International Journal of Parallel Programming (2026) 54:19
https://doi.org/10.1007/s10766-026-00825-7

CNN-Based Equalization for Communications: Achieving Gigabit Throughput with a Flexible FPGA Hardware Architecture

Jonas Ney[1] · Christoph Füllner[2] · Vincent Lauinger[3] · Laurent Schmalen[3] · Sebastian Randel[2] · Norbert Wehn[1]

Received: 19 April 2024 / Accepted: 4 April 2026

Abstract
To satisfy the growing throughput demand of data-intensive applications, the performance of optical communication systems increased dramatically in recent years. With higher throughput, more advanced equalizers are crucial, to compensate for impairments caused by inter-symbol interference (ISI). The latest research shows that artificial neural network (ANN)-based equalizers are promising candidates to replace traditional algorithms for high-throughput communications. On the other hand, not only throughput but also flexibility is a main objective of beyond-5 G and 6 G communication systems. A platform that is able to satisfy the strict throughput and flexibility requirements of modern communication systems are field programmable gate arrays (FPGAs). Thus, in this work, we present a high-performance FPGA implementation of an ANN-based equalizer, which meets the throughput requirements of modern optical communication systems. Further, our architecture is highly flexible since it includes a variable degree of paralellism (DOP) and therefore can also be applied to low-cost or low-power applications which is demonstrated for a magnetic recording channel. The implementation is based on a cross-layer design approach featuring optimizations from the algorithm down to the hardware architecture, including a detailed quantization analysis. Moreover, we present a framework to reduce the latency of the ANN-based equalizer under given throughput constraints. As a result, the bit error rate (BER) of our equalizer for the optical fiber channel is around four times lower than that of a conventional one, while the corresponding FPGA implementation achieves a throughput of more than 40GBd, outperforming a high-performance graphics processing unit (GPU) by three orders of magnitude for a similar batch size.

Keywords FPGA · Machine learning · Neural networks · Optical communications

Extended author information available on the last page of the article

Published online: 29 June 2026

1 Introduction

In recent years, the achievable throughput of optical communication systems grew dramatically, driven by the increasing demand for high-speed data transmission in various applications such as data centers, video streaming, and cloud computing [1]. The higher throughput leads to lower signal-to-noise ratio (SNR) and an increased inter-symbol interference (ISI) which strongly impairs the system's performance. As a result, the design and implementation of advanced signal processing techniques has become crucial for maintaining high communication performance and low bit error rate (BER). On the other hand, in addition to high throughput, future communication standards strongly focus on flexible architectures [2], [3] to satisfy application requirements for various use cases.

To prevent an increase in BER while providing high flexibility, latest research put a strong emphasis on artificial neural network (ANN)-based algorithms for communication systems [4], [5], [6]. Especially the equalizer, responsible for compensating channel impairments of the received signal, is a component that can benefit from the advancements of ANN research. In particular, ANNs have shown remarkable results for channels with non-linear effects, for which no exact analytical solutions exist for equalization [7]. In this work, we study a 40GBd optical intensity modulation with direct detection (IM/DD) channel, where non-linear distortions are caused by chromatic dispersion (CD) [8], as an exemplary use case for a high-throughput ANN-based equalization. Further, we provide results for a low-cost telephone channel to highlight the flexibility of our hardware architecture.

Compared to conventional algorithms, which have been optimized for decades, ANNs introduce high computational complexity, limiting the achievable throughput on general-purpose processors like central processing units (CPUs) or graphics processing units (GPUs). In contrast, application-specific platforms like field programmable gate arrays (FPGAs) or application-specific integrated circuits (ASICs) provide huge parallelism and customizability. As compared to ASICs, the deployment of FPGAs is more cost-efficient for low-volume productions and they can provide a shorter time-to-market [9, Ch. 1]. Further, FPGAs can be reprogrammed to provide the required flexibility.

