Waveform Memory for Real-Time FPGA Test of Fiber-Optic Receiver DSPs

Waveform Memory for Real-Time FPGA Test of Fiber-Optic Receiver DSPs

Rafael Romón Sagredo
Dept. Computer Science and Engineering
Chalmers University of Technology
Gothenburg, Sweden
rafaelromon@protonmail.com

Erik Börjeson
Dept. Computer Science and Engineering
Chalmers University of Technology
Gothenburg, Sweden
erikbor@chalmers.se

Ali Mirani
Photonics Laboratory
Chalmers University of Technology
Gothenburg, Sweden
mirani@chalmers.se

Magnus Karlsson
Photonics Laboratory
Chalmers University of Technology
Gothenburg, Sweden
magnus.karlsson@chalmers.se

Per Larsson-Edefors
Dept. Computer Science and Engineering
Chalmers University of Technology
Gothenburg, Sweden
perla@chalmers.se

***Abstract*—Verification of advanced circuit implementations poses many challenges. For complex digital signal processing (DSP) circuits, logic simulations may be prohibitively slow when non-stationary scenarios are considered. A real-time emulation technique like the Fiber-on-Chip (FoC) approach can significantly speed up DSP logic verification. However, a potential weakness with this type of emulation is that it does not use data obtained from experiments, but synthetically creates test data. We introduce a waveform memory, which can be integrated with FoC systems and similar emulators, and which allows measured waveforms to be stored and fed to DSP circuits under test. We perform real-time FPGA experiments where we evaluate a carrier-phase recovery (CPR) module that is tested using either waveform data or synthetic data. Our results for the two different data sets show that the CPR module behaves similarly, both qualitatively and quantitatively, which indicates that the synthetic phase-noise model is a valid replacement of measured data.**

***Index Terms*—DSP, emulation, real-time, FPGA, optical communication**

I. Introduction

Today's communication systems critically rely on digital signal processing (DSP) algorithms to compensate for different channel impairments. As communication systems evolve, more advanced and complex algorithms are employed to improve system performance. However, the increasing algorithm complexity has negative consequences for the following DSP hardware implementation steps. For example, it is challenging to verify advanced DSP circuit implementations under varying channel conditions, since logic simulation run-time becomes prohibitively long.

General-purpose emulator platforms, such as Palladium from Cadence and Veloce Strato from Siemens, facilitate hardware debug and reduce logic verification time. Targeting fiber-optic communication systems, the Fiber-on-Chip (FoC) emulation approach considers not only the receiver DSP to be verified, but it additionally emulates both transmitter and communication channel so that a complete end-to-end communication system is integrated in an FPGA or ASIC [1, 2, 3]. Here, transmitter and channel impairments are synthetically modeled in digital circuits. Since the emulator circuits are under software control, we can run infinitely long emulations during which time-varying channel effects can be programmed, making the test and verification process reproducible.

While the synthetic models in an FoC system have been calibrated to experimental data, they do not represent actual data from experiments. This paper addresses the integration of experimental waveforms in an FoC system to enable side-by-side run-time analysis of synthetic and real channel data.

II. Related Work

Concepts similar to the waveform memory of this paper have been used for verification of other types of systems. The wireless open-access research platform (WARP) proposed in [4] is an FPGA-based radio platform for prototyping radio networks. Capture and playback features allow a designer to move RF transmission data to a computer for further analysis or process these data items using custom DSP algorithms on the FPGA. Zhao et al. introduced a waveform playback system for radar applications [5], which uses two FPGAs connected to a disk array server for storage. Waveform data stored in the server are used by the FPGAs for waveform playback; one FPGA for wide-band transmissions, another for narrow-band transmissions. The waveforms reconstructed by the FPGAs can then be used for real-time DSP tests.

III. Background

Optical fiber communication refers to the method of transmitting information by sending modulated pulses of light through an optical fiber. As with most analog channels, a signal transmitted through an optical fiber is subjected to noise, in this case caused by optical and electronic phenomena. The primary source of noise in fiber-optical systems is amplified spontaneous emission, which is due to spontaneous emissions generated by optical amplifiers and which can be modeled as additive white Gaussian noise (AWGN).

