Fiber-on-Chip: Digital Emulation of Channel Impairments for Real-Time DSP Evaluation

Citation for the original published paper (version of record):
Börjeson, E., Larsson-Edefors, P. (2023). Fiber-on-Chip: Digital Emulation of Channel Impairments for Real-Time DSP Evaluation. Journal of Lightwave Technology, 41(3): 888-896.
http://dx.doi.org/10.1109/JLT.2022.3200248

(article starts on next page)

Fiber-on-Chip: Digital Emulation of Channel Impairments for Real-Time DSP Evaluation

Erik Börjeson, *Student Member, IEEE* and Per Larsson-Edefors, *Senior Member, IEEE*

***Abstract*—We describe the Fiber-on-Chip (FoC) approach to verification of digital signal processing (DSP) circuits, where digital models of a fiber-optic communication system are implemented in the same hardware as the DSP under test. The approach can enable cost-effective long-term DSP evaluations without the need for complex optical-electronic testbeds with high-speed interfaces, shortening verification time and enabling deep bit-error rate evaluations. Our FoC system currently contains a digital model of a transmitter generating a pseudo-random bitstream and a digital model of a channel with additive white Gaussian noise, phase noise and polarization-mode dispersion. In addition, the FoC system contains digital features for real-time control of channel parameters, using low-speed communication interfaces, and for autonomous real-time analysis, which enable us to batch multiple unsupervised emulations on the same hardware. The FoC system can target both field-programmable gate arrays, for fast evaluation of fixed-point logic, and application-specific integrated circuits, for accurate power dissipation measurements.**

I. Introduction

A common way to evaluate digital signal processing (DSP) algorithms for communication systems is to simulate the complete end-to-end system in software on a computer. Here, the transmitter and channel are modeled as floating-point algorithms which generate data to some DSP functions which are represented as a floating-point algorithm. This approach has the benefit of short evaluation times. However, to accurately evaluate a DSP hardware implementation, circuit simulations must be performed. Here, the simulated channel output can be fed, cycle by cycle, to a software simulation of a fixed-point DSP circuit description. This method makes it possible to correctly capture the bit- and cycle-level behavior of the circuit, accounting for aspects such as algorithm approximations and digital logic optimizations. Additionally, since signal switching statistics are available, credible estimations of power dissipation can be performed. However, simulation of detailed circuit representations requires very long run-times, making it difficult to evaluate the long-term behavior and perform deep bit-error rate (BER) analysis of a DSP system.

On the other side of the spectrum, we find real-time optical communication experiments. This approach typically requires complex testbeds with both optical and electronic components, possibly also including mechanical components to emulate time-varying properties like polarization changes [1], [2]. The added complexity makes real-time experiments more expensive and more complicated to set up, especially when the DSP functionality is implemented in application-specific integrated circuits (ASICs) [3] instead of the more common method of offline processing. For the latter, a significant disadvantage is the limited memory of the oscilloscopes typically used to store the output data snapshot. This makes it hard to save continuous output waveforms, which hampers long-term analyses.

E. Börjeson and P. Larsson-Edefors are with the Department of Computer Science and Engineering, Chalmers University of Technology, 41296 Gothenburg, Sweden (e-mail: erikbor@chalmers.se, perla@chalmers.se).

As an extension to our invited OFC 2022 contribution [4], we describe in this paper the Fiber-on-Chip (FoC) approach which represents a middle-ground between the two previously described approaches and which can be used for real-time evaluation of coherent receiver DSP circuit implementations. Based around on-chip digital models of both transmitter and channel, the FoC system aims to move time-consuming circuit simulations in software to circuits emulated on dedicated hardware. Here, the receiver DSP circuits—the *DSP under test (DUT)*—are co-located with the transmitter and the channel on either an ASIC or a field-programmable gate array (FPGA). All digital impairment models are under software control, which makes is possible to set up long-term, unsupervised evaluation runs with dynamic control of the parameters in the system. In addition, the output of the DUT can be continuously monitored and stored in memory.

A. Fiber-on-Chip (FoC)

Our first FoC version [5] emulated a single polarization system, including additive white Gaussian noise (AWGN), to control the signal-to-noise ratio (SNR), and phase noise, to facilitate long-term evaluations of the cycle-slip rate of carrier phase recovery circuits [6]. In later contributions, more extensive analysis tools as well as support for dual-polarization (DP) systems and polarization-mode dispersion (PMD) emulation were added, to enable equalizer evaluations based on time-varying impairments [4], [7].

An overview of an FoC system, in which a DUT is integrated, is shown in Fig. 1. All operations are performed onboard the FPGA/ASIC using fixed-point arithmetic and parameters can be adjusted from a computer using a low-speed communication interface. Thus, high-speed data interfaces can be completely avoided, simplifying prototyping work. To make it possible to evaluate complex DUTs, the hardware resources used by the FoC system should be minimized. This means that some FoC parameters, such as filter lengths, need to be hardcoded. Other parameters, such as the amount of AWGN or PMD rotations, can be controlled during run-time.

Inside the FoC system, a pseudo-random bitstream is modulated and the symbols are pulse shaped before being fed to the channel emulator. The random-number generator (RNG),

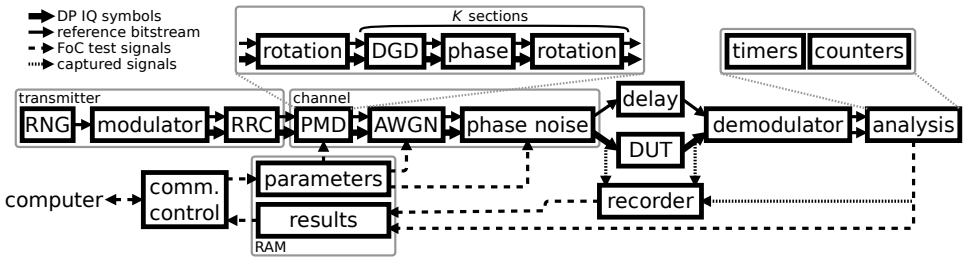

Fig. 1: Block diagram of an example FoC system used to evaluate the DSP under test (DUT).

which is used to generate the bitstream, and the Gaussian random number generators (GNG), which are used to emulate some of the impairments, have a very long periodicity, creating a virtually endless stream of symbols at the output of the digital channel model. After processing in the DUT and demodulation of the symbols, the resulting bitstream can be compared to a reference bitstream, which is propagated in a parallel channel from the RNG through the complete system. Typically, the number of transmitted and erroneously demodulated bits are counted to calculate the BER, but other analysis tools are also available, such as timers to measure, e.g., the convergence speed of an equalizer, and recorders to capture any signal in the system for later software analysis.