Most previous works utilized FPGAs as a platform for performance evaluation and prototyping of ASIC implementations [10], [11], [12], [13], [14]. In contrast, in this work, we aim to satisfy the strict throughput requirements on the FPGA itself. This way the design can either be deployed on the FPGA or be used as a well-grounded base for an ASIC design. However, when the design is deployed on an FPGA, it has to satisfy the strict performance requirements on this resource-constrained device, which is a great challenge even with advanced FPGA platforms. In [10], the FPGA implementation of a recurrent neural network (RNN)-based equalizer for a 34GBd single-channel dual-polarization as well as approximations of non-linear activation functions are presented. Their throughput requirements can be satisfied by using 5 FPGAs and a high number of parallel outputs (61) of each RNN, which is not feasible for the 40GBd IM/DD channel we focus on in this work, as it results in significantly increased BER. In [11] and [12], a channel more similar to ours is considered. However, the implementation of [11] only achieves a throughput of 2.6Gbps for a

channel with a data-rate of 50Gbps. In [12] a pruned ANN is utilized for equalization of a 50GBd pulse-amplitude modulation (PAM)-4 channel. To achieve the required throughput on FPGA, the BER requirement needs to be relaxed from $1 \cdot 10^{-5}$ and $3.8 \cdot 10^{-3}$. In [13], an RNN is compared to a fully-connected ANN for equalization of an IM/DD channel. While the RNN achieves a much lower BER, only the ANN is able to meet the throughput requirements of the optical communication channel. In [14] a novel unsupervised loss function for ANN-based equalization is proposed. Further, a trainable FPGA implementation of the approach is presented which enables daptation to varying channel conditions during runtime. However, the implementation does not achieve the required channel throughput of 25GBd.

In this work, we present a high-throughput FPGA implementation of a convolutional neural network (CNN)-based equalizer for an optical IMDD channel. In contrast to previous works, we apply a cross-layer design methodology, which involves an extensive design space exploration of the CNN topology, and a framework for selecting the appropriate sequence length per CNN instance. Moreover, a detailed quantization analysis based on an automatic quantization approach is conducted. Further, we show how our design approach can also be applied to low-cost channels with lower throughput constraints. For the high-throughput, 40GBd channel, we focus on high parallelism across all design layers, from algorithm down to implementation. Further, special attention is given to low latency which is crucial for optical communication used in high-frequency trading or telemedicine.

As a result, our FPGA implementation achieves a BER around one order of magnitude lower than that of a conventional equalizer, while satisfying the throughput requirement of 40GBd. Further, our approach outperforms an implementation on a high-performance GPU by four orders of magnitude for a similar batch size. This article is an extension of the work presented in [15]. It significantly extends the previous publication by introducing finite impulse response (FIR) filters and Volterra kernels to the design space exploration to provide a fairer comparison, by conducting a comprehensive quantization analysis, by extending the hardware architecture with a flexible degree of paralellism (DOP) to enable the adaptation to different application scenarios, by applying and evaluating the approach for a magnetic recording channel, by analyzing the influence of the DOP on the throughput and the power consumption, and by extending the comparisons to TensorRT implementations, embedded GPU implementations and providing an analysis of the dynamic power consumption.

In summary, our novel contributions are:

- A detailed design space exploration of the CNN, featuring cross-layer analysis and automatic quantization, resulting in a network with a BER one order of magnitude lower than that of a conventional equalizer;
- An efficient hardware architecture, suited for high-throughput as well as low-cost application scenarios by providing flexibile DOPs on multiple implementation levels
- An in-depth trade-of analysis of our automatic quantization approach
- A framework allowing to trade-off throughput against latency to adapt for application requirements, based on a timing model of our architecture;
- An advanced implementation of a high-performance CNN-based equalizer for

optical communication, achieving a throughput of more than 40GBd

2 Investigated Communication Channels

For our CNN-based equalization approach we mainly focus on the high-throughput IM/DD channel presented in Sect. 2.1. To further highlight the flexibility of our approach we also give results for the band-limited magnetic recording channel as described in Sect. 2.2. The first channel is based on an experimental setup where the input and output data is captured while the second channel is simulated in a Python environment.