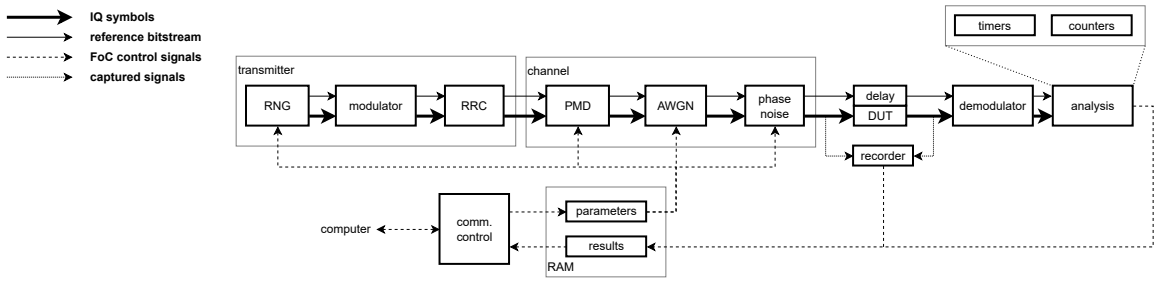

Fig. 1: Block diagram of an FoC system (adapted from [2]).

Since coherent optical communication schemes gained traction around 2005 [6], many transmission impairments have been electronically compensated at the receiver by DSP circuits [7] and even as early as 2008 a real-time prototype of a coherent receiver was published [8]. To test these DSP circuits during development, one (very complex) option is to set up optical-electronic experiments that continuously provide the DSP electronics with data. Another option is to use logic simulations where the DSP-under-test (DUT) is fed with one batch of input test vectors captured in an oscilloscope. A different concept is to emulate the whole system; not only the DUT but also the transmitter and channel. This is the approach taken in the Fiber-on-Chip (FoC) concept [1, 2, 3], where synthetic digital models of channel impairments are used to generate a continuous stream of data to the DUT.

Fig. 1 shows an FoC system with data generation, modulation and pulse shaping (RRC) in the transmitter as well as channel impairments, such as AWGN, polarization-mode dispersion (PMD) and phase noise. Consider now, for instance, that we want to investigate how a carrier phase recovery (CPR) algorithm performs in the presence of the phase noise introduced by phase fluctuations in the carrier and local oscillators. Using the FoC system in Fig. 1, we can then implement the CPR algorithm in VHDL inside the DUT module, activate phase noise emulation (but bypass PMD emulation), and perform a real-time test on an FPGA.

IV. Design and Implementation

This section will describe how we design, implement and integrate a waveform memory in an FoC system. The implementation presented in this paper is partially based on the Chalmers Optical Fiber Channel Emulator (CHOICE), a VHDL implementation of the FoC approach [9].

Fig. 2 shows an example of how the waveform memory can be integrated in an FoC system using the CHOICE environment. The waveform memory is implemented as a new experimental mode of operation, which is based on measurement data and which is implemented in parallel to FoC's synthetic mode of random input data generation and synthetic impairment models. Since we need synchronization interfaces to both the double data rate (DDR) memory and the DSP under test, two different clock regions are required; a memory clock region and a playback clock region. For the latter, we use a configurable clock that is set to the working clock rate of the DSP module, as long as this rate is lower than that of the memory clock region.

In Fig. 2, grayed out entities show the synthetic channel path. A multiplexer-like entity is used to switch between the synthetic and experimental modes of operations. The functionality of the latter mode is encapsulated in the two top-level modules described in the following sections, Sections IV-A and IV-B:

A. Control Module

The *control* module is located in the memory clock region and handles the interfacing to external components and communication with the testbench.

1) DDR Controller: The generic DDR controller performs read and write operations to an external DDR module. In order to improve portability and reduce the complexity of the implementation, the controller relies on an external Xilinx IP block, the memory interface generator (MIG), to interface with the DDR module. The MIG generates a pre-engineered

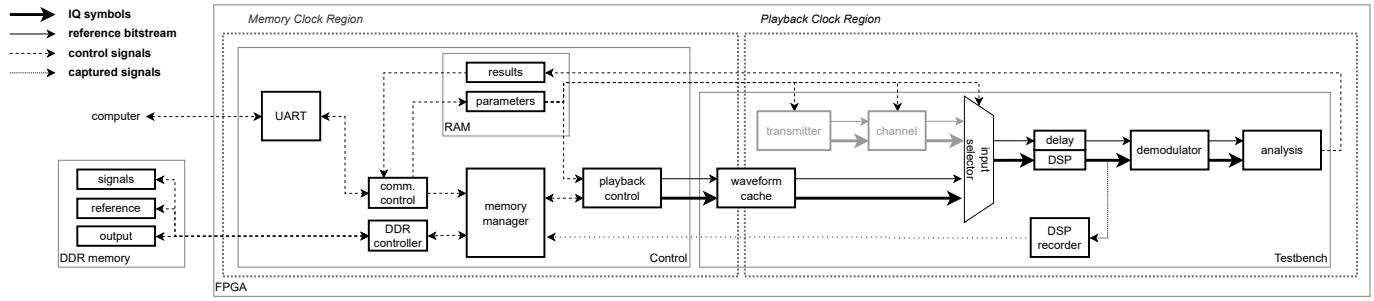

Fig. 2: Block diagram of the waveform memory implemented into an FoC system.

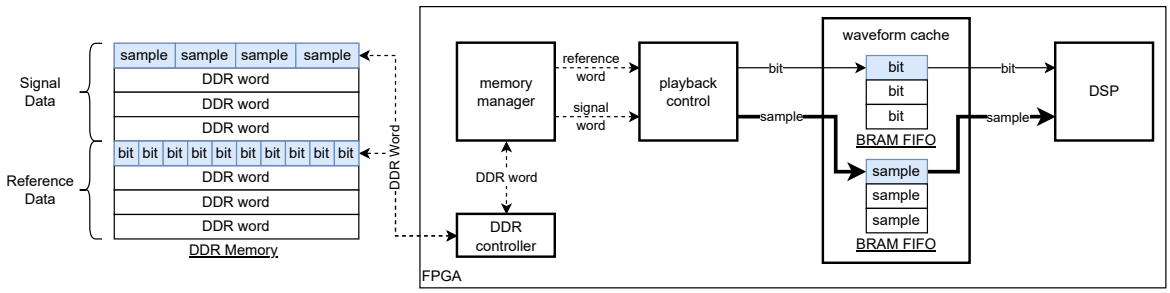

Fig. 3: Block diagram for the memory resources in the waveform memory.

controller and a physical layer (PHY) to interface Xilinx 7-series FPGA user designs with external DDR3 SDRAM devices [10].

2) Memory Manager: This submodule handles all memory operations requested by the other components in the waveform interface, interacting directly with the DDR controller. It keeps track of the different sections of the DDR memory used to store sample (symbol) and result data; essentially turning the DDR memory into a series of circular buffers.

3) Playback Controller: The playback controller slices each 512-bit word stored in the DDR memory into the corresponding number of samples (symbols) of experimental data and loads them into the cache memory used in the testbench entity. It essentially acts as a bridge between the memory manager and the testbench waveform cache, controlling the flow of the DSP tests.

4) Communications Controller: The communications controller allows a designer to control the waveform memory through an UART interface and enables loading of experimental data and retrieval of recorded DSP outputs. It is also used to start and stop the current real-time experiment.

B. Testbench Module

The *testbench* module is located in the playback clock region and houses the DSP under test along with other support modules used during playback.

1) Waveform Cache: This cache memory is used to store the section of experimental data currently being processed by the DSP under test, as two BRAM FIFO buffers sharing read and write enable signals. One buffer stores signal samples and one reference symbol, with one sample/symbol per word. Since this entity interacts with the memory clock region, the write and read interfaces of the FIFO buffer run on different clocks; see Section IV-C for more details on how memory resources are interconnected within the waveform memory.

2) DSP Recorder: The recorder captures a rolling window of the output of the DSP and stores it into the external DDR memory via the memory manager.

3) Analysis: The analysis submodule is made up of a series of test circuits that allow continuous and autonomous evaluation of a DSP implementation, regarding bit errors, etc.

C. Memory Resources

When implementing high-throughput memory-intensive systems, DDR SDRAM is often used as a low-bandwidth bulk data storage, while block RAM (BRAM) on the FPGA is used as higher-bandwidth cache memories throughout the system [11]. In the waveform memory shown in Fig. 3, an external DDR memory module is used as long-term storage for experimental and reference data, with multiple samples/symbols and reference bits stored per row. Small BRAM FIFOs are used to store single samples/symbols with their corresponding bits in each row, acting as a cache memory for the data currently being processed by the DSP under test.

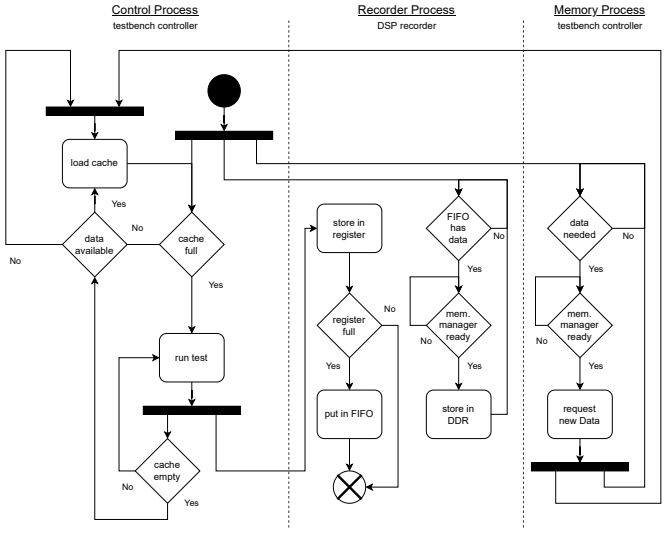

Fig. 4: Flow diagram of the DSP tests.

D. Testing DSP Implementations with the Waveform Memory

In order to test a DSP implementation using the waveform memory in an FoC system, the designer sets the operating mode to *memory* using the UART interface, loads the DDR memory with experimental data and resets the current test run.

After an appropriate amount of time has passed, the designer can download a window of the DSP output and access the performance metrics generated by the *analysis* module through the same UART interface.

Fig. 4 shows a flow diagram that illustrates the functionality and the interaction of the different modules used in the implementation. Three main parallel processes are used:

- The *control process* loads the current section of experimental data into the cache memory and performs

the DSP tests. Since the cache memory cannot provide samples/symbols to the DSP while it is being loaded with new data, the target DSP is stalled by driving its clock enable signals low.

- The *memory process* requests new data to the memory manager when needed by the control process.
- The *recorder process* stores results from the DSP test into DDR memory.

E. Design Decisions and Trade-offs

1) Using DDR memory for bulk storage: Even though its low bandwidth results in the use of supporting cache memories for the system to maintain a high throughput, the DDR's high storage capacity allows for storing the large amounts of data required by this implementation.

2) Using memory interface generator (MIG): Even though relying on an external IP block (MIG) requires the use of Xilinx boards for implementation, the increased complexity of implementing a native DDR controller could result in a longer implementation process and reduce the robustness of the final solution

3) Using a UART interface for communications: UART was chosen mainly due to its wide availability in FPGA boards, however, its low speed represents a bottleneck in the testing process. Future versions of the waveform memory would benefit from a faster communication interface like Peripheral Component Interconnect (PCI) or Ethernet.

V. Use Case: BPS Verification

In this section we present the result of real-time verifications of a blind-phase search (BPS) carrier-phase recovery (CPR) algorithm [12], using both a synthetic channel and the developed waveform memory. For these experiments, an FoC system including the waveform memory was synthesized and uploaded to a Xilinx KC705 evaluation board, which features a Kintex-7 FPGA and a 1-GB DDR3 memory [13]. For a 16-QAM single-polarization system, using 8 bits to represent each I/Q symbol component, 1 GB of memory corresponds to $4 \cdot 10^8$ samples.

A. DSP Under Test

We use a BPS circuit implementation which is described in [14]. In the BPS algorithm, each input symbol is rotated with a number of test phases. The current phase is estimated as the test phase, which results in the minimum average distance between the rotated input symbols and the closest constellation point. Finally, the transmitted symbol is recovered by using the complex conjugate of the recovered phase to rotate the input symbol. Since the number of test phases has a large impact on the BPS circuit's total resource usage, the parabolic interpolation method described in [15] is used. Our implementation uses an 8-bit representation for each symbol component, an averaging window of 64 symbols and two different settings for the number of test phases: 4 and 8, where the former has previously been shown to result in an inadequate phase compensation [14].

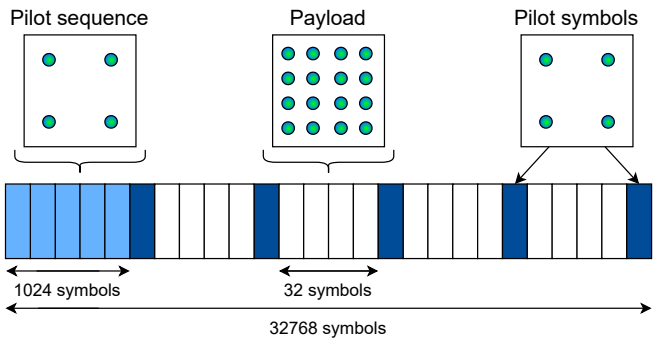

Fig. 5: Waveform frame obtained from the optical testbed.

B. Experimental Data and Synthetic Channel

To provide the implemented waveform memory with data for testing and validation, we use fiber transmission data captured from a 10-GBd back-to-back link in an optical testbed with an optical SNR (OSNR) of 33 dB and a laser source with a wavelength of 1550.12 nm and a linewidth $<$100 kHz. These recorded transmissions consist of a series of frames made up of 4-QAM pilots and a 16-QAM payload. As shown in Fig. 5, each frame consists of 2^{15} symbols, starting with a 1024-symbol 4-QAM pilot sequence, followed by the 16-QAM payload, with a 4-QAM phase pilot introduced every 32 symbols.

Since the captured data contain impairments that are typically compensated for by DSP modules in front of the CPR, all impairments aside from phase noise were compensated for in software using the pilot symbols and the QAMPy Python DSP chain [16]. After removing the pilot symbols, the symbol stream was looped front-to-back to create a continuous waveform with a continuous phase change, as shown in Fig. 6, before being transferred to the FPGA.

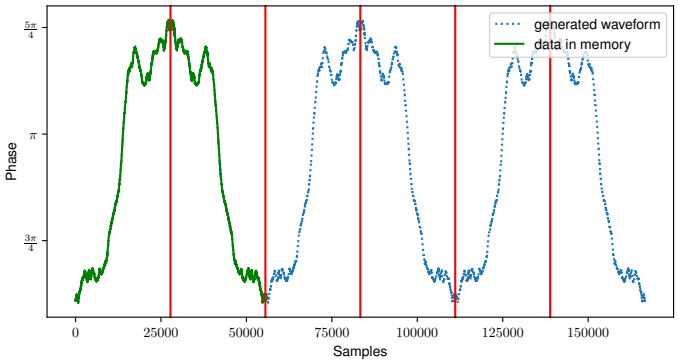

Fig. 6: Phase noise of the data transmitted to the DSP.

The synthetic channel path of the FoC system contains a transmitter with random data generation, and a channel emulator with AWGN [17] and a phase noise generator [1]. These generators were set to mirror the properties of the experimental data from the optical testbed, using the same amount of AWGN and a linewidth symbol-duration product of $100\ \text{kHz}/10\ \text{GBd} = 10^{-5}$.

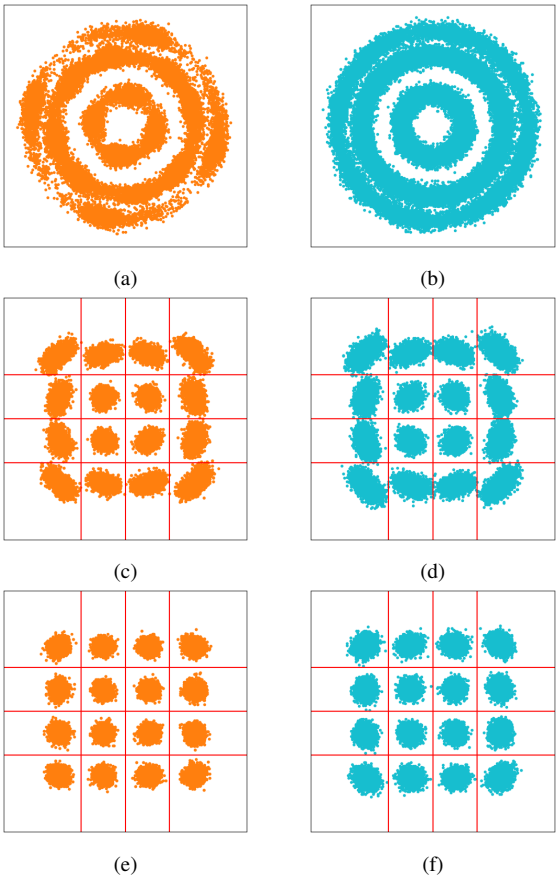

Fig. 7: Constellation plots for the experimental data (orange, left) and the synthetic channel model (cyan, right). (a) and (b) show the input symbols, (c) and (d) the BPS output with four test phases, and (e) and (f) the BPS output using eight test phases. The 16-QAM decision regions are marked with red lines.

C. Test and Analysis

Constellation diagrams of 55,552 BPS input symbols, corresponding to one forward and one backward run of the experimental data shown in Fig. 6, are shown in Fig. 7a and 7b for the experimental and synthetic data, respectively. Since the synthetic channel path is based on pseudo-random generators, the two different data sets are not identical but they exhibit similar properties.

A total of 55,552 symbols were captured by the recorder module at the output of the CPR module (Fig. 2), stored in the waveform memory and subsequently downloaded from the FPGA to allow for plotting of constellations. The output constellations for a BPS using four test phases are shown in Fig. 7c and 7d, for the experimental and synthetic data, respectively. The former results in a bit-error rate (BER) of $0.99 \cdot 10^{-4}$, while the latter results in $1.3 \cdot 10^{-4}$. As expected, four test phases yield a relatively high BER; indeed the BPS requires more test phases to perform adequately for 16-QAM, as predicted in previous work [14].

Increasing the number of test phases to eight and resynthesizing the DSP circuit on the FPGA results in the constellations shown in Fig. 7e and 7f. Here, the BPS is able to recover the received samples back to the original 16-QAM constellation; the BER is less than $1.8 \cdot 10^{-5}$. Most of the points recovered appear centered within the decision regions of the demodulator, shown by red lines in the diagram.

A higher BER is expected for the synthetic data, as the linewidth setting used when generating the synthetic phase noise represents a worst-case scenario for the lasers used in the optical testbed. However, the results from the experimental data and the synthetic channel model are in good agreement, showing that the synthetic model can replace measurement data to enable real-time DSP tests.

VI. Conclusion

In this paper, we introduced a waveform memory design that offers a novel way of testing DSP implementations for communication systems. Our approach allows data captured from communication transmission experiments to be transferred to an FPGA system to drive real-time DSP circuit evaluations. When using the waveform memory to test DSP implementations we found the results to be qualitatively and quantitatively in agreement with tests conducted with synthetic data based on pseudo-random data generation and synthetic, digital channel models. Having access to both an experimental data mode and a synthetic channel mode on a real-time DSP evaluation platform offers many new test capabilities; it becomes possible to 1) perform very deep BER evaluations, 2) use reproducible test scenarios, and 3) stress test DSP modules for worst-case system parameter variations that are difficult to capture in experimental data.

The limited memory of the oscilloscopes typically used to capture the experimental data puts an upper bound on the number of symbols that can be stored. A synthetic channel does not have this limitation, allowing for longer simulations, where system parameters can be easily updated at run-time, enabling analysis of deep-BER properties of the DSP circuits. The maximum length of experimental data, which the waveform memory is able to store, depends on the type of transmission and the target hardware of the implementation. For the configuration presented in this paper, the waveform memory is able to store up to $4 \cdot 10^8$ samples with a throughput of 0.8 Gbit/s which corresponds to 0.004 s in a 10-GBd transmission, enough to calculate the BER of complex DSP implementations.

Even though the waveform memory was presented as a way to test DSP for optical communication systems, there is nothing inherent to optical fiber communications in the waveform memory. The concepts introduced in this paper can be used to implement a generic waveform playback device able to test DSP implementations for multiple types of communication systems.

References

[1] E. Börjeson, C. Fougstedt, and P. Larsson-Edefors, "Towards FPGA emulation of fiber-optic channels for deep-BER evaluation of DSP implementations," *Advanced Photonics Congress, SPPCom*, p. SpTh1E.4, 2019.

[2] 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.

[3] E. Börjeson and P. Larsson-Edefors, "Fiber-on-Chip: Digital emulation of channel impairments for real-time DSP evaluation," *IEEE J. of Lightwave Technology*, early access, Aug. 19, 2022, doi: 10.1109/JLT.2022.3200248.

[4] J. L. Hershberger, E. A. Thompson, and T. S. Loos, "A real-time WARP-based data capture and playback test bed for DSP applications," in *IEEE Digital Signal Processing and Signal Processing Education Meeting (DSP/SPE)*, 2013, pp. 48–53.

[5] Y. Zhao, Y. Su, R. Huang, P. Hu, and Z. Chen, "Design and implementation of a radar waveform playback system for real-time digital signal processing test," in *Sixth Asia-Pacific Conf. on Antennas and Propagation (APCAP)*, 2017, pp. 1–3.

[6] S. Tsukamoto, D.-S. Ly-Gagnon, K. Katoh, and K. Kikuchi, "Coherent demodulation of 40-Gbit/s polarization-multiplexed QPSK signals with 16-GHz spacing after 200-km transmission," in *Opt. Fiber Commun. Conf. (OFC)*, 2005, p. PDP29.

[7] J. Zhao, Y. Liu, and T. Xu, "Advanced DSP for coherent optical fiber communication," *Applied Sciences*, vol. 9, no. 19, 2019.

[8] H. Sun, K.-T. Wu, and K. Roberts, "Real-time measurements of a 40 Gb/s coherent system," *Opt. Express*, vol. 16, no. 2, pp. 873–879, Jan. 2008.

[9] E. Börjeson and P. Larsson-Edefors, "CHOICE - Chalmers Optical Fiber Channel Emulator," 2022. [Online]. Available: https://www.cse.chalmers.se/research/group/vlsi/choice/

[10] *7 Series FPGAs Memory Interface Solutions*, Xilinx. [Online]. Available: https://docs.xilinx.com/v/u/1.7-English/ug586_7Series_MIS

[11] M. Milford and J. McAllister, "Valved dataflow for FPGA memory hierarchy synthesis," in *IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, 2012, pp. 1645–1648.

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

[13] *KC705 Evaluation Board for the Kintex-7 FPGA*, Xilinx. [Online]. Available: https://docs.xilinx.com/v/u/en-US/ug883_K7_KC705_Eval_Kit

[14] E. Börjeson, C. Fougstedt, and P. Larsson-Edefors, "VLSI implementations of carrier phase recovery algorithms for M-QAM fiber-optic systems," *IEEE J. of Lightwave Technology*, vol. 38, no. 14, pp. 3616–3623, 2020.

[15] H. Sun, K. Wu, S. Thomson, and Y. Wu, "Novel 16QAM carrier recovery based on blind phase search," in *Eur. Conf. Opt. Commun. (ECOC)*, 2014, p. Tu.1.3.4.

[16] J. Schröder, M. Mazur, and M. Brehler, "QAMPy a DSP chain for optical communications," 2019. [Online]. Available: https://zenodo.org/record/2638956

[17] G. Liu, "Opencores: Gaussian noise generator," 2015. [Online]. Available: https://opencores.org/projects/gng