The system is written in a hardware description language (HDL) called VHDL and is mainly targeting Xilinx FPGAs, but ASIC versions have also been developed, manufactured and tested. The fixed-point HDL description has been verified against a floating-point MATLAB model of the same system, and the HDL code has been released as the Chalmers Optical Fiber Channel Emulator (CHOICE)[1]. In fact, Fig. 1 represents the FoC system of CHOICE.

B. Similar Approaches

When evaluating forward error-correction (FEC) circuits for optical communication, it is very challenging to reach the required, very low post-FEC BERs. We have previously used FPGA-based emulation schemes to evaluate soft-decision low-density parity check FEC implementations [8], [9]. In these works, RNGs are used to generate the log-likelihood ratios at the input of the FEC, thus the generation of a complete channel can be avoided. ZTE has presented many impressive FPGA-based FEC experiments over the years. For example, FEC solutions for OIF-800GZR have been evaluated using a large number of FPGAs to reach very low BERs within short run-times [10]. Here, a complete transmitter is implemented on FPGAs, to generate a pseudo-random input bitstream, perform FEC encoding, interleaving and symbol mapping. An AWGN channel is used before the receiver and FEC decoder. It should be noted that in the context of system emulation, FEC has one big advantage over DSP: FEC operates on demodulated binary data, which are much easier to emulate in the digital domain than modulated data.

For DSP evaluation, Fraunhofer's high-speed digital signal processing platform is available [11]. This is a modular system which uses FPGAs to perform signal processing on real channel data. For other applications, FPGA emulation of wireless channels has also been suggested, e.g., for Wi-Fi [12] and for autonomous vehicle control channels [13]. Radar development is another application where FPGA emulation is used to accelerate evaluation of target signatures [14].

C. Outline

Section II contains a short introduction to the most relevant aspects of digital circuit design, followed by descriptions of the parts of the CHOICE FoC system in Section III–V. The resource utilization for an FPGA implementation of CHOICE is given in Section VI. Before concluding the paper in Section VIII, two case studies, demonstrating one FPGA- and one ASIC-based FoC implementation, are presented in Section VII.

II. Digital Implementation on ASIC and FPGA

To enable a real-time end-to-end communication system, the transmitter and channel emulators share the hardware resources with the DSP under test (DUT). To maximize the capabilities of the FoC system, the digital hardware implementation of the emulators needs to be optimized in terms of resources. But depending on whether we target an ASIC or an FPGA, this optimization process is different.

Floating-point representation is common in, e.g., scientific computing, but for embedded applications which require high throughput, low latency, and low resource usage, fixed-point circuits are preferred. The use of a fixed number of bits, *the wordlength*, to represent a digital word is very hardware efficient, but it makes the implementation process challenging. This is because we need to consider, for each fixed-point signal, what the wordlength should be (as short as possible to save hardware) and where the binary point should be located to keep the desired information within the chosen data wordlength. Fig. 2a shows a simplified example of an unsigned 8-bit word with two integer bits[2]. With the general goal to use as few bits of information as possible to save hardware, a significant part of DSP hardware implementation is spent on *wordlength and scaling optimization*. Fig. 2b shows an example of the binary multiplication of two digital words which leads to wordlength growth at the output. In DSP where multiplications are very common, it is not sustainable to keep

[1]A GIT repo of the CHOICE FoC environment is available at https://www.cse.chalmers.se/research/group/vlsi/choice.

[2]In fact, our fixed-point implementations use signed numbers based on the widely used two's complement representation.

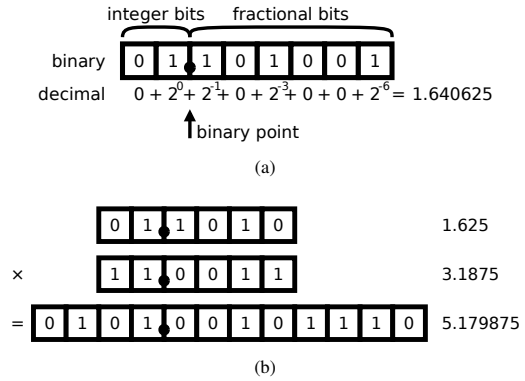

Fig. 2: Illustration of (a) an 8-bit unsigned fixed-point data word with two integer bits and 6 fractional bits, and (b) an unsigned multiplication of two 6-bit data words.

the full multiplication output resolution but the output must be truncated.

FPGAs are made up of hardware modules of greater complexity than the logic gates of ASICs and this impacts the implementation optimization. While it is in general beneficial to reduce the wordlength, reducing it with only one bit is likely to have no effect on an FPGA system that uses DSP slices with, by default, tens of bits per input. In contrast, digital ASICs are made up of fine-grain elements such as 2-input logic gates. Thus, here it pays off to optimize the wordlength aggressively, since this allows for significant circuit area reductions during the *synthesis phase* which translates HDL code down to logic gates.

There are two other aspects of digital implementation often ignored when algorithms are designed: *pipelining* and *parallelism*. The critical path in a digital circuit is the longest path between two delay elements (registers), as shown in Fig. 3a. This is the path that limits the maximum clock rate of a digital system, since the signal needs to propagate through all logic elements on the path in one clock cycle. By inserting pipeline stages, i.e., additional delay elements, on this path, as in Fig. 3b, the clock rate can be increased at the cost of increased cycle latency. An added benefit is that pipelining reduces the probability of glitches; short unwanted signal toggles that increase power dissipation. The approach to pipelining differs somewhat between FPGA and ASIC implementations, since the former usually includes pipelining stages in the DSP slices. These can often be used without incurring an extra resource penalty, if the DSP slices are fully utilized.

Parallel processing is a way to increase data throughput by duplicating functional elements, as shown in Fig. 3c, thus enabling the processing of multiple data samples per clock cycle. Parallelism is necessary for systems where the symbol rate is faster than the clock rate of the signal processing system, which is the case for most fiber-optic communication systems. A drawback of parallel processing is the larger amount of resources occupied by the DSP hardware; as the degree of parallelism increases, the resources consumed can quickly become prohibitively large. Both pipelining and parallel implementation of a circuit are relatively straightforward for feed-forward algorithms, but for systems with a feedback path, such as an adaptive equalizer, the latency added by pipelining can become so large that it impacts performance. Contrary to DSP circuits, parallel processing is not strictly necessary for the FoC system, but it is just another option to further increase the emulation speed.

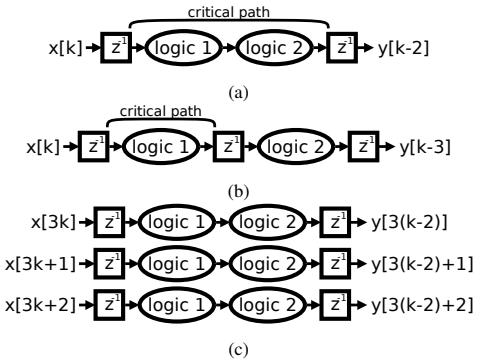

Fig. 3: Illustration of how the critical path in (a) can be shortened by insertion of a pipeline register (b). A 3x-parallel implementation of the same logic is shown in (c).

III. Transmitter

The transmitter part of the CHOICE FoC system shown in Fig. 1 consists of an RNG, a modulator and an optional pulse-shaping filter. The RNG generates a pseudo-random bit sequence (PRBS) using linear-feedback shift registers [15]. The internal wordlength of the shift registers, k, is adjustable, and the resulting periodicity is $2^k - 1$ bits. In this paper, we use $k = 64$, which gives a periodicity of $\approx 10^{19}$. The modulator currently supports Gray-coded BSPK, QPSK, 16QAM, 64QAM, and 256QAM, but can easily be extended to other formats. The modulator outputs both the symbols and the corresponding bits, which are fed as a bitstream through all succeeding processing modules to be used as a reference after demodulation.

When using the PMD emulator to evaluate an equalizer, the modulated signal needs to be upsampled and pulse shaped. This is performed in a root-raised cosine (RRC) filter which supports 2x upsampling. The filter is realized as a symmetric direct-form FIR filter with constant coefficients. The use of constant filter taps has the benefit of allowing the synthesis tool to infer, on FPGAs, add-and-shift operations instead of precious DSP slices or, on ASICs, real multipliers [16].

IV. Channel Emulation

The emulation of the channel shown in Fig. 1 consists of three separate impairment emulators that can be used either separately or in combination: PMD, AWGN and phase noise. The AWGN and phase noise generators have previously been outlined in [5], while the PMD emulator is an extension of the work done in [7]. These impairments were selected as the CHOICE environment was originally designed to evaluate new carrier phase recovery and equalization circuits.

A. Polarization Mode Dispersion

PMD can be modelled as a concatenation of K birefringent fiber sections described by its Jones matrices [17]. The result is the combined transfer function

$$N(\omega) = \prod_{k=1}^{K} \overbrace{\begin{bmatrix} \cos\theta_k & \sin\theta_k \\ -\sin\theta_k & \cos\theta_k \end{bmatrix}}^{\text{polarization rotation}} \overbrace{\begin{bmatrix} e^{j\phi_k} & 0 \\ 0 & e^{-j\phi_k} \end{bmatrix}}^{\text{phase rotation}} \overbrace{\begin{bmatrix} e^{j\omega\tau_k/2} & 0 \\ 0 & e^{-j\omega\tau_k/2} \end{bmatrix}}^{\text{DGD}}, \tag{1}$$

where θ is the polarization rotation angle, ϕ is the phase rotation angle, and τ is the differential group delay (DGD). One section of this model will excite two of three degrees of freedom of the Poincaré sphere, but after a few sections all points on the sphere can be reached. As shown in Fig. 1, our PMD emulator uses the same functional separation, implementing each matrix in a separate processing module.

Polarization Rotation: The polarization rotation is described by the first matrix in (1) and a block diagram of our emulator circuit implementation is shown in Fig. 4a. A look-up-table (LUT) indexed by the θ input is used to output $\sin\theta$ and $\cos\theta$, which are used as inputs to the matrix multiplication. As outlined in Section II, the unavoidable wordlength growth must be controlled; in this case by rounding back to the input wordlength after the addition.

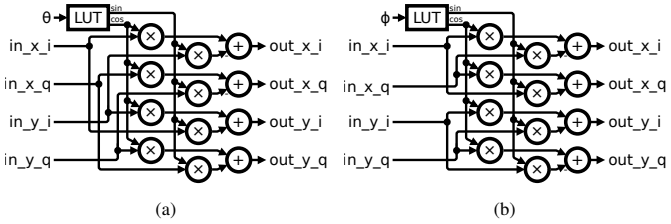

Fig. 4: Simplified block diagrams showing a single lane of (a) the PMD polarization rotation and (b) the PMD phase rotation blocks.

A fixed-point implementation approximates an ideal algorithm. Fig. 5 shows the maximum error magnitude for varying fixed-point wordlengths compared to a floating-point implementation, using random symbol data and rotation angles θ. The results show that using rotation-angle wordlengths larger than the symbol wordlength + 2 bits does not improve performance. Above this limit, the error is caused by the limited resolution of symbol data, and is smaller than the least-significant bit (LSB) of each I/Q signal component.

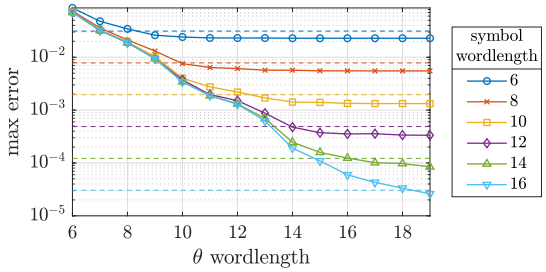

Fig. 5: Maximum error of the fixed-point output signals from the PMD polarization rotation implementation compared to a floating-point version. The dashed lines show the value of the LSB of the symbols.

Phase Rotation: The phase rotation is described by the second matrix in (1). Fig. 4b shows a block diagram of the emulator circuit implementation. Not only is the implementation very similar to the polarization rotation block, but its fixed-point behavior is almost identical to that of the polarization rotation shown in Fig. 5.

Differential Group Delay: A function of the optical frequency (ω), the DGD is described by the third matrix in (1). This frequency dependency implies that the straightforward way of implementing the DGD in a circuit would be in the frequency domain. However, since the polarization and phase rotations are more easily implemented in the time domain, a large number of FFT/IFFT blocks would be needed. In order to keep the hardware resource usage of a complete FoC system as low as possible, we choose to approximate the DGD in the time domain using fractional-delay FIR filters.

Laakso et al. [18] provide descriptions of many of the methods available to calculate the coefficients of a DGD filter. In our case, Lagrange interpolation was deemed most suitable because of the smooth magnitude response, especially at low frequencies, and the fact that the coefficients can be calculated with relative ease using

$$h[n] = \prod_{k=0,k\neq n}^{N} \frac{D-k}{n-k} \text{ for } n = 0, 1, 2, ...N, \tag{2}$$

where N is the filter order and D is the fractional delay.

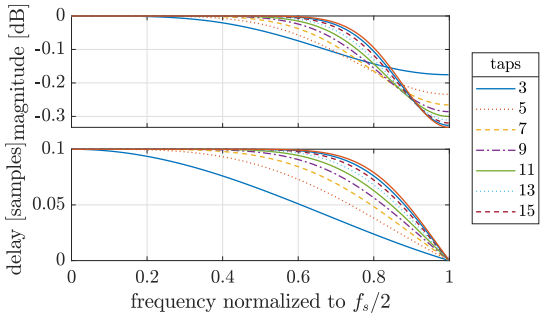

Fig. 6: Frequency response of floating-point Lagrange-interpolation fractional delay filters for different tap counts at delay $d = 0.1T_s$.

However, the time-domain approximation of the DGD results in a frequency dependency in both the magnitude and the delay caused by the finite length of the filter, as shown in Fig. 6. An odd number of taps is used, since that results in a flatter magnitude response for small delays. The approximation of the fractional delay is excellent at lower frequencies, but the bandwidth grows slowly with increasing filter order [18].

As shown in Fig. 7, the DGD filter is implemented as a 2x-parallel, transposed FIR filter. The filter is extensively pipelined to match the architecture of the DSP slices available in the FPGA platforms that we mainly target. The full resolution of the multiplier outputs are used for all summations, while the sum is rounded to the appropriate wordlength before being output.

To see how the fixed-point filter response affects the band-limited transmitted symbols, simulations of a 16QAM transmission was carried out, where a $0.1T_s$ DGD was applied to

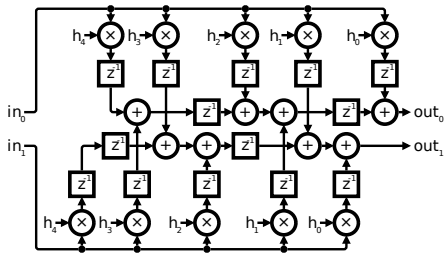

Fig. 7: Simplified block diagram of the PMD delay block for a 2x-parallel 5-tap filter, where h represents the filter taps.

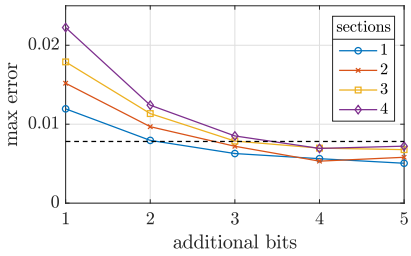

Fig. 9: The effect of using extra bits for the internal data representation in the PMD emulator, shown for 11-tap DGD filters. The dashed line marks the value of the LSB for the 8-bit symbol representation.

the 8-bit quantized symbols either using floating-point math in the frequency domain or using our HDL time-domain implementation. The symbols were upsampled 2x in a 51-tap root-raised cosine pulse-shaping filter, with $\beta = 0.1$, before being fed to the two implementations. Fig. 8 shows that using 7 taps or more for the time-domain filter and ≥ 10 bits for the wordlength results in an error smaller than the value of the LSB bit of the symbols. In general, a larger DGD requires a larger number of taps to keep the error low, since the outer filter coefficients become larger. However, since they affect the fixed-point quantization error, the actual coefficient values also matter.

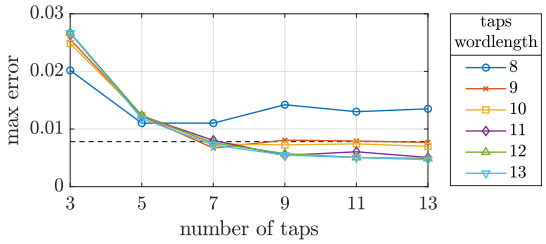

Fig. 8: Maximum difference between a floating-point frequency-domain DGD calculation and a fixed-point time-domain HDL implementation using an 8-bit 16QAM signal. The value of the LSB of the symbol representation is marked with a dashed line.

Because of the time-domain approximation used for the DGD, the combination of parameters for the pulse-shaping filter and the DGD filter must be carefully selected. If a larger roll-off factor is needed for the pulse-shaping filter, the order of the DGD filter needs to be increased. Additionally, other impairments affecting the higher frequencies of the spectrum, such as AWGN, should be added after the PMD, to avoid affecting their frequency response.

Multiple PMD Sections: Even though care is taken to minimize the difference between the time-domain implementation and a frequency-domain floating-point reference model for each PMD component, the rounding errors will inevitably accumulate when concatenating multiple PMD emulator modules in multiple sections. These errors are due in part to the limited resolution of the fixed-point representations and in part to the time-domain approximation of the DGD.

To reduce the first effect, the wordlength of the signals used in the PMD emulator can be increased, essentially hiding the rounding errors in the LSBs that will be removed before the next stage. Fig. 9 shows how a time-domain PMD emulator for QPSK, using 11-tap DGD filters with $\tau = 0.1T_S$, and random, static settings for θ and ϕ, is affected by increasing the internal wordlength for 1–4 sections. For ≤ 2 sections, it is enough to use 3 additional bits, but for higher number of sections more bits are needed to keep the error lower than 1 LSB. However, this number is highly dependent on the τ used. For > 5 sections, longer DGD filters are necessary if we want to keep the difference between the two models small. If the target architecture is an FPGA, additional bits can be added without a significant increase in resource utilization, as long as the longest internal wordlength does not exceed the wordlength of the dedicated DSP slices. For an ASIC implementation, this solution would incur an area penalty.

Increasing the number of taps used in the DGD filters will decrease the time-domain approximation error, at the cost of extra multipliers. Since the filters are 2x-parallel and applied on the I/Q components on both polarizations, this corresponds to 8 extra multipliers (or FPGA DSP slices) per added filter tap.

The low-pass response of the DGD filter also affects how time-dependent impairments can be added to the FoC system. If the PMD polarization rotation and/or phase rotation is emulated as, e.g., a random walk, the high frequency content of the signals will be increased, reducing the accuracy of the DGD approximation. The effect is especially pronounced for fast changes of θ and ϕ and for short DGD filters.

B. Additive White Gaussian Noise

An open-source Gaussian noise generator (GNG) IP [19] is used as the basic building block for the AWGN emulation block. The GNG generates a 64-bit uniformly-distributed pseudo-random number every clock cycle [20], and the resulting uniform distribution is transformed to a Gaussian distribution using the inverse cumulative distribution function

$$\mathrm{ICDF}(x) = \sqrt{2}\mathrm{erf}^{-1}(2x-1). \tag{3}$$

To reduce the area of the GNG circuit, the ICDF is implemented using a piecewise polynomial approximation and a non-uniform segmentation scheme similar to [21]. The result is a random distribution in the range $\pm 9.1\sigma$ with a periodicity of approximately 2^{176} [19].

A block diagram of the AWGN block is shown in Fig. 10 for one single lane, which needs two GNGs; one for each complex dimension. When a higher degree of parallelism is needed, this circuit is repeated for all lanes but with different seeds for all GNGs. The *scaling* signal controls the SNR by

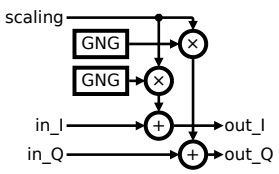

Fig. 10: Block diagram of the AWGN emulator for a single lane.

adjusting the amount of AWGN added to the inputs, and can in a real-time experiment be changed during run-time.

C. Phase Noise

Phase noise can be modelled as a Wiener process,

$$\theta_i = \theta_{i-1} + \Delta_i, \tag{4}$$

where θ_i is the phase of the ith sample and Δ_i is a normally distributed random variable with zero mean and a variance $\sigma^2_{\Delta_i} = 2\pi\Delta f T_s$. The variance is a function of Δf, which is the combined linewidth of the transmitter and local oscillator lasers, and T_s, which is the symbol duration.

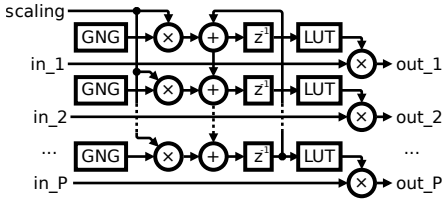

Fig. 11: Block diagram of the phase noise emulator for P parallel lanes. Note that the inputs and outputs are I/Q signals.

An overview of the circuit implementation is shown in Fig. 11, for P parallel lanes. The same type of GNG is used as for the AWGN block and the GNG outputs are scaled with the linewidth symbol-duration product ($\Delta f T_s$), which is given by the *scaling* input and can be updated during run-time. Different seeds are used for each GNG. The resulting phase offset for each parallel lane (Δ_i) is summed with the previous phase to calculate θ_i for each input symbol. Finally, a LUT is used to convert the phase to a complex number of unit length, which is multiplied with the input symbol to realize the phase rotation.

Since the first lane needs the previous phase offset from the last lane, a feedback loop is created which is inherent to the Wiener process. The critical path of this loop limits the number of parallel lanes that can be used for the phase noise block for a given clock rate.

V. DSP Under Test and Analysis

As shown in Fig. 1, the channel emulator drives the DUT which in principle can be made up of any receiver DSP functionality. Depending on the DSP functions selected, the relevant digital channel impairment models are chosen and used in the real-time evaluation.

Downstream from the DUT's output, the demodulator maps the symbols to the corresponding bits, which are subject to further analysis. For example, the reference bits from the RNG can be used to compare the reference and the DUT output bitstreams and count the number of erroneously decoded bits. A second counter can be used to store the total number of processed bits and the result can be downloaded to a computer to calculate the BER. This post-processing operation is better performed outside of the FoC system, since the division is complex to implement in digital hardware.

The FoC system also contains a timer that counts the number of clock cycles that has passed until a certain event happens. This feature can be useful to capture, e.g., equalizer convergence by stopping the timer when the average equalizer error drops below a set threshold.

The recorder block in Fig. 1 can be used to capture any signal inside the system before, after or around a trigger point. Currently, the amount of free block SRAM memory on an FPGA is limiting the number of samples that can be captured, but the functionality can be extended to use external DRAM memory for the data, which would enable capture of very long data series.

TABLE I: Synthesis results for a QPSK FoC system including AWGN, phase noise and 5 sections of PMD. The percentage of the total available resources is shown in parenthesis where relevant. DUT resources are not included.

Component	LUT	CARRY8	DSP
Transmitter	5346	808	0
RRC ×2	2578	404	0
Channel	18013	134	502 (28%)
PMD	6894	0	456 (25%)
Rotation ×6	587	0	16 (0.9%)
DGD ×5	72	0	56 (3.1%)
Phase ×5	586	0	16 (0.9%)
AWGN	2415	88	24 (1.3%)
Phase noise	8704	46	22 (1.2%)
Analysis	123	16	0
Control etc.	63345	66	0
Total	86582 (8%)	1024 (0.8%)	502 (28%)

VI. FPGA Resource Usage

To estimate the resource utilization of the FoC system presented above, a complete DP-QPSK emulator was synthesized targeting the Xilinx VCU110 development board [22], featuring a Virtex Ultrascale XCVU190 FPGA. The implemented system follows the organization in Fig. 1, excluding the recorder and trigger components. It uses a single lane, which becomes 2x-parallel after pulse shaping and oversampling in a 51-tap RRC filter. Each I/Q component of the two polarizations is represented by 8 bits. The channel emulation includes AWGN, phase noise and five PMD sections. The PMD DGD filters use 11 taps, and the internal signals of the PMD emulator use 3 additional bits, to reduce the fixed-point errors as described above. A 100-MHz clock is used, resulting in an emulation speed of 400 Mbit/s.

The FPGA resources used for each of the larger components are shown in Table I, which clearly indicates that the DSP slices are the limiting factor rather than the configurable logic blocks (CLBs), which contain LUTs, registers, carry chains etc. The DSP slices are mainly used for the many multiplications necessary in the channel emulation, especially in the DGD filters. The benefit of using constant filter-tap values is shown when comparing DGD to RRC filters. Even

though $26 \cdot 2 = 52$ multiplications need to be performed for each 51-tap symmetric RRC filter, the logic is simplified to shift-and-add operations, utilizing the LUTs and CARRY8 chains included in the CLBs instead of the DSP slices. If run-time control of the DGD is not necessary, this static solution can be used also for the DGD filters, resulting in a significant reduction of the resources occupied by the FoC system.

VII. Case Studies

As a demonstration of the capabilities of a CHOICE FoC approach, this section will describe both an FPGA implementation and a manufactured ASIC design. We have used these designs to explore the properties of a DSP subsystem for QPSK; the DUT consists of a 9-tap adaptive CMA equalizer followed by a 4th-power Viterbi-Viterbi (VV) phase estimator. The channel emulation consists of an AWGN generator and a 10-section PMD emulator, using a DGD of 0.05 samples/section. The polarization rotation for each section is set to use a time-dependent rotation of 10 to 100 krad/s with random start values and rotation direction. All other channel impairments are assumed to be fully compensated by other DSP units. Due to the different hardware platforms used for the two examples, the two circuits are not an exact one-to-one match.

The CMA equalizer can converge to output the transmitted X polarization at either the X or at the Y output, with the transmitted Y polarization at the other output. Additionally, the symmetry of the QPSK constellation makes it possible for the VV unit to lock the phase tracking in steps of $\pi/2$. The result is eight different potential DSP output states. To avoid introducing pilot symbols in the system, which would enable detection of the correct state at the cost of increased circuit complexity, we choose to use eight demodulators instead, each with a separate error counter. The demodulators and counters are relatively small, so the additional area is negligible compared to other parts of the system. Each demodulator is configured for one output state and by looking at the counter with the smallest number of errors, we can find the state to which the system has converged.

One of the main motivations for using an FoC system is evaluation speed-up. In Fig. 12, we plot BER curves for the two systems, capturing at least 10 bit errors per data point. Generation of these curves took just over an hour for the FPGA and less than 15 minutes on the ASIC, due to its faster clock rate. A simulation of a software model of the circuit, using a Dell PowerEdge server, would require more than 6 months to process the same number of bits.

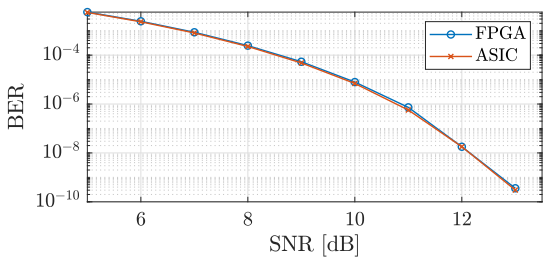

Fig. 12: BER of a DUT captured after equalizer convergence on an FPGA and an ASIC, respectively.

A. FPGA

The FPGA can be used for fast real-time evaluation of circuit properties and Fig. 13 shows constellation diagrams of the input and the output of the DUT. The symbols are captured using the recorder component after equalizer convergence, which is detected by comparing the average equalizer error signal with an adjustable threshold.

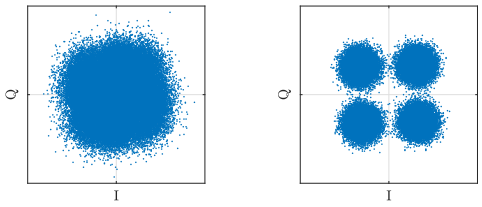

Fig. 13: Symbol constellations at (a) the input and (b) the output of the DUT subsystem, using an SNR of 10 dB.

Since the digital CMA equalizer circuit is designed to be partially programmable using logic signals, its step size can be adjusted during run-time. This further accelerates analysis of the convergence behavior, since no time-consuming re-synthesis is necessary between emulation runs. By capturing how much the bit-error counters have increased over the last 128 transmitted bits, the bit-error probability can be plotted as a function of the number of transmitted samples, as shown in Fig. 14, for four different step sizes. To remove the effect of $\pi/2$ phase jumps at the VV output during the equalizer convergence phase, where it has trouble locking, each data point shows the value of the error counter resulting in the smallest error. These runs use the same PMD settings as Fig. 13, and an SNR of 5 dB.

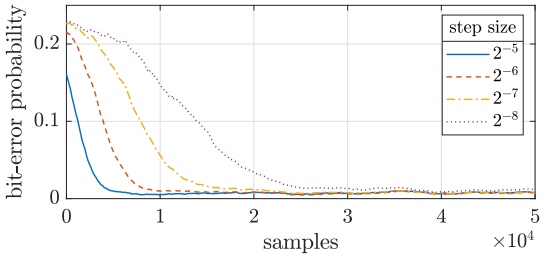

Fig. 14: Bit-error probabilities shown for the convergence phase of the adaptive CMA equalizer used in the DSP subsystem. The data have been smoothed using a moving average after downloading from the FPGA.

A simple event generator circuit allows us to temporarily increase the polarization rotation rate to study how the equalizer responds to PMD changes, called PMD events. During such a PMD event, we set the polarization rotation in four of the ten sections to 1 Mrad/s for a set duration. We use the recorder to capture the bit-error counters around these events, which takes place deep into an emulation run, and download the results to a computer for calculation of the bit-error probability and smoothing using a moving average. An SNR of 6 dB was used for these runs, as a slightly higher SNR was necessary to avoid losing the VV phase lock during the PMD event. The results are shown in Fig. 15.

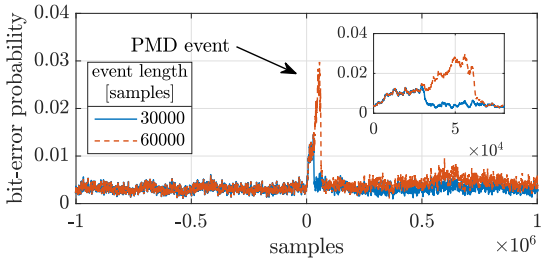

Fig. 15: Bit-error probabilities around a PMD event, smoothed with a moving average after downloading from the FPGA.

B. ASIC

An ASIC version of the FoC system was developed and manufactured on the side of a multi-project wafer in a 22-nm fully-depleted silicon-on-insulator (FD-SOI) CMOS process. Compared to the FPGA version, minor updates to the HDL code were necessary to account for, mainly, a different pipelining strategy. A cropped chip photo of the system is shown in Fig. 16, illustrating how the different subsystems are placed. The total cell area of this implementation is 0.24 mm^2, where the equalizer accounts for 0.082 mm^2.

Not only is it difficult to separate out the power dissipated in distinct modules on an FPGA board, but the fact that FPGAs dissipate considerably more power than their ASICs counterparts makes them unsuitable for evaluations of DSP power dissipation. In contrast, by evaluating an ASIC FoC we can obtain power and energy dissipation numbers that show the true potential of gate-level customization of DSP functions.

As in the FPGA system above, the DSP DUT is made up of an equalizer followed by an phase estimator. To track the power dissipation of the equalizer, this is placed in a separate power domain with a dedicated power supply net, leading to a dedicated chip pin. The power dissipation is directly related to the switching frequency of the logic gates, which means that having access to reliable input data is key to producing credible power measurements. Using the FoC system to generate the input symbols on-chip obviates the need to implement high-speed optical-electronic interfaces and to set up complex optical experiments. The benefit is significantly faster system setup and easy dynamic variation of system parameters.

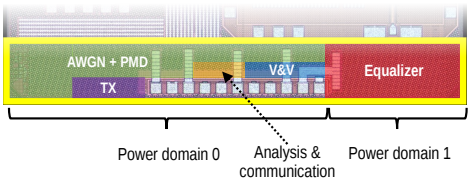

Fig. 16: Chip die photo of an ASIC implementation of an FoC system, with annotated modules.

VIII. Conclusion

The Fiber-on-Chip (FoC) approach to DSP hardware verification is based on models of fiber-optic transmission system components in fixed-point sequential hardware. In our CHOICE FoC system, these digital models include the transmitter itself, with pseudo random data generation, fiber channel impairments, demodulation and multiple analysis tools. Currently, impairment models include AWGN, phase noise and PMD. These models can be used when evaluating circuit implementations of DSP algorithms, where they can replace optical and electronic components with an emulator running on the same hardware as the DSP, either on FPGAs or on ASICs.

Since the impairments are modelled in the time domain using limited resolution, there are limitations in how closely they match a floating-point model. This effect is especially pronounced for the DGD emulation included in the digital PMD model. Two methods to reduce the difference between the digital fixed-point model and the floating-point model have been presented: temporarily increasing the resolution for the PMD emulator and increasing the number of filter taps in the DGD filter.

Implementing the channel model on the same hardware as the DSP increases the evaluation speed with orders of magnitude, compared to logic simulation of bit-accurate circuit models on a computer. This speed-up enables deep-BER evaluation of circuits and analysis of rare phenomena, such as cycle slips for carrier phase estimation implementations. When including an FoC system on an ASIC, the emulator can be used to generate meaningful input data to the DSP under test, enabling accurate measurements of ASIC power dissipation without the need for complex external test equipment and optical-electronic testbeds.

The parameters of the FoC system can be programmed from a computer, allowing for multiple different simulation scenarios, each with dynamically updated and time-varying impairments. Since these run-time updates can be performed in one and the same FoC hardware implementation, we avoid the time-consuming re-synthesis between emulation runs. The FoC system also includes modules for monitoring internal states of the DSP under test, which facilitates debugging. The pseudo-random bitstream and impairment generators ensure reproducible results and repeatability of evaluations.

Possible future extensions to our FoC system include other digital models, such as frequency offset, I/Q skew and the Kerr non-linearity, as well as feature additions, such as pattern memories to store waveforms from optical experiments. The latter will make it possible to switch between synthetic and experimental data.

Acknowledgements

We thank Magnus Karlsson and Mikael Mazur for useful discussions, Victor Åberg and Lars Svensson for support with ASIC tape-out and GlobalFoundries for chip fabrication.

References

[1] T. Pfau, C. Wordehoff, R. Peveling, S. K. Ibrahim, S. Hoffmann, O. Adamczyk, S. Bhandare, M. Porrmann, R. Noe, A. Koslovsky, Y. Achiam, D. Schlieder, N. Grossard, J. Hauden, and H. Porte, "Ultra-fast adaptive digital polarization control in a realtime coherent polarization-multiplexed QPSK receiver," in *Opt. Fiber Commun. Conf. (OFC)*, 2008, p. OTuM3.

[2] S. Yamamoto, T. Inui, H. Kawakami, S. Yamanaka, T. Kawai, T. Ono, K. Mori, M. Suzuki, A. Iwaki, T. Kataoka, M. Fukutoku, T. Nakagawa, T. Sakano, M. Tomizawa, Y. Miyamoto, S. Suzuki, K. Murata, T. Kotanigawa, and A. Maeda, "Hybrid 40-Gb/s and 100-Gb/s PDM-QPSK DWDM transmission using real-time DSP in field testbed," in *Natl. Fiber Opt. Eng. Conf. (NFOEC)*, 2012, p. JW2A.4.

[3] C. R. S. Fludger, J. C. Geyer, T. Duthel, S. Wiese, and C. Schulien, "Real-time prototypes for digital coherent receivers," in *Opt. Fiber Commun. Conf. (OFC)*, 2010, p. OMS1.

[4] P. Larsson-Edefors and E. Börjeson, "Fiber-on-chip: Digital FPGA emulation of channel impairments for real-time evaluation of DSP," in *Opt. Fiber Commun. Conf. (OFC)*, 2022, p. W3H.3.

[5] E. Börjeson, C. Fougstedt, and P. Larsson-Edefors, "Towards FPGA emulation of fiber-optic channels for deep-BER evaluation of DSP implementations," in *Signal Process. in Photonic Commun. (SPPCom)*, July 2019, p. SpTh1E.4.

[6] E. Börjeson and P. Larsson-Edefors, "Cycle-slip rate analysis of blind phase search DSP circuit implementations," in *Opt. Fiber Commun. Conf. (OFC)*, Mar. 2020, p. M4J.3.

[7] H. Kan, H. Zhou, E. Börjeson, M. Karlsson, and P. Larsson-Edefors, "Digital emulation of time-varying PMD for real-time DSP evaluations," in *Asia Communications and Photonics Conf.*, Oct. 2021, p. M4H.4.

[8] K. Cushon, P. Larsson-Edefors, and P. Andrekson, "A high-throughput low-power soft bit-flipping LDPC decoder in 28 nm FD-SOI," in *European Solid-State Circuits Conf.*, 2018, pp. 102–105.

[9] K. Cushon, P. Larsson-Edefors, and P. Andrekson, "Low-power 400-Gbps soft-decision LDPC FEC for optical transport networks," *IEEE J. Lightw. Technol.*, vol. 34, no. 18, pp. 4304–4311, Sept. 2016.

[10] L. Zhang, K. Tao, W. Qian, W. Wang, J. Liang, Y. Cai, and Z. Feng, "Real-time FPGA investigation of interplay between probabilistic shaping and forward error correction," *IEEE J. Lightw. Technol.*, vol. 40, no. 5, pp. 1339–1345, 2022.

[11] *High-Speed Digital Signal Processing Platform*, Fraunhofer Heinrich Hertz Institute, 2022, https://www.hhi.fraunhofer.de/dsp-platform.

[12] A. Dassatti, G. Masera, M. Nicola, A. Concil, and A. Poloni, "High performance channel model hardware emulator for 802.11n," in *IEEE Int. Conf. on Field-Programmable Technology*, 2005, pp. 303–304.

[13] M. Hofer, Z. Xu, D. Vlastaras, B. Schrenk, D. Löschenbrand, F. Tufvesson, and T. Zemen, "Real-time geometry-based wireless channel emulation," *IEEE Trans. Veh. Technol.*, vol. 68, no. 2, pp. 1631–1645, 2019.

[14] J. Zhang, L. Zhang, P. Gao, and F. Shen, "An FPGA based real-time radar target simulator with high spur suppression," in *IEEE Int. Conf. on Signal Processing*, vol. 1, 2020, pp. 126–130.

[15] G. Marsaglia, "Xorshift RNGs," *J. Stat. Softw.*, vol. 8, no. 1, pp. 1–6, 2003.

[16] P. Cappello and K. Steiglitz, "Some complexity issues in digital signal processing," *IEEE Trans. Acoustics, Speech, and Signal Processing*, vol. 32, no. 5, pp. 1037–1041, 1984.

[17] J. P. Gordon and H. Kogelnik, "PMD fundamentals: Polarization mode dispersion in optical fibers," *Proc. National Academy of Sciences of the United States of America*, vol. 97, no. 9, pp. 4541–4550, 2000.

[18] T. Laakso, V. Valimaki, M. Karjalainen, and U. Laine, "Splitting the unit delay," *IEEE Signal Process. Mag.*, vol. 13, no. 1, pp. 30–60, 1996.

[19] G. Liu, "OpenCores: Gaussian noise generator," 2015, https://opencores.org/projects/gng.

[20] P. L'Ecuyer, "Maximally equidistributed combined Tausworthe generators," *Mathematics of Computation*, no. 213, pp. 203–213, 1996.

[21] R. C. C. Cheung, D.-U. Lee, W. Luk, and J. D. Villasenor, "Hardware generation of arbitrary random number distributions from uniform distributions via the inversion method," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 15, no. 8, pp. 952–962, 2007.

[22] *Virtex Ultrascale FPGA VCU110 Development Kit*, Xilinx Inc., 2019, https://www.xilinx.com/products/boards-and-kits/dk-u1-vcu110-g.html.