2.1 Fiber-Optical Channel

In fiber-optic communications, the application of ANN-based equalizers has been proven beneficial for mitigation of nonlinear distortions for which no analytic expression exists [16, 17]. In this work, as an example of a high-throughput channel, we chose an IM/DD transmission system, where non-linear impairments are caused by the interplay between CD and direct detection to evaluate the performance of the ANN-based equalizer. Specifically, we modulate the intensity of 1550nm with a high-speed zero-chirp mach-zehnder modulator (MZM) that is biased at the quadrature point. Following the recommendations of [18], we use a pseudo random sequence based on the Mersenne-Twister algorithm as a transmit pattern and drive the MZM with a 40GBd pulse amplitude modulation signal with two levels (PAM2) and a root-raised-cosine spectral shape. The resulting optical on-off-keying signal is launched into a standard single-mode fiber with a length of 31.5km that features a CD coefficient of approximately 16 ps nm $^{-1}$ km $^{-1}$. At the receiver side, we employ a 40GHz photodetector to detect the envelope of the optical signal. Since CD is an effect related to the optical field, it impairs the photocurrent obtained after square-law detection in a nonlinear way. Finally, the electrical signal is recorded by a real-time oscilloscope. We digitally resample the captured waveforms and apply a timing recovery algorithm to align the received waveform with the transmit pattern for the training of the ANN. We digitally precompensate the frequency-dependent attenuation of the transmitter components so that transceiver noise and CD remain as the effects impairing the quality of the received signal.

2.2 Magnetic Recording Channel

As a second channel, with a smaller bandwidth and therefore a lower maximal throughput, we simulate a linear bad-quality communication channel as described in [19, Ch. 9.4-3]. The channel is known as *Proakis-B* and has the following discrete impulse response:

$$\mathrm{h}_{\mathrm{ch,\,ProB}} = [0.407,\ 0.815,\ 0.407]\ .$$

In the simulation, the transmitted symbols x, are convolved with a raised-cosine (RC) pulse shaping filter and the linear channel impulse response of the channel $h_{\mathrm{ch,\,ProB}}$. Afterwards, the received vector is superimposed by Gaussian noise. Similar to the experimental setup, we run our simulation with an oversampling rate of $N_{\mathrm{os}} = 2$.

3 CNN Design Space Exploration

The following design space exploration is performed for the optical fiber channel described in Sect. 2.1. At the end of the section, we show that the obtained model can also be successfully applied to the simulated magnetic recording channel.

An optimized neural network topology is crucial for hardware implementation of ANN-based algorithms, as it has a huge influence on the power consumption, throughput, and latency of the final implementation. However, the design of efficient ANN topologies is characterized by an enormous design space with various hyperparameters. This design space includes the layer type, the number of layers, the size of each layer, the activation function, and multiple more hyperparameters. An exploration of all those parameters is nearly infeasible, thus we restrict our analysis to the topology template presented in the following, which provides sufficient configurability while comprising a manageable design space.

3.1 CNN Topology Template

The core of our equalizer is a one-dimensional CNN since it resembles the structure of traditional convolutional filters. The CNN is based on a customizable topology template shown in Fig. 1, where specific parameters are determined in an extensive design space exploration.

The CNN is composed of L convolutional layers with identical kernel size K and padding P. Each convolutional layer but the last is followed by batch normalization and rectified linear unit (ReLU) activation functions. One channel is used for the input sequence, while subsequent activations consist of C channels. The output of the CNN is based on V_{p} channels, thus V_{p} values are calculated in parallel for one pass of the network. To shift the input sequence accordingly, the first layer has a stride of V_{p}, while the following layers have a stride of one, and the last stride is set corresponding to the oversampling factor N_{os}. After the last convolutional layer, the feature map is

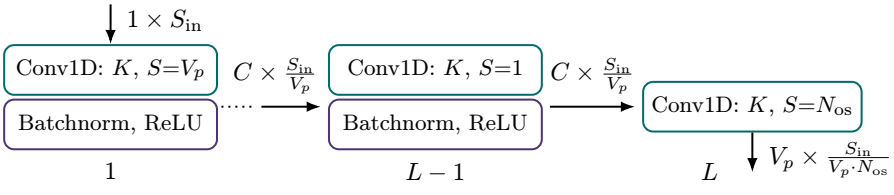


Fig. 1 Topology template of the equalizer CNN. The feature map dimensions are given next to the arrows, where the first dimension corresponds to the number of channels and the second one to the width

flattened so that each element of the feature map corresponds to one output symbol. Afterwards, the output is mapped to the closest constellation symbol.

