latency-efficient parallel architecture).

3.2.3. *Deep-parallel of algorithms*

For a high-speed data stream, large memory is required to process the stream in the latency-costly parallel architecture, since P samples is possible to arrive at the digital receiver simultaneously. However, going back and forth to the memory is expensive for FPGA design. To avoid visiting dedicated large memory, a deep-parallel architecture is developed based on the enhanced P-DD-LMS-DAE.

As described in Section 2, there are two challenges existing in the DD-LMS-DAE parallel implementation. In order to deal with the challenge of output dependency resulted from the successive input, a data re-allocation scheme is proposed. As shown in Fig. 6, P parallel input signals are re-allocated as $P+N-1$ parallel input signals, and then broadcasted to P parallel P-DD-LMS-DAE modules at the same clock cycle. It is worth noting that the re-allocation implementation only needs several registers to eliminate the penalties from large memory access directly in the latency-costly architecture. In this way, all the parallel P-DD-LMS-DAE modules could get the required successive input signals for calculation at one clock cycle. Therefore, the dependency of parallel P-DD-LMS-DAE modules is de-correlated and the operations are conducted independently. The spatial continuity of the input signals is still preserved. In other words, the sequential calculation is flattened to all the parallel processing modules. Benefiting from the re-allocation scheme, there is no discontinuities between adjacent processing modules in this architecture since all the input signals are ultimately processed by all processing modules. Therefore, only a short zero-padding and one training sequence at the beginning is required for the data transmission.

To cope with the iterative adaptive update challenge, a look-ahead computation technique is introduced. Intrinsically, the filter coefficients are updated at intervals since the P parallel error signals $e_k(n)$ are

Table 1
Latency of algorithms in receiver.

$L_{Processing}$[a]							$L_{Caching}$
Cases		Algorithms					
		Norm +DS	Synchronization	DD-LMS-DAE	De-map	BER calculation	Memories/Re-allocation
A	*Real*[b]	3	$3\times S$	$3\times N+2$	1	$3+\lceil\log_2 P\rceil$	$3\times\lceil(P-1)\times F/P\rceil$
	Complex[c]	5	$5\times S$	$5\times N+2$	1	$3+\lceil\log_2 P\rceil$	
B	*Real*[b]	3	$1+\lceil\log_2 S\rceil$	$4+\lceil\log_2 N\rceil$	1	$3+\lceil\log_2 P\rceil$	$\left\lceil\frac{P+S+2\times N-3}{P}\right\rceil+1$
	Complex[c]	5	$3+\lceil\log_2 S\rceil$	$6+\lceil\log_2 N\rceil$	1	$3+\lceil\log_2 P\rceil$	

S: length of synchronization header; N: number of filter coefficients; P: number of parallel lanes; F: frame length.
[a]Without pipeline registers.
[b]Data for processing are real value.
[c]Data for processing are complex value.

available at the same time. But all the P error signals $e_k(n)$ contain valid information for updating the filter coefficients. Each estimated error signal refers to a distinct distance vector towards the optimal point. Applying the look-ahead computation technique, all the information of P distance vectors is combined to generate the new set of filter coefficients. Notably, the update process maintained spatial iterative, which means the sequential iteration is folded horizontally. Assuming P is 3 lanes, the LMS updating procedure with look-ahead computation technique is expressed as

$$\begin{aligned} c_{k=1}(n) &= c_{k=0}(n)+2\cdot\mu\cdot e_{k=0}(n)\cdot {x_{k=0}}^*(n)\\ c_{k=2}(n) &= c_{k=1}(n)+2\cdot\mu\cdot e_{k=1}(n)\cdot {x_{k=1}}^*(n)\\ c_{k=3}(n) &= c_{k=2}(n)+2\cdot\mu\cdot e_{k=2}(n)\cdot {x_{k=2}}^*(n)\\ c(n+1) &= c_{k=3}(n), c(n)=c_{k=0}(n) \end{aligned} \tag{5}$$

Thanks to the look-ahead computation technique, the adaptive updating efficiency is enhanced, and the length of training sequence is shortened by P times. Combining the look-ahead computation technique and parallel technique, Eqs. (1)–(3) are turned into

$$y_k(n)=\sum_{i=0}^{N-1} c(n-D)_i\cdot x_k(n-i), k=0,1,\ldots,P-1 \tag{6}$$

$$c(n+1)_i = c(n)_i+2\cdot\mu\cdot\sum_{k=0}^{P-1} e_k(n-D)\cdot {x_k}^*(n-D-i) \tag{7}$$

$$e_k(n)=d_k(n)-y_k(n) \tag{8}$$

where D is the feedback loop delay. According to Eqs. (6)–(8), a deep-parallel fully-pipeline DD-LMS-DAE is built, in which the successive input samples and iterative updating scheme are both flatten to spatial domain. The data flow has been demonstrated in Fig. 6 in detail. In addition, a deep-parallel fully-pipeline symbol synchronization is also realized with the help of the data re-allocation scheme. All the algorithms at the receiver are implemented in parallel from top level to bottom level, enabling the data stream to flow smoothly through the processing modules without caching in large memories. Therefore, the receiver can receive, process, and output data at a throughput of P samples per clock cycle.

3.2.4. Latency analysis

Generally, an adder tree with N input data has a latency of $\lceil\log_2(N)\rceil$ clock cycles. So the latency of a real-valued P-FIR is $1+\lceil\log_2(N)\rceil$ clock cycles and a real-valued P-DD-LMS-DAE is $4+\lceil\log_2(N)\rceil$ clock cycles. As for the latency of other algorithms, the details are listed in Table 1. Furthermore, the latency of fundamental calculating elements is doubled for fully-pipeline since every intermediate result is registered. Hence, a fully-pipeline real-valued P-DD-LMS-DAE occupies a latency of $8+2\times\lceil\log_2(N)\rceil$ clock cycles.

Compared with the latency-costly parallel architecture, the main latency caused by caching in large memories is removed. In this architecture, only several registers are used to re-allocate P parallel input signals to $P+N-1$ parallel input signals for deeply parallel processing in the DD-LMS-DAE. Similarly, P parallel input signals are also re-allocated to $P+S-1$ parallel input signals for symbol synchronization. To optimize the hardware utilization efficiency, only one data re-allocation block is employed for the whole digital receiver. $P+N+S-2$ parallel input signals are simultaneously generated and shared by both the symbol synchronization and DD-LMS-DAE. Consequently, the caching latency of this architecture is $\lceil(P+S+2\times N-3)/P\rceil+1$ clock cycles in total. The total latency of the real-valued digital receiver is summation of the re-allocation block and algorithms except for symbol synchronization, $12+\lceil\log 2(N)\rceil+\lceil\log 2(P)\rceil+\lceil(P+S+2\times N-3)/P\rceil$ clock cycles. Here, the latency caused by the pipeline registers is ignored.

A comparison of the calculated latency between the latency-costly architecture (case A) and the latency-efficient architecture (case B) is shown in Fig. 7. The charts are made based on the assumption of zero pipeline registers, 10000 symbols frame length (F), 128 symbols synchronization header length (S), 8 parallel lanes (P) when N grows, and 4 taps (N) when P increases. In case A, caching latency is the dominant proportion and grows obviously with P increment, while processing latency rises slightly with the increment of both N and P. In case B, caching latency grows with the increment of N, but decreases as P increases. As for the processing latency, it has a tiny increment with both N and P as well. As we can see in Fig. 7, the total latency of case B is much smaller than case A. Under the condition that P is 8 lanes and N is 4 taps, the total latency of case B is 35 clock cycles, which is just 0.13% of 26658 clock cycles for case A. Considering a FPGA operation frequency of 200 MHz, the total DSP-introduced latency of case A exceeds 75 μs (15000 clock cycles). Whereas, the latency of case B is only around 0.2 μs (40 clock cycles) which gives more latency headroom for other parts in latency-critical applications.

3.2.5. Cost analysis

System cost is another consideration factor when designing a real-time system. We proposed to use deep-parallel scheme in our design to reduce the system latency. Parallel implementation is necessary when the required transmission data rate is much higher than the operating frequency of FPGA. Straightforward parallel is latency-costly due to the data caching (as shown in case A of Fig. 7), while much smaller caching latency is required by deeply parallel (as shown in case B of Fig. 7). The main difference between the straightforward parallel implementation and the deeply parallel implementation is the bottom-level realization. Serial-FIR is used for straightforward parallel, while parallel-FIR is used for deeply parallel. Parallel-FIR needs more hardware resources than serial-FIR. For example, N times of multipliers and $log_2(N)$ times of adders are needed for parallel-FIR. However, deeply parallel architecture does not need large memories for data caching. Memory is expensive hardware resource on FPGA. Thus, the total system cost is related to the cost trade-off between the multiplier and the memory.

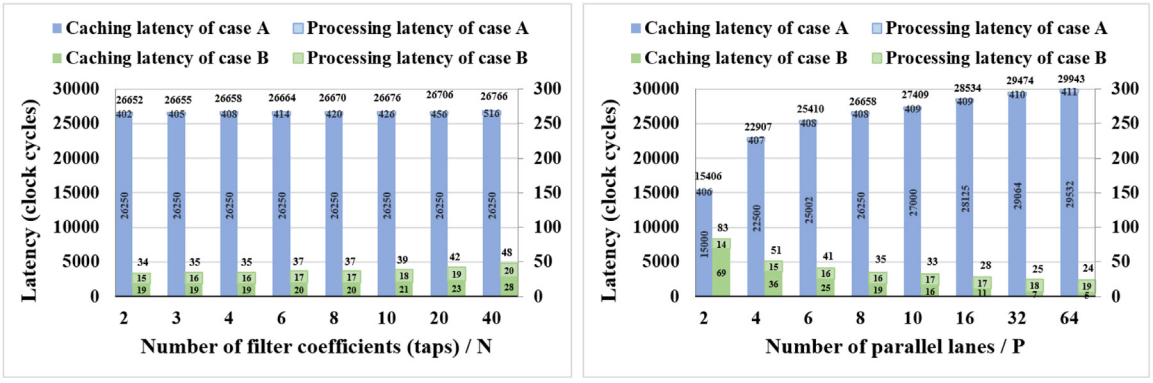

Fig. 7. Calculated latency comparison between latency-costly parallel architecture (case A) and latency-efficient parallel architecture (case B).

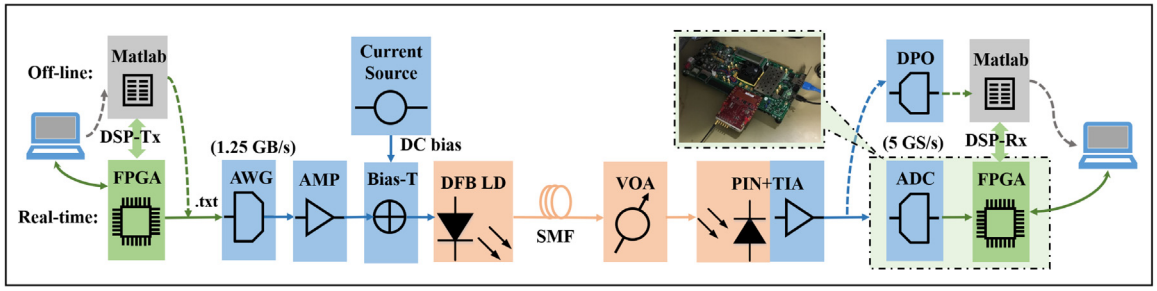

Fig. 8. Experimental setup of the optical transmission system. FPGA: field programmable gate array; AWG: arbitrary waveform generator; AMP: amplifier; DFB LD: distributed feedback laser diode; DC: direct current; SMF: single mode fiber; VOA: variable optical attenuator; TIA: trans-impedance amplifier; ADC: analog-to-digital converter; DPO: digital phosphor oscilloscope.

4. Experiment and results

The developed deep-parallel fully-pipeline FPGA-based PAM-4 receiver is investigated in an optical fiber link. The experimental setup is shown in Fig. 8. At the transmitter side, a digital baseband PAM-4 signal with synchronization header and training sequence is generated in the FPGA and stored as a text file. The length of the repeating data stream is 80000 symbols, including 128 symbols for symbol synchronization and 8000 symbols for start-up training. Then, the digital baseband signal is sent to an arbitrary waveform generator (AWG) to generate a 1.25 GBaud/s analog PAM-4 signal, yielding a bit rate of 2.5 Gbit/s. After that, the output signal from AWG is amplified by an electrical amplifier and combined with the bias current (DC) via a bias-T, which is used to drive an off-the-shelf C-band distributed feedback laser diode (DFB-LD). The optimal values of DC and input voltage are 50 mA and 3.5 V, respectively. After the DFB-LD, the modulated optical signal is transmitted through a span of single mode fiber (SMF). At the receiver side, an off-the-shelf optical-to-electrical convertor, including a pin-photodiode (PIN-PD) and a trans-impedance amplifier (TIA), is used for signal detection. And a variable optical attenuator (VOA) is placed at the front of the PD to adjust the received optical power. Then, the detected electrical analog signal arrived at our developed FPGA-based real-time receiver for signal processing. On the FPGA evaluation board, a commercial FPGA Mezzanine Card (FMC) carrying a 10-bit quad ADC (E2V-EV10AQ190) is used for analog to digital signal conversion. The high-speed data stream is sampled by four interleaved ADC cores running at the maximum sampling rate of 5 GSample/s. These four interleaved digital signals are transformed to 32 parallel signals with the help of high-speed serializer/deserializer (SerDes). After down-sampling by 4 times, the digital signals are parallel processed with 8 lanes at a clock frequency of 156.25 MHz on FPGA. The DSP-introduced latency and BER performances are measured with ChipScope from Xilinx, which is the virtual probe for internal signals debugging on FPGA.

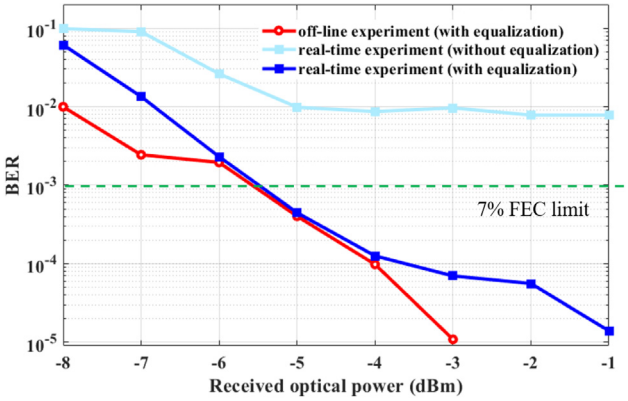

Fig. 9. BER performance as a function of received optical power for 2.5 Gbit/s PAM-4.

Before the real-time BER measurement, off-line experiments are carried out for performance estimation and comparison. As displayed in Fig. 8, the experimental setup is a little different from the real-time version: at the transmitter, the digital PAM-4 signal is generated by Matlab and then sent to the AWG for analog signal generation; at the receiver, the detected analog signal is captured by a digital processing oscilloscope (DPO) and processed with Matlab. According to the off-line processing, a 4-tap symbol-space (T-space) DAE with the main cursor positioned at 1 is adequate to compensate the distortion in the transmission link, as shown in Fig. 9 of red curve. The calculated BER values across the received optical power are ranged from −8 dBm to −3 dBm by 1 dBm step.

Then the real-time experiment is demonstrated based on the DAE parameters specified in the off-line experiment. The real-time BER performance as a function of received optical power is presented in Fig. 9. Considering the 7% FEC limit of 1×10^{-3}, the receiver sensitivity

Table 2
Resource utilization of the implemented PAM4 receiver.

Resource	Slice LUTs (303600)	Slice registers (607200)	Slice (75900)	LUT as logic (303600)	LUT flip flop pairs (303600)	Block RAM tile (1030)	DSPs (2800)
ADC interface	2.62%	1.47%	4.26%	2.59%	1.24%	3.93%	0%
DSP-RX	8.8%	15.94%	26.88%	8.47%	7.96%	17.52%	38.86%
DAE	0.72%	1.72%	2.84%	0.72%	0.63%	0.05%	2.28%

of the real-time receiver with equalization (blue curve in Fig. 9) is about −5.5 dBm, which is consistent with the value of the offline experiment. The case of real-time receiver without equalization is also shown in Fig. 9, where error-floor occurs. Therefore, the feasibility of the FPGA-based PAM-4 receiver with equalization is verified.

In the real-time experiment, the DSP-introduced latency is 0.3904 μs (61 clock cycles) in total, in which 0.3072 μs (35 clock cycles) is consistent with the former calculation in Section 3. The additional 0.1088 μs (17 clock cycles) is caused by the pipeline registering and 0.0576 μs (9 clock cycles) is introduced by signal reset and control. The training latency is not included in the total DSP-introduced latency above. For both off-line and real-time experiment, a training sequence with 8000 symbols and an LMS updating step of 0.004 are applied to help the DAE tap weights converge to the targeted point. Benefiting from parallel implementation, the latency for training is shorten by the number of parallel lanes, leading to a latency of 6.4 μs (1000 clock cycles).

In this work, the used FPGA is the Xilinx VC707 evaluation board featured with a Virtex-7 XC7VX485T-2FFG1761 device, and the ADC we used is FMC126 with E2V-EV10AQ190 chip. The ADC sampling resolution is 10-bit and the FPGA internal signal processing precision is 16-bit. The resource utilization of the implemented PAM-4 receiver is listed in Table 2. It is worth mentioning that the proposed deep-parallel method is potentially effective for 100 Gbit/s or higher transmission. In our demonstration, the data rate is limited by the employed hardware device (mainly ADC), not the proposed method. When the transmission data rate increases, the number of parallel lanes and hardware utilization increase. For 100 Gbit/s or higher data rates, ASIC with abundant hardware resource can be used for our proposed deep-parallel method to reduce the latency. For ASIC implementation, the number of parallel lanes can be reduced at least by half since the operating frequency of ASIC is much higher than FPGA. The typical operating frequency of FPGA is 200 MHz, while >500 MHz is attainable with ASIC. Moreover, for FPGA implementation, structured hardware resources are utilized, leading to the waste of some unnecessary area in layout design. ASIC implementation is more flexible since the implementation can be mapped to logic gates. In this way, the layout design of ASIC can be optimized. This leads to reduced cost due to the reduced chip area required. Thus, the proposed method is effective to be extended to ASIC implementation for >100 Gbit/s direct detection or coherent detection systems with reduced latency and cost.

5. Conclusion

We have designed and implemented a low-latency real-time PAM-4 receiver, enabling by a latency-efficient parallel architecture. The DAE techniques are applied to improve channel performance. A novel re-allocation scheme is proposed to cope with the issue of the output dependency resulting from the successive input samples, and release the utilization of large memories. A look-ahead computation technique is also introduced to improve the adaptive updating efficiency. Compared with the classical straightforward parallel method, the achieved latency is dramatically reduced by using the latency-efficient parallel architecture, resulting in the DSP-introduced latency of average 0.4 μs, which meets the end-to-end latency requirement of 5G networks for latency-sensitive scenarios. The feasibility of the developed low-latency PAM-4 receiver has been verified in an optical fiber transmission link with 2.5 Gbit/s data rate. Moreover, the low-latency real-time PAM-4 receiver with more than 100 Gbit/s data rate is feasible to be implemented by using our proposed deep-parallel technique and high-performance hardware platform.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.optcom.2021.127836.

References

[1] Xuewu Xu, Yuechao Pan, Phyu Phyu Mar Yi Lwin, Xinan Liang, 3D holographic display and its data transmission requirement, in: 2011 International Conference on Information Photonics and Optical Communications, IEEE, 2011, pp. 1–4.
[2] Erik Agrell, Magnus Karlsson, AR Chraplyvy, David J Richardson, Peter M Krummrich, Peter Winzer, Kim Roberts, Johannes Karl Fischer, Seb J Savory, Benjamin J Eggleton, et al., Roadmap of optical communications, J. Opt. 18 (6) (2016) 063002.
[3] Honghang Zhou, Yan Li, Yuyang Liu, Lei Yue, Chao Gao, Wei Li, Jifang Qiu, Hongxiang Guo, Xiaobin Hong, Yong Zuo, et al., Recent advances in equalization technologies for short-reach optical links based on PAM4 modulation: A review, Appl. Sci. 9 (11) (2019) 2342.
[4] Ethernet Task Force, IEEE Standard P802. 3bs 200 Gb/s and 400 Gb/s, 2017.
[5] Xizi Tang, Shuangyue Liu, Xuekai Xu, Jia Qi, Mengqi Guo, Ji Zhou, Yaojun Qiao, 50-GB/s PAM4 over 50-km single mode fiber transmission using efficient equalization technique, in: Optical Fiber Communication Conference, Optical Society of America, 2019, pp. W2A–45.
[6] Imtiaz Parvez, Ali Rahmati, Ismail Guvenc, Arif I Sarwat, Huaiyu Dai, A survey on low latency towards 5G: RAN, core network and caching solutions, IEEE Commun. Surv. Tutor. 20 (4) (2018) 3098–3130.
[7] NGMN Alliance, NGMN 5G white paper, in: Next Generation Mobile Networks, White Paper, 2015, pp. 1–125.
[8] Chen Chen, Xuefeng Tang, Zhuhong Zhang, Transmission of 56-Gb/s PAM-4 over 26-km single mode fiber using maximum likelihood sequence estimation, in: Optical Fiber Communication Conference, Optical Society of America, 2015, pp. Th4A–5.
[9] Kangping Zhong, Xian Zhou, Yuliang Gao, Wei Chen, Jiangwei Man, Li Zeng, Alan Pak Tao Lau, Chao Lu, 140-GB/s 20-km transmission of PAM-4 signal at 1.3 μm for short reach communications, IEEE Photonics Technol. Lett. 27 (16) (2015) 1757–1760.
[10] Xiaodan Pang, Oskars Ozolins, Simone Gaiarin, Aditya Kakkar, Jaime Rodrigo Navarro, Miguel Iglesias Olmedo, Richard Schatz, Aleksejs Udalcovs, Urban Westergren, Darko Zibar, et al., Experimental study of 1.55-μm EML-based optical IM/DD PAM-4/8 short reach systems, IEEE Photonics Technol. Lett. 29 (6) (2017) 523–526.
[11] Qian Hu, Karsten Schuh, Mathieu Chagnon, Fred Buchali, Henning Bülow, 84 GBD faster-than-Nyquist PAM-4 transmission using only linear equalizer at receiver, in: Optical Fiber Communication Conference, Optical Society of America, 2019, pp. W4I–2.
[12] Jinlong Wei, Nicklas Eiselt, Helmut Griesser, Klaus Grobe, Michael H Eiselt, Juan Jose Vegas Olmos, Idelfonso Tafur Monroy, Joerg-Peter Elbers, Demonstration of the first real-time end-to-end 40-Gb/s PAM-4 for next-generation access applications using 10-Gb/s transmitter, J. Lightwave Technol. 34 (7) (2016) 1628–1635.
[13] Nicklas Eiselt, Jinlong Wei, Helmut Griesser, Annika Dochhan, Michael H Eiselt, Jörg-Peter Elbers, Juan José Vegas Olmos, Idelfonso Tafur Monroy, Evaluation of real-time 8× 56.25 Gb/s (400G) PAM-4 for inter-data center application over 80 km of SSMF at 1550 nm, J. Lightwave Technol. 35 (4) (2016) 955–962.
[14] Huaiyu Zeng, Xiang Liu, Sharief Megeed, Naresh Chand, Frank Effenberger, Real-time demonstration of CPRI-compatible efficient mobile fronthaul using FPGA, J. Lightwave Technol. 35 (6) (2017) 1241–1247.
[15] Junwen Zhang, Xin Xiao, Jianjun Yu, Jun Shan Wey, Xingang Huang, Zhuang Ma, Real-time FPGA demonstration of PAM-4 burst-mode all-digital clock and data recovery for single wavelength 50G PON application, in: Optical Fiber Communication Conference, Optical Society of America, 2018, pp. M1B–7.
[16] Benedikt Baeuerle, Arne Josten, Marco Eppenberger, David Hillerkuss, Juerg Leuthold, Low-complexity real-time receiver for coherent Nyquist-FDM signals, J. Lightwave Technol. 36 (24) (2018) 5728–5737.

[17] Peilin Wang, Chao Li, Zhengyuan Xu, A cost-efficient real-time 25 Mb/s system for LED-UOWC: design, channel coding, FPGA implementation, and characterization, J. Lightwave Technol. 36 (13) (2018) 2627–2637.
[18] Ehab Al-Rawachy, Roger Philip Giddings, Jianming Tang, Experimental demonstration of a real-time digital filter multiple access PON with low complexity DSP-based interference cancellation, J. Lightwave Technol. 37 (17) (2019) 4315–4329.
[19] Carlo Safarian, Tokunbo Ogunfunmi, Walter J Kozacky, Basant K Mohanty, FPGA implementation of LMS-based FIR adaptive filter for real time digital signal processing applications, in: 2015 IEEE International Conference on Digital Signal Processing, DSP, IEEE, 2015, pp. 1251–1255.
[20] Mohammad Sadegh Alizadeh, Javad Bagherzadeh, Mohammad Sharifkhani, A low-latency QRD-RLS architecture for high-throughput adaptive applications, IEEE Trans. Circuits Syst. II Express Briefs 63 (7) (2016) 708–712.
[21] Nongmaithem Lalleima, Bidyalaxmi Devi, Vimal Kant Pandey, Comparative analysis of design methodologies for parallel FIR filter, Int. J. Comput. Appl. 975 (2014) 8887.
[22] Konstantinos Maragos, Christos Spatharakis, George Lentaris, Panagiotis Kontzilas, Stefanos Dris, Paraskevas Bakopoulos, Hercules Avramopoulos, Dimitrios Soudris, A flexible, high-performance FPGA implementation of a feed-forward equalizer for optical interconnects up to 112 Gb/s, J. Signal Process. Syst. 88 (2) (2017) 107–125.
[23] Janusz A. Starzyk, Mohammad Eshghi, Highly parallel adaptive filter, in: IEEE International Symposium on Circuits and Systems, IEEE, 1989, pp. 2116–2119.
[24] Katsushige Matsubara, Kiyoshi Nishikawa, Hitoshi Kiya, Pipelined LMS adaptive filter using a new look-ahead transformation, IEEE Trans. Circuits Syst. II Analog Digital Signal Process. 46 (1) (1999) 51–55.
[25] Dimitris G. Manolakis, Vinay K. Ingle, Stephen M. Kogon, Statistical and adaptive signal processing: spectral estimation, signal modeling, adaptive filtering, and array processing, 2000.