Carrier Phase Recovery of 64 GBd Optical 16-QAM Using Extensive Parallelization on an FPGA

Vatistas Kostalampros, Konstantinos Maragos
George Lentaris, Dimitrios Soudris
Microprocessors and Digital Systems Lab.
National Technical University of Athens
{vkostalampros, komaragos}@microlab.ntua.gr

Christos Spatharakis, Nikos Argyris
Hercules Avramopoulos
Photonics Communications Research Lab.
National Technical University of Athens
{cspatha, nargyris}@mail.ntua.gr

Stefanos Dris
André Richter
VPIphotonics GmbH
stefanos.dris@vpiphotonics.com

***Abstract*—Carrier phase recovery (CPR) for phase noise mitigation is a vital part of modern coherent optical systems. As higher order M-QAM modulation formats are being employed to increase network capacity, CPR complexity grows as well, and more powerful chips are needed to cope with the advanced DSP. In this paper we present an FPGA-based flexible architecture of the Nonlinear Least Squares (NLS) CPR algorithm, targeting present-day and future generation systems. We describe optimization approaches at the algorithmic and HW levels to facilitate HW efficiency, while we combine multiple parallelization techniques to achieve high-throughput processing. Considering various FPGA devices, we perform fine-grain exploration with respect to cost, throughput and accuracy to support up to 64 GBd 16-QAM links with less than 1 dB SNR penalty.**

I. Introduction

New deployments of 100G channels are replacing legacy 10G/40G direct detection systems in core optical networks. Such high capacity systems rely on Dual-Polarization (DP) QPSK coherent transceivers (TRxs), equipped with high performance electronics for the implementation of modulation/demodulation algorithms at symbol rates of 28/32 GBd. Demand for increased network capacity is now fueling the development of systems relying on higher order modulation formats, and higher line-rates: DP-16-QAM operating at 32 and 64 GBd is expected to feature prominently in next-generation 400G and Terabit channel deployments.

Such advanced architectures benefit from the integration of powerful DSP chips, for real-time mitigation of impairments originating from fiber transmission, as well as from the transceiver components themselves [1]. An example of the latter is the phase noise induced by the free-running transmitter (Tx) and local oscillator (LO) lasers employed in intradyne coherent systems; a carrier phase recovery (CPR) stage is therefore necessary in the receiver.

CPR implementations for QPSK are often based on the Viterbi-Viterbi 4^{th} Power estimator (VV4E) [2]. Since it performs full modulation stripping, accurate phase tracking can be achieved. However, when migrating to higher M-QAM schemes, the VV4E algorithm cannot be directly applied to wipe the modulation, since not all symbols lie on equidistant phase axes (90°), as is the case with QPSK. Blind Phase Search (BPS) [3] and QPSK-partitioning [4] are alternative approaches employed for CPR of high-order QAM constellations. The former comes at the expense of high computational complexity, while the latter performs poorly with formats higher than 16-QAM. A potentially low-complexity alternative is the VV4E-based NLS scheme. This exploits a symbol magnitude-dependent nonlinear weighting function that minimizes the least squares error of the final phase estimate [5]. It has been shown to outperform BPS in terms of linewidth tolerance [6].

In the direction of employing powerful chips for this type of DSP, besides the traditional ASIC approach followed in commercialization, the community also relies on FPGA platforms (especially for prototyping and R&D). FPGAs provide impressive performance per Watt by facilitating low-level parallel processing of excessive workloads with increased I/O capabilities. Moreover, they enable the reconfiguration of the chip and allow different designs to be loaded offline, or even at run-time. This can be leveraged to accelerate time-to-market and decrease equipment cost, while also allowing future upgrades in response to changing demands. The application of FPGAs for telecom systems has been demonstrated for various DSP building blocks, such as adaptive equalizers reaching up to 56 GBd [7], joint Carrier Frequency and Phase Recovery stages suitable for multi-format operation [8], as well as for real-time MIMO receivers for Spatial Division Multiplexed (SDM) systems [9].

In this work we propose a HW architecture able to support high-throughput CPR with algorithms based on the magnitude-weighted VV4E approach. The resulting circuit provides an indicative throughput/cost for this type of feed-forward CPR, while it can be easily modified at compile-time to adapt to the specifics of each implementation, e.g., to the required weighting function. As a proof-of-concept, we specifically consider the NLS algorithm for 16-QAM signals, targeting symbol rates in use currently (32 GBd), as well as for next generation systems (64 GBd). We begin with a high-level investigation of NLS CPR in MATLAB and optimize it towards efficient HW design. Subsequently, we develop a parametric VHDL model based on various parallelization techniques, such as deep pipelining, systolic arrays, data decomposition with resource reuse (for processing multiple inputs in parallel), and parallelization at arithmetic operation level (for fast formula calculation). Finally, we fine-tune the architectural parameters to exploit the trade-offs and adapt to specific FPGA devices, e.g., Xilinx Virtex-7 VH580T, achieving a processing rate of up to 36 GBd with $\leqslant 1$ dB SNR penalty at BER$= 10^{-3}$ (vs. the theoretical limit), or Virtex UltraScale+ XCVU9P, achieving 64 GBd with $\leqslant 0.9$ dB SNR penalty.

978-1-5386-4881-0/18/$31.00 ©2018 IEEE

II. NLS Algorithm for Carrier Phase Recovery

Let $x[n] = \rho[n]\, e^{j\phi[n]}$ be a received sequence of complex M-QAM symbols corrupted by AWGN and phase noise due to fiber transmission and the combined linewdith of the Tx and LO lasers. As described in [6], a nonlinear transformation is applied on the received digital symbol's magnitude, $\rho[n]$, along with a 4^{th} power operation on its phase, $\phi[n]$, as follows:

$$y[n] = F(\rho[n])\, e^{j4\phi[n]} \tag{1}$$

The transformed complex signal, $y[n]$, is then passed through an FIR filter $w[k]$, with L coefficients. The phase estimate at the $n-\Delta$ received symbol, $\Delta = (L-1)/2$, is then given by:

$$\theta[n-\Delta] = \frac{1}{4}\arg\left\{\sum_{k=0}^{L-1} w[k]\, y[n-k]\right\} \tag{2}$$

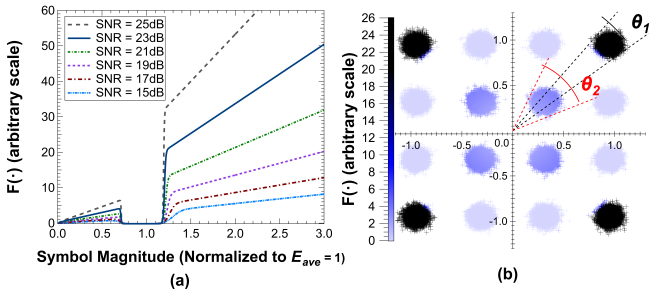

Fig. 1. (a) $F(\cdot)$ as a function of received symbol magnitude (16-QAM). (b) Relative weighting applied by $F(\cdot)$ to a 16-QAM signal (SNR= 23 dB).

$F(\cdot)$ in Eq. 1 is a weighting function dependent on the modulation format. It determines the contribution of each received noisy symbol to the final phase estimate, depending on its magnitude. Fig. 1(a) shows the plots of $F(\cdot)$ as a function of the received symbol magnitude for a 16-QAM constellation and several SNRs. It is observed that the phases of symbols belonging to the middle ring of the constellation do not contribute at all to the final estimate; these symbols have phases that do not lie on equidistant 90° axes and as a result, a 4^{th} power operation will not remove the modulation (a necessary condition for accurate phase tracking). In contrast, symbols belonging to the innermost and the outermost rings are assigned weights toward the final estimate, as each of these rings forms a QPSK subset. A 4^{th} power operation on these symbols will completely wipe the modulation and allow for precise phase estimation.

Another important aspect of the non-linear function is illustrated in Fig. 1(b). Highest amplitude symbols (belonging to the outer QPSK subset) contribute more to the final phase estimate. The phase error due to AWGN is lower for higher-amplitude symbols ($\theta_1 < \theta_2$ in Fig. 1(b)), thus by applying more weighting to these points, the performance of the estimator is improved.

A. Calculation of the optimal non-linearity, $F(\cdot)$

The function $F(\cdot)$ that minimizes the asymptotic variance of the phase estimate is given by [5]:

$$F(\rho[n]) = \frac{g_2(\rho[n])}{g_1(\rho[n]) - g_3(\rho[n])} \tag{3}$$

where:

$$g_i(\rho[n]) = (-1)^{i-1}\frac{8\rho[n]}{M\sigma^2} e^{-(\rho^2[n]/\sigma^2)} \cdot \sum_{\rho_c,\phi_c\in\mathcal{C}} \left[\cos(4(i-1)\phi_c) e^{-(\rho_c^2/\sigma^2)} I_{4(i-1)}\left(\frac{2\rho[n]\rho_c}{\sigma^2}\right)\right] \tag{4}$$

where σ^2 is the AWGN variance, $I_n(z)$ is the n^{th} order modified Bessel function of the first kind, and $\mathcal{C}$ is the set of M constellation points.

B. Initial Evaluation and HW-oriented Optimizations

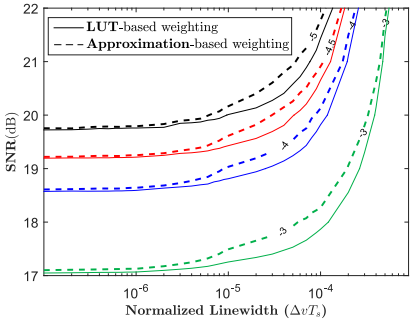

Fig. 2. log(BER) contour plots obtained with the NLS CPR, using the approximation- and LUT-based weighting functions.

The complexity of the nonlinear function, $F(\cdot)$, makes NLS impractical for direct hardware implementation. To tackle this issue, we exploit the weighting curves' partial linearity to perform a 2-stage piecewise polynomial approximation of $F(\cdot)$. We derive a coefficient table, C, which along with the received SNR, facilitates the calculation of Eq. 5 to generate an approximate weight for the input symbol:

$$W(\text{SNR}, \text{Mag}) = (c_1 \cdot \text{SNR} + c_2) \cdot \text{Mag} + (c_3 \cdot \text{SNR} + c_4) \tag{5}$$

where Mag indicates the symbol magnitude. The above formula relies on significantly simpler computations than the initial NLS Eqs. 3 and 4. Moreover, it allows for a hardware realization adapting to the respective SNR. Above all, it significantly decreases the number of stored elements required for the computation: Considering that the $F(\cdot)$ curves can be expressed as a big Look-Up-Table comprised of 301 rows (SNR ranging from 10 to 40.5) and 5000 columns (magnitude ranging from 0 to 5), our derived table consists of only 12 elements (4 coefficients for each of the 3 piecewise linear curves' parts). Translated into bits, this optimization alleviates almost 2 Gb and enables implementation on an FPGA (memory limited to $\approx$ 100 Mb). When tested with MATLAB floating-point simulations, the approximation results in a small accuracy penalty. Fig. 2 compares the BER after phase recovery when employing either the initial NLS curves LUT or the approximation method. The SNR penalty between the 2 implementations is less than 0.3 dB, even for the high phase noise values ($\approx 10^{-4}$), while the lower phase noise values result in an SNR penalty of approximately 0.1 dB.

978-1-5386-4881-0/18/$31.00 ©2018 IEEE

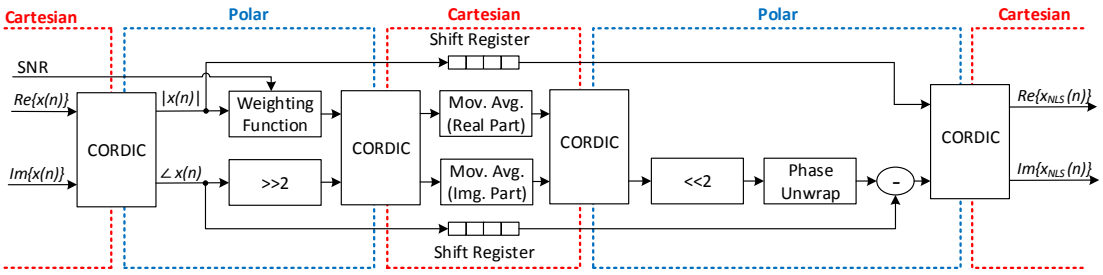

Fig. 3. Block diagram of the NLS hardware architecture.

III. Proposed Hardware Architecture

This section describes the design of the NLS algorithm on an FPGA. Fig. 3 depicts the block-diagram overview of the system architecture. The red- and blue-dashed regions indicate computations performed with Cartesian and Polar coordinates, respectively. The baseline architecture inputs one IQ symbol per cycle, together with the received SNR. In a pipelined fashion, the symbol is forwarded through the remaining stages until its phase has been recovered. First, a CORDIC transforms the symbol into polar coordinates. The magnitude is input to a LUT-based component, which calculates the weighting function of NLS by also taking into account the SNR. The angle is multiplied by 4 (shifted by 2) to strip off the modulation. The weighted and rotated vectors are transformed and filtered via moving average to remove AWGN and effectively track the rapid fluctuations caused by phase noise. The phase of the filtered data is divided by 4 (to cancel out the previous multiplication) and unwrapped before being subtracted from the initial angle (which is delayed via shift registers, along with the magnitude). Finally, the corrected symbol is output in Cartesian coordinates. The following paragraphs provide details for the key components in the proposed architecture and the parallelization techniques devised to increase their throughput.

The coordinate transformations (Cartesian to Polar and vice-versa) are realized with the 4 CORDIC Xilinx IPs shown in Fig. 3. The input/output phase format is set to "Scaled Radians" in order to avoid the costly multiplication with π in the phase unwrap module (Eq. 6). To increase throughput, we instantiate multiple CORDICs per stage, i.e., equal to the parallelization factor P of our architecture. Therefore, we utilize $4 \cdot P$ CORDICs and process P symbols in parallel.

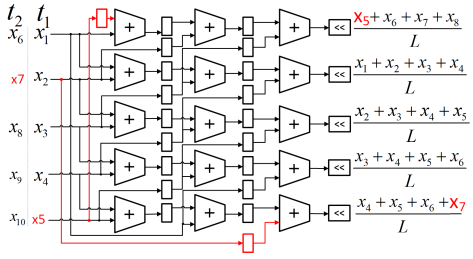

Fig. 4. Moving average filter example (parallelism P=5, filter length L=4).

The moving average function is expressed in the form of $y_n = (x_n + x_{n-1} + \ldots + x_{n-(L-1)})/L$, where x_n denotes the incoming symbol, y_n is the filter's output and L is the length of the filter (equal to an integer power of 2 for simple shift-based division). In serial implementation, the HW circuit utilizes an L-depth FIFO and an accumulator, which adds the new input but subtracts the L-th from the past. To increase throughput, i.e., to process P symbols in parallel, we designed the systolic array shown in Fig. 4. In a parametric fashion, the array utilizes $O(2{\cdot}P{\cdot}L)$ registers to store past information and $P{\cdot}(L-1)$ adders to gradually form the P sums. Fig. 4 presents an example where the parallelism factor is not aligned with the filter length (P=5, L=4). Notice that dedicated registers are automatically placed (red color) to synchronize the symbols (e.g., $x5$, $x7$) that need to be processed by the filters in the upcoming time frames.

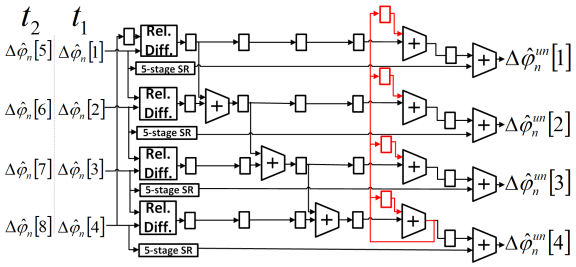

Fig. 5. Phase unwrap architecture example with parallelism order $P = 4$.

The phase unwrap algorithm is formulated as:

$$\Delta\hat{\phi}_n^{un}[n] = \Delta\hat{\phi}_n[n] + \frac{\pi}{2}\left\lfloor 0.5 + \frac{2}{\pi}(\Delta\hat{\phi}_n^{un}[n-1] - \Delta\hat{\phi}_n[n])\right\rfloor \quad (6)$$

where $\Delta\hat{\phi}_n^{un}[n]$ denotes the unwrapped phase estimate of the input wrapped phase $\Delta\hat{\phi}_n[n]$. The recursive nature of this formula imposes a challenge for parallelization, which we tackle by decomposing the input/output data and reusing the sums among the distinct datapaths, as shown in Fig. 5 for $P = 4$. Our parallelization method is based on the recursive substitution of $(\Delta\hat{\phi}_n^{un}[n-1] - \Delta\hat{\phi}_n[n])$ with $(\Delta\hat{\phi}_n[n-1] - \Delta\hat{\phi}_n[n])$ and introduction of a cumulative term $\sum \Delta\phi_n$ to store all the previously processed wrapped estimates. Thus, we form pairs of successive inputs $\Delta\hat{\phi}_n$ to feed the "relative difference" units (Fig. 5), which calculate $\frac{\pi}{2}\lfloor 0.5 + \frac{2}{\pi}(\Delta\hat{\phi}_n[n-1] - \Delta\hat{\phi}_n[n])\rfloor$. Subsequently, in a synchronized cascade, each adder computes the partial $\sum \Delta\phi_n$ for its assigned symbol. Notice that, this partial sum is augmented with the total sum of the previous input P-tuple, which is stored in the last register of the architecture (Fig. 5) and provided via a feedback loop (red color) to all P adders (with overflow control). At the final stage, each total sum (excess phase estimate) is added to the input $\Delta\hat{\phi}_n[n]$ (delayed via shift registers) to output the unwrapped $\Delta\hat{\phi}_n^{un}[n]$.

978-1-5386-4881-0/18/$31.00 ©2018 IEEE

The weighting function module implements the approximated expression described in Eq. 5. More specifically, we employ four tables of three cells each to store the required coefficients, as well as 3 multiply-add units to compute the weighted magnitude. To parallelize the calculation of Eq. 5, we store the coefficient tables in four distinct memories to facilitate the fetching of all 4 requested coefficients in a single cycle. Moreover, we exploit the integrated DSP blocks of the Virtex-7 FPGA to map each multiply-add operation ($\alpha\chi+\beta$) onto a single DSP block, and hence, reduce the critical utilization of logic resources. To parallelize at the symbol level, similarly to CORDIC, we instantiate P independent "weighting functions".

Overall, the entire design is based on deep pipelining and fine-tuning of the critical paths at register-transfer level. The final architecture consists of up to $(120 + P)$ pipeline stages (latency) and delivers P outputs per clock cycle due to the sophisticated parallelization techniques described above. The synchronization/control of all operations is achieved via two simple shift registers forwarding global enable/reset signals (instead of using multiple signals traveling within the multiple computational pipelines). As a result, we decrease the register utilization and routing congestion. Finally, we highlight that the developed architecture is fully parameterizable with respect to I/O accuracy, word-lengths in the datapath of each component, coefficient bits and parallelism, P, in order to facilitate adaptation to various throughput and BER specifications.

IV. Evaluation Results

Our design was tested on a VC7222 board hosting a Xilinx Virtex-7 (VH580T) FPGA and it was further evaluated for the bigger Ultrascale+ XCUV9P device via Xilinx Vivado 2017.1. The functionality of the NLS algorithm was verified by applying it on sequences of 10^6 16-QAM modulated symbols generated in MATLAB and impaired by AWGN and laser phase noise that follows a Gaussian random walk.

Our parametric architecture allows us to explore the design space and determine the optimal cost-throughput-accuracy trade-off for each device/application. For the Xilinx CORDIC IP, the analysis leads to 17 pipeline stages for 11 bits of input and 10 bits of output accuracy. The accuracy was determined initially by a coarse-grain exploration with MATLAB's fixed-point toolbox, but in practice, VHDL fine-tuning showed that 2 extra bits were needed (the black-box CORDIC functions of MATLAB and Xillinx differ). A similar 2-stage word-length optimization for the datapaths of the key components leads to 10 bits of accuracy. Indicatively, for 2 extra bits of accuracy we get around 29% throughput degradation. Additionally, the length of the moving average filter was set to 32 taps for all the hardware configurations. Finally, by varying the parallelization parameter, P, we explore the number of input symbols that can be processed by each device while respecting throughput increase and device capacity. Table I includes representative configurations showing the scaling of the architecture with respect to FPGA resource utilization and throughput, for a given SNR penalty of $\leqslant 1$ dB.

For the VH580T, we get a maximum of $P = 119$ for the aforementioned accuracy parameters, with the achieved frequency decreasing gracefully from 357 MHz for $P = 1$ to 303 MHz for $P = 119$. Therefore, the proposed architecture can process up to 36 GBd when fully utilizing the VH580T or, e.g., 22.4 GBd for 50% LUT utilization: The throughput increases almost proportionally to HW resources. The majority of resources are devoted to CORDIC (59% LUTs and 100% BRAMs of the overall NLS), with the second most demanding module by far being the moving average filter with a 10-fold increase in LUT utilization when compared to that of the phase unwrap (17% LUTs of the NLS utilization). For XCUV9P, the maximum throughput increases to 64 GBd with 66% LUT utilization. Put into perspective, such execution is 200x faster than an Intel Core 2 Duo P7450 (1-thread, MATLAB).

TABLE I. Scaling of the proposed architecture on FPGA

Device	P	LUTs	DSPs	max f	GBd
VH580T	1	1.7K (0.49%)	3 (0.18%)	357MHz	0.36
VH580T	84	238.6K (65.78%)	252 (15%)	333MHz	28
VH580T	119	341.6K (94.18%)	357 (21.25%)	303MHz	36
XCUV9P	163	483.2K (40.8%)	489 (7.15%)	345MHz	56.2
XCUV9P	256	778.7K (65.8%)	768 (11.23%)	250MHz	64

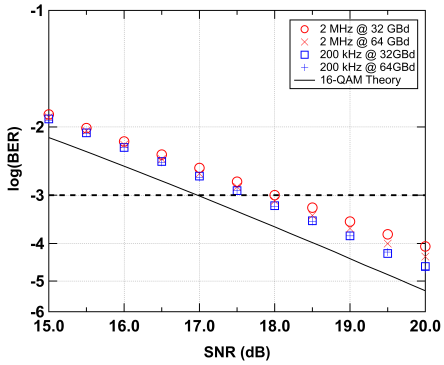

Fig. 6. BER vs SNR for the VHDL implementation for two different linewidth values and symbol rates. The theoretical limit of 16-QAM is also plotted.

The best implementation for each device was further explored with respect to the quality of CPR, as shown in Fig. 6. By varying both the SNR (AWGN noise), from 14 to 20 dB, and the laser phase noise from 200 kHz to 2 MHz, we get a maximum SNR penalty of 1 dB and a minimum of 0.6 dB at BER=10^{-3} (w.r.t the theoretical limit). Compared to similar works in the literature, the proposed implementation proves to be more tolerant to phase noise while achieving both the 32 and 64 GBd standards. Compared to the BPS FPGA implementation of [10], we demonstrate a better SNR penalty for 32 GBd 16-QAM operation ($\leqslant 1$ dB vs 2.3 dB).

V. Conclusions

We proposed an FPGA-based architecture of the NLS CPR algorithm, along with various sophisticated methods towards increased HW-efficiency and parallel processing. The proposed architecture is flexible enough to support diverse system requirements (throughput, BER) of current and future coherent optical systems. Experimental results based on 16-QAM revealed up to 36 and 64 GBd throughputs (on Virtex-7 VH580T and UltraScale+ XCVU9P FPGAs respectively), with a small ($\leqslant 1$ dB) SNR penalty.

978-1-5386-4881-0/18/$31.00 ©2018 IEEE

REFERENCES

[1] C. Laperle and M. OSullivan, "Advances in high-speed DACs, ADCs, and DSP for optical coherent transceivers," *Journal of Lightwave Technology*, vol. 32, no. 4, pp. 629–643, 2014.

[2] E. Ip and J. Kahn, "Feedforward Carrier Recovery for Coherent Optical Communications," *Journal of Lightwave Technology*, vol. 25, no. 9, pp. 2675–2692, 2007.

[3] T. Pfau, S. Hoffmann, and R. Noe, "Hardware-efficient coherent digital receiver concept with feedforward carrier recovery for M-QAM constellations," *Journal of Lightwave Technology*, vol. 27, no. 8, pp. 989–999, 2009.

[4] I. Fatadin, D. Ives, and S. J. Savory, "Laser Linewidth Tolerance for 16-QAM Coherent Optical Systems Using QPSK Partitioning," *IEEE Photonics Technology Letters*, vol. 22, no. 9, pp. 631–633, 2010.

[5] Y. Wang, E. Serpedin, and P. Ciblat, "Optimal blind nonlinear least-squares carrier phase and frequency offset estimation for general QAM modulations," *IEEE Transactions on Wireless Communications*, vol. 2, no. 5, pp. 1040–1054, 2003.

[6] N. Argyris, S. Dris, C. Spatharakis, and H. Avramopoulos, "High performance carrier phase recovery for coherent optical QAM," in *Optical Fiber Communications Conference and Exhibition*, 2015, pp. 1–3.

[7] K. Maragos, C. Spatharakis, G. Lentaris, P. Kontzilas, S. Dris, P. Bakopoulos, H. Avramopoulos, and D. Soudris, "A flexible, high-performance FPGA implementation of a feed-forward equalizer for optical interconnects up to 112 Gb/s," *Journal of Signal Processing Systems*, vol. 88, no. 2, pp. 107–125, 2017.

[8] B. Baeuerle, A. Josten, F. Abrecht, M. Eppenberger, E. Dornbierer, D. Hillerkuss, and J. Leuthold, "Multi-format carrier recovery for coherent real-time reception with processing in polar coordinates," *Opt. Express*, vol. 24, no. 22, pp. 25 629–25 640, 2016.

[9] S. Randel, S. Corteselli, D. Badini, D. Pilori, S. Caelles, S. Chandrasekhar, J. Gripp, H. Chen, N. K. Fontaine, R. Ryf *et al.*, "First real-time coherent MIMO-DSP for six coupled mode transmission," in *Photonics Conference (IPC), 2015*. IEEE, 2015, pp. 1–2.

[10] B. Baeuerle, A. Josten, F. C. Abrecht, E. Dornbierer, J. Boesser, M. Dreschmann, J. Becker, J. Leuthold, and D. Hillerkuss, "Multiplier-free, carrier-phase recovery for real-time receivers using processing in polar coordinates," in *Optical Fiber Communication Conference*. Optical Society of America, 2015, pp. W1E–2.

978-1-5386-4881-0/18/$31.00 ©2018 IEEE