3.2 Linear Equalizer

To compare the performance of our CNN-based equalizer, we also include a conventional linear feedforward equalizer in our design space exploration. The equalizer is based on a FIR filter with M taps, which is convolved with the input sequence to calculate the output sequence. For a time-discrete system, the output sequence is the weighted sum of M inputs:

$$y_i = \sum_{m=-M^*}^{M^*} x_{i+m} \cdot w(m + M^*) \ , \tag{1}$$

where x is the input sequence, y is the output sequence, $w(m)$ is the weight of the $m\mathrm{th}$ tap, $M^* = \lfloor * \rfloor M \,/\, 2$ and $\lfloor * \rfloor \cdot$ declares the floor operator. For an oversampling factor of $N_{\mathrm{os}} = 2$, every second output sample corresponds to an output symbol. Similar to the CNN, after equalization, the output is mapped to the closest constellation symbol. We train the linear equalizer in a supervised manner using the mean-squared-error (MSE) loss and the Adam optimizer.

3.3 Volterra Equalizer

For further comparison, we also include a more complex equalizer which is based on the Volterra kernel. This nonlinear equalizer calculates the sum of multidimensional convolutions with Volterra kernels up to an order of P. Since high-order Volterra kernels are associated with high complexity, we restrict our exploration up to kernels of order 3. A Volterrra equalizer of order 3 can be described by the following equation:

$$\begin{aligned} y_i = w_0 &+ \sum_{m_1=-M_1^*}^{M_1^*} x_{i+m_1} \cdot w_1(m_1 + M_1^*) + \\ &\sum_{m_1=-M_2^*}^{M_2^*} \sum_{m_2=-M_2^*}^{M_2^*} x_{i+m_1} \cdot x_{i+m_2} \cdot w_2(m_1 + M_2^*, m_2 + M_2^*) + \\ &\sum_{m_1=-M_3^*}^{M_3^*} \sum_{m_2=-M_3^*}^{M_3^*} \sum_{m_3=-M_3^*}^{M_3^*} x_{i+m_1} \cdot x_{i+m_2} \cdot x_{i+m_3} \cdot w_3(m_1 + M_3^*, m_2 + M_3^*, m_3 + M_3^*) \ , \end{aligned}$$

where x is the input sequence, y is the output sequence, M_p is the memory length of the $p\mathrm{th}$ order kernel, $M_p^* = \lfloor * \rfloor M_p \,/\, 2$, $\lfloor * \rfloor \cdot$ declares the floor operator, w_1 are the first-order weights, w_2 are the second-order weights and w_3 are the third-order weights. Similar to the CNN-based and the linear equalizer, the Volterra equalizer is trained in a supervised fashion with MSE loss and Adam optimizer.

3.4 Design Space Exploration Framework

To explore this design space of various CNN configurations, we design a framework that allows us to automatically evaluate multiple configurations which are compared in terms of communication performance and complexity. Further, the framework features cross-layer analysis by providing an estimate of the achievable throughput. Thus hardware metrics are already included in the topology search, which greatly reduces the development cycles since multiple models can already be discarded in an early design phase.

As configurable hyperparameters of the CNN, we select the number of layers L, the kernel size K, the number of channels C, and the symbols calculated in parallel V_p. We train each configuration three times for 10000 iterations with an initial learning rate of 0.001 with the Adam optimizer and the MSE loss. After training, the highest achieved BER of the three training runs and the corresponding multiply-accumulate (MAC) operations per input symbol of each configuration are determined by our framework. This way, a trade-off between communication performance and hardware complexity can be found.

For the linear equalizer and the Volterra equalizer, we explore the design space by varying the number of taps. While this corresponds to a one-dimensional exploration for the linear equalizer, the Volterra equalizer contains taps in three dimensions. Similar to the CNN, we evaluate the complexity of both equalizers based on the number of MAC operations to calculate one output symbol.

3.5 Results of Design Space Exploration

In Fig. 2, the results of the design space exploration are shown. Our design space for the CNN is spanned by the following four dimensions: symbols calculated in parallel $V_p \in \{1, 2, 4, 8, 16\}$, network depth $L \in \{3, 4, 5\}$, kernel size $K \in \{9, 15, 21\}$ and the number of channels $C \in \{3, 4, 5\}$. Thus, overall 135 different models are trained and evaluated. The average MAC operations per symbol $\mathrm{MAC}_{\mathrm{sym}}$ can be calculated as follows:

$$\mathrm{MAC}_{\mathrm{sym}} = \frac{K \cdot C}{V_p} + (L-2) \cdot \frac{K \cdot C \cdot C}{V_p} + \frac{K \cdot C}{N_{\mathrm{os}}} \,.$$

For the linear equalizer, the design space is spanned by the number of taps $M \in \{3, 5, 9, 17, 25, 41, 57, 89, 121, 185, 249, 377, 505, 761, 1017\}$ and for the Volterra equalizer by the number of taps of each order $M_1 \in \{3, 9, 15, 25, 35, 55, 75, 89, 121\}$, $M_2 \in \{1, 3, 9, 15, 25, 30, 35\}$ and $M_3 \in \{1, 3, 9, 15\}$.

The Pareto optimal models of each approach correspond to the most promising candidates for implementation since they provide the best trade-off between complexity and communication performance.

Further, the framework approximates the maximal $\mathrm{MAC}_{\mathrm{sym}}$ to achieve the required throughput T_{req} of 40GBd based on the clock frequency f_{clk}, and the available DSPs of our target device as follows:

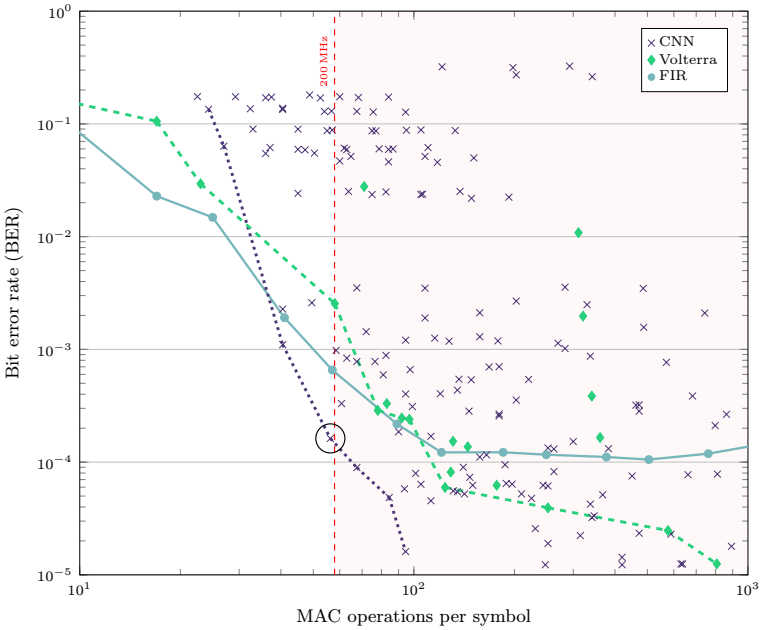


Fig. 2 Results of design space exploration of the different equalization approaches. The maximal $\mathrm{MAC}_{\mathrm{sym}}$ to achieve the throughput of 40GBd with a clock frequency of 200MHz is given by the vertical red line. The Pareto optimal models of the CNN-based equalizer, the Volterra kernel, and the FIR filter are connected by the dotted, solid, and dashed lines respectively

$$\mathrm{MAC}_{\mathrm{sym,\ max}} = \frac{\mathrm{DSP}_{\mathrm{avail}}}{T_{\mathrm{req}}} \cdot f_{\mathrm{clk}} \cdot 1.2\ .$$

A factor of 1.2 is introduced since some arithmetic operations are implemented using look-up table (LUT) resources instead of DSPs which increases the total number of feasible MAC operations. Note that the equation is an approximation, mostly based on previous investigations and experiments, specifically designed for our hardware architecture. The equation might not be universally applicable to other hardware architectures or use cases.

Previous experiments showed that a clock frequency above 200MHz often results in timing violations. Thus, in our design-space exploration, we set the limit for $\mathrm{MAC}_{\mathrm{sym}}$ to the value which corresponds to an approximated throughput of 40GBd with a clock frequency of 200MHz.

As a result, in Fig. 2 we can see that for a fixed complexity in terms of MAC operations per symbol, the Pareto optimal CNN models are outperforming the linear equalizer starting from a BER of around $10^{-}2$ with respect to communication performance. Only when constrained to really low complexities of around 20 MAC operations per symbol, the linear equalizer provides a lower BER. It can also be seen that the linear equalizer's performance saturates at a BER of around 10^{-4}. This is probably the result of non-linear distortions of the optical fiber and CD since those effects can not be fully compensated by a linear equalizer. In contrast to the FIR filter, the

Volterra kernel introduces non-linearity in the equalization process. This way, similar to the CNN, it is able to compensate for non-linear distortions. Thus, with sufficient complexity, the Volterra kernel provides a lower BER than the FIR filter. However, for similar complexity, the Volterra kernel is outperformed by multiple CNN configurations in terms of communication performance.

For our hardware implementation, we select the configuration with the lowest BER while satisfying our throughput requirements of 40GBd with a clock frequency of 200MHz. This configuration is highlighted by the black circle and corresponds to a model with $V_p = 8$, $L = 3$, $K = 9$, and $C = 5$, as visualized in Fig. 3. In Fig. 2, we can see that the BER achieved by a linear equalizer with the same complexity as the CNN is around four times higher, while the Volterra's BER is more than one order of magnitude higher. This shows that our CNN topology is well suited for the equalization of the optical fiber channel.

3.6 Performance for the Magnetic Recording Channel

To show the applicability of the CNN-based equalizer to different application scenarios, we train the CNN selected in the design space exploration for the magnetic recording channel presented in Sec. 2.2. We model the SNR of the bad-quality channel with 20dB. Similar to the high-throughput channel, we compare FIR-filter-based equalizers and Volterra-based equalizers to our CNN approach in terms of complexity and communication performance.

The results are shown in Fig. 4. The CNN achieves a BER of 8.4 e-3 while the linear FIR filter's BER with similar complexity is 9.6e−3. Thus the gap between the two equalization approaches is much smaller as compared the optical fiber channel. This is reasonable since the main advantage of the CNN lies in the compensation of non-linear distortions which are not present in the simulated magnetic recording channel. Thus, for the linear distortions of this channel, the FIR filter achieves comparable performance. However, the CNN still slightly outperforms the conventional

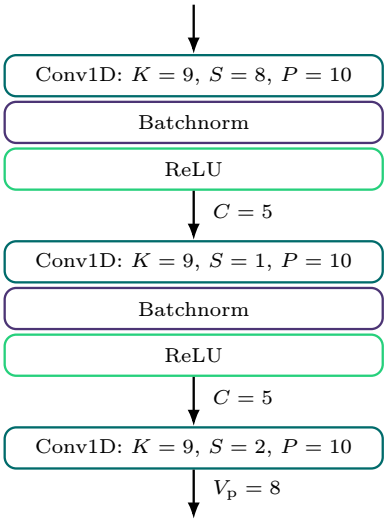


Fig. 3 Final topology of the CNN-based equalizer with three layers, where K corresponds to the kernel size, S to the stride, P to the padding, and V_p to the symbols calculated in parallel

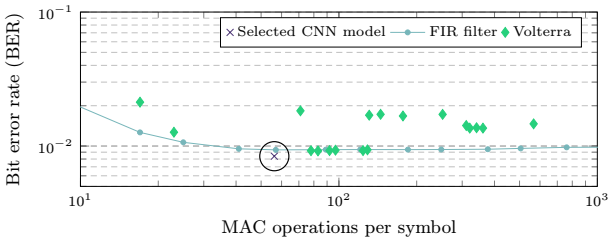


Fig. 4 Complexity and communication performance of the selected model as compared to conventional FIR filters and Volterra kernels for the magnetic recording channel

FIR filter with similar complexity in terms of communication performance. The Volterra equalizer provides similar performance as the FIR filter, with the drawback of slightly higher complexity. The results show that even though the CNN topology is selected based on the high-throughput channel, it still achieves sufficient performance for other channels. This further demonstrates the high flexibility of CNN-based equalizers.

4 Quantization

In addition to the network topology, another essential aspect of an efficient ANN implementation on resource-constrained devices is the quantization of weights and activations. In contrast to the 32-bit floating-point format used in software, each value on the FPGA is represented in fixed-point format with arbitrary decimal and fractional width on the FPGA.

To explore the quantization efficiently, we include an automated quantization approach in our framework, similar to the one proposed in [20]. Therefore, the loss function is modified to simultaneously learn the precision of each layer while optimizing the accuracy of the ANN during training. This is achieved by using a differentiable interpolation of the bit-widths, which allows to train them using back-propagation. Similar to [20], we include a quantization trade-off factor (QLF) in the loss function, which determines how aggressively to quantize. This enables efficient exploration of the trade-off between bit width and communication performance.

The quantization-aware loss can be described by the following equation:

$$\text{loss} = \frac{1}{S_{\text{in}}} \sum_{i=1}^{S_{\text{in}}} (y_i - x_i)^2 + \text{QLF} \cdot \frac{\text{B}_\text{p} + \text{B}_\text{a}}{2} \ ,$$

where S_{in} is the input sequence length, $\text{y} = (y_1, y_2, \ldots)$ are the predicted symbols, $\text{x} = (x_1, x_2, \ldots)$ are the transmitted symbols, B_p is the average number of bits of the trainable parameters and B_a is the average number of bits of the activations.

In contrast to [20], where an integer representation together with a scaling factor that is coupled to the bit-width of the values is learned, we adjust the algorithm to separately learn the integer and fraction width. This way, there is no need to scale the

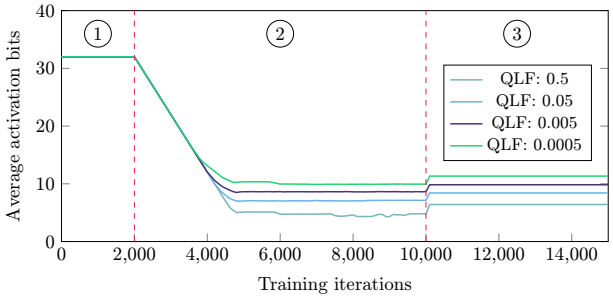


Fig. 5 Course of the average activation bit width during the three phases of quantized training for different QLFs

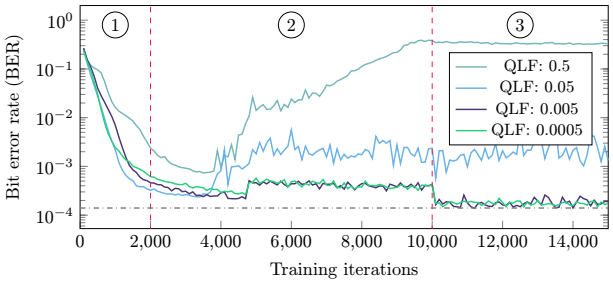


Fig. 6 Course of the BER during the three phases of quantized training for different QLFs. The BER of the full precision model is given by the gray horizontal line

values in hardware during computation, as the numbers are directly represented by their corresponding fixed-point value and can directly be mapped to our hardware architecture. With the help of our framework, we perform a quantization analysis for the most promising CNN model found in our design space exploration.

Our training process for quantization can be divided into three steps:

1. Full precision training: Perform training in full precision to find a well-initialized model for quantization,
2. Bit-width-aware training: Train bit width of weights and activations and optimize communication performance simultaneously,
3. Fine-tuning: Fix bit width and perform further training iterations to improve communication performance.

In Fig. 5, the training of the bit widths is shown for different QLFs. In the first phase, the bit width is fixed to 32 bit, where 16 bits are used for the integer and 16 bits for the fractