Real-Time Implementation of Machine-Learning DSP

Erik Börjeson, Keren Liu, Christian Häger, and Per Larsson-Edefors
Chalmers University of Technology, Gothenburg, Sweden
erikbor@chalmers.se

Abstract: While ML algorithms can learn and adapt to channel characteristics, implementation of ML-based DSP hardware is challenging. We demonstrate a real-time implementation of a model-based ML equalizer that compensates a non-linear and time-varying channel. © 2024 The Author(s)

1. Introduction

Machine learning (ML) has garnered significant interest in recent literature as a potential avenue for advancing digital signal processing (DSP) within optical communication receivers. This trend is driven by the potential of ML to learn and compensate for signal impairments that are not easily addressed by conventional techniques, such as non-linear effects. Indeed, traditional methods often rely on predefined algorithms that may not be well-suited to handle the complex and dynamic nature of optical channels. In contrast, ML algorithms can learn and adapt to the channel characteristics, potentially leading to more robust and efficient signal processing provided that a suitable hardware implementation of these, often complex, algorithms can be developed. Generic function approximators like deep neural networks, while powerful and general purpose, may require significant training data and often lack interpretability due to their black-box nature. Model-based ML techniques on the other hand, take into account domain-specific insights [1]: By infusing models with domain knowledge, such as the physical characteristics of optical channels, these techniques can guide the learning process, potentially reducing the amount of required training data and relaxing computational resources.

Real-time DSP ML implementations can be tested out on field-programmable gate arrays (FPGAs). Compared to the graphics processing units (GPUs) more commonly associated with cloud-based ML, FPGAs have a number of advantages when used for real-time processing. FPGAs are customizable, enabling a designer to tailor the implementation to a specific problem, using custom operations and datatypes, meeting requirements on data throughput and latency. Often an FPGA design is used as a precursor to higher-throughput application-specific integrated circuits (ASICs), commonly used in field-deployed DSP systems.

FPGA implementations of neural network (NN) ML equalizers with static non-linearity compensation have been demonstrated for IM/DD links [2], passive optical networks [3], optical interconnects [4], and coherent systems [5]. Additionally, ASIC design for model-based ML static equalization was studied in [6]. In all of the papers cited above, training is performed offline and only the inference is implemented in hardware. The addition of the ability to conduct on-chip training would allow for real-time adaptation to changing channel conditions. However, this approach is less common in the literature, due to the high hardware complexity associated with implementing online training. In [7], we introduced an FPGA implementation of a model-based multi-layer ML equalizer with pilot-based on-chip training, which enables dynamic compensation of both linear and non-linear impairments. FPGA-based on-chip training has been demonstrated also for other systems, such as IM/DD in [8] and a self-coherent system in [9], which features an adaptive equalizer based on unsupervised K-means clustering.

Here, we introduce some of the concepts related to circuit implementations of ML algorithms, using as example an extended version of our previous equalizer [7]. We demonstrate real-time ML equalization using an FPGA-based Fiber-on-Chip (FoC) [10] test system and illustrate design tradeoffs for resource-efficient implementations.

2. Circuit Design for Machine Learning

When developing DSP circuits, a fixed-point number representation is often preferred over floating-point as it results in a circuit with higher throughput, lower latency and lower resource usage. However, the precision is limited and selecting a suitable wordlength for all signals in a design is a non-trivial task. Due to the bit-growth associated with performing operations such as multiplication, truncation or rounding is extensively used throughout a circuit implementation [10], which can affect the performance of both conventional DSP designs and neural networks [11]. Additionally, complex expressions including e.g. trigonometric functions and exponentials do not always have straightforward circuit implementations, but often need to be implemented using extensive look-up tables (LUTs) or approximated using methods like Taylor expansions, resulting in a performance penalty.

Developing an FPGA design of an ML system comes with its own set of challenges, especially when including on-chip training. Both the inference, or forward propagation (FP), and the training, or backward propagation (BP), paths contain a large number of mathematical operations. Hardware resources in the FPGA fabric such as DSP slices, which are specialized circuits containing efficient multipliers, can quickly become scarce. Since they are a limited resource, we need to use DSP slices in an economical way. Many NN FPGA designs reuse the same

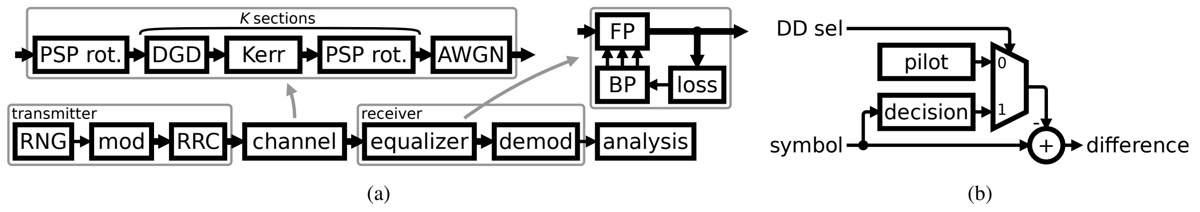


Fig. 1: Block diagrams showing (a) the integration of the ML equalizer into our FoC emulation environment and (b) the logic added to the loss block to enable the equalizer to run in decision-directed (DD) mode.

multipliers for multiple different operations, but since the symbol rates targeted in fiber-optic communication systems are much higher than the FPGA clock rate, this is not possible for the FP path. Here, we need to unroll the NN layers and introduce parallel processing of samples to reach the target throughput. In the BP path, we can process the data in mini-batches, which can facilitate reuse of FPGA resources. To reduce the number of weights used during inference, pruning is commonly done. However, this works best for offline training: When online training is used, pruning becomes difficult as the model parameters are unknown when designing the circuit.

Many FPGAs include multipliers with input wordlengths of 18 bits or more. Using longer wordlengths than 18 can thus incur a large resource overhead. In contrast, when targeting ASICs, each additional data bit can have a significant impact on chip area: The wordlengths should thus be minimized under performance penalty constraints. The DSP slice usage remains constant as long as the total number of data bits fits in one slice. The choice of wordlength will clearly have an impact on other FPGA resources, but for DSP implementations the FPGA's DSP slices are typically the limiting resource.

Including the BP on the FPGA instead of using conventional offline training, e.g. in TensorFlow, creates a feedback loop in the design; this poses an implementation challenge. Firstly, the delay in the FP does not just affect the equalizer's latency, but the combined FP and BP delay affects both the equalizer convergence speed and the speed with which the equalizer can track time-dependent impairments. The delay is caused by the pipelining registers inserted in the design to enable practical clock rates, which in turn affects how large the parallelism needs to be to reach the target throughput. Secondly, with dynamic updates of the NN weights, the DSP system becomes sensitive to small BP path errors, which may accumulate over time, potentially degrading system stability.

3. Model-Based ML Equalizer

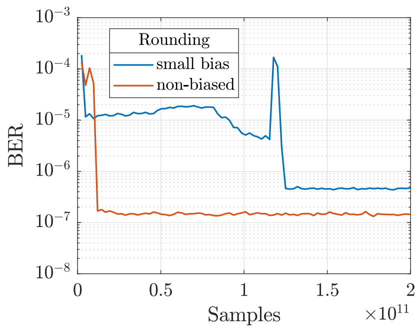
The model-based ML equalizer used in this work is based on the split-step solution of the inverse Manakov polarization mode dispersion (PMD) equation. The FP path consists of three sections, each with one fixed non-linear Kerr-effect compensator and a trainable linear step. The linear steps are implemented as 2×2 multiple-input multiple-output finite impulse response (MIMO-FIR) filters, which use real-value coefficients and are applied independently to the I and Q parts of the two polarizations. We use 2x-oversampling for the input signals, which are downsampled in a matched filter (MF), in the final FP stage. The BP path follows the same structure as the FP path, having linear and non-linear layers. To reduce the resource usage, the BP of the MF and linear layers use the extra clock cycles provided by batch-processing to reuse hardware. Basic reusable operators have been extracted from the mathematical descriptions of the BP to enable resource sharing when simultaneous processing is unnecessary. Compared to our previous paper [7], where more details of the implementation can be found, the equalizer has been updated with a decision-directed (DD) mode and an alternate version of the BP Kerr module, which uses LUTs instead of first-order Taylor expansions.

4. Evaluation Method

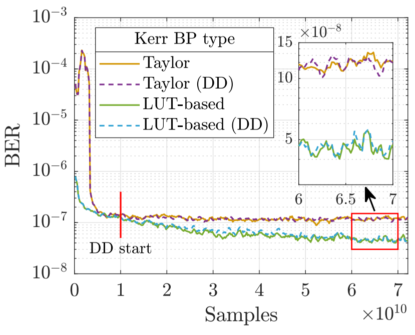
Designs containing feedback paths can suffer from stability issues, which can be challenging to uncover due to the long time it takes to run logic simulations of complex circuit implementations. To speed up analysis, we use our FoC system model, which emulates a fiber-optic communication system on an FPGA [10]. The FoC system uses digital models of the transmitter and channel and on-board analysis tools, to reduce the evaluation times of DSP modules by orders of magnitude over simulation. The FoC system block diagram in Fig. 1a includes the Kerr-effect emulator described in [12]. This system enables us to change, during emulation, the amount of additive white Gaussian noise (AWGN), principal state of polarization (PSP) rotations, differential group delay (DGD) and Kerr effect. We can also control equalizer parameters, such as activation of DD mode.

5. Results

In the BP path, a non-biased round-to-even approach is used in all places where the signal wordlengths are reduced to prevent bit-growth. To evaluate how a small rounding bias in the BP path affects the convergence and stability of the equalizer, we replace one of these rounding operations in the loss function with round-half-up. This type of rounding is very common in digital circuit design due to its simplicity, but it introduces a small positive bias to the result by rounding all half-way values up. In the FoC evaluation, we emulate a 32-Gbaud dual-polarization QPSK



(a)



(b)

Fig. 2: Results from FoC emulation runs showing (a) how the BER stability is affected by a small rounding bias in the loss function, and (b) the difference between using a Taylor expansion and a LUT in the Kerr BP calculation, including the use of a DD mode.

	Slice LUTs	DSP slices	Block RAM
FoC	33,250	274	104
Equalizer	190,996	1,662	0
-FP	22,804	342	0
-BP	168,365	1,320	0
--loss	4,543	0	0
--MF	13,247	104	0
--linear	139,856	768	0
--Kerr	19,615	736	0
Total	52%	54%	7%

Table 1: FPGA resources used on a Xilinx VC709 development board.

system, transmitting over a 300-km fiber using a 3-section channel model, shown in Fig. 1a, with randomized PSP rotations and a PMD parameter of $D = 0.2$ ps/$\sqrt{\text{km}}$. The launch power is set to 10 dBm and the Kerr non-linearity to $\gamma = 1.2$ rad/W/km. Finally, we add AWGN to reach a signal-to-noise ratio (SNR) of 16 dB. We run the FoC system for 2^{11} samples, capture the number of processed bits and bit errors every minute, and use those to calculate the bit-error rate (BER) for the last minute of runtime, which corresponds to a window of approximately $2.5 \cdot 10^9$ samples. The results, shown in Fig. 2a, demonstrate that a small bias in the BP path can have a large effect on the convergence time, stability and the BER after convergence. It also illustrates the need for long simulations to uncover stability issues. Such simulations would be hard to accomplish without a system like FoC, which cuts the runtime by approximately 5 orders of magnitude compared to software-based logic simulations.

To compare the two different implementations of the Kerr BP module, we perform FoC runs of both solutions using the same channel parameters as above with identical randomized data and channel properties. To be able to view fast changes we use a smaller capture window of 5 seconds. The results are shown in Fig. 2b, where the LUT-based solution has a faster convergence speed and a lower final BER. A demonstration of the DD mode, which is started after 10^{10} samples to be well away from the convergence phase, is also included in the graph. The DD mode tracks the pilot-based version well, which is expected for this relatively high SNR.

The total utilization of a complete FPGA system targeting a Xilinx VC709 development board is shown in Table 1, including an equalizer using the Taylor expansion in the Kerr BP module and non-biased rounding. The majority of the resources are used by the BP portion of the equalizer, while the FoC system is relatively small. Using non-biased rounding instead of the simpler round-half-up does not have any significant impact on the resource usage. However, using a LUT-based solution instead of a Taylor expansion in the Kerr BP module increases the total DSP usage of the equalizer with 13%. An additional issue is that the propagation delay of the slower LUT-based Kerr BP module becomes a limiter to the maximum clock rate of the system. This can be solved by introducing an extra pipelining stage at the cost of a higher latency in the feedback loop.

6. Conclusion

We have discussed some of the issues related to designing a machine-learning system targeting field-programmable gate arrays (FPGAs) or application-specific integrated circuits (ASICs). We focus on a model-based approach, where we include domain-specific insights to design a system, as opposed to a more generic deep neural network. Using a non-linear multiple-input multiple-output (MIMO) equalizer as an example and an FPGA-based emulation environment, we show how minor rounding biases and approximations, such as Taylor expansions, can have significant effects on the equalizer output, illustrating the need for long simulations runs and highlighting the challenge of finding good trade-offs between circuit complexity and DSP performance.

Acknowledgements: The work of C. Häger was supported by the Swedish Research Council under grant no. 2020-04718.

References

1. C. Häger *et al.*, "Physics-based deep learning for fiber-optic communication systems," IEEE J. Sel. Areas Commun. **39**, 280–294 (2021).
2. M. Chagnon *et al.*, "Quantized deep neural network empowering an IM-DD link running in realtime on a field programmable ..." in *Proc. ECOC,* (2019).
3. N. Kaneda *et al.*, "FPGA implementation of deep neural network based equalizers for high-speed PON," in *Proc. OFC,* (2020).
4. M. Li *et al.*, "FPGA implementation of time-interleaved pruning neural network equalizer for short reach optical interconnects," in *Proc ACP,* (2021).
5. P. J. Freire *et al.*, "Implementing neural network-based equalizers in a coherent optical transmission system using field-programmable gate arrays," JLT (2023).
6. C. Fougstedt *et al.*, "ASIC implementation of time-domain digital backpropagation with deep-learned chromatic dispersion filters," in *Proc. ECOC,* (2018).
7. K. Liu *et al.*, "FPGA implementation of multi-layer machine learning equalizer with on-chip training," in *OFC,* (2023), p. M1F.4.
8. J. Ney *et al.*, "Unsupervised ANN-based equalizer and its trainable FPGA implementation," in *Proc. EuCNC/6G Summit,* (2023).
9. E. Giacoumidis *et al.*, "Real-time machine learning based fiber-induced nonlinearity compensation in energy-efficient ..." APL Photonics **5**, 2–7 (2020).
10. E. Börjeson *et al.*, "Fiber-on-Chip: Digital emulation of channel impairments for real-time DSP evaluation," IEEE JLT **41**, 888–896 (2023).
11. S. Gupta *et al.*, "Deep learning with limited numerical precision," in *Proc. ICML,* (2015), p. 1737–1746.
12. K. Liu *et al.*, "FPGA-based optical Kerr effect emulator," in *SPPCom,* (2022), p. SpTh1I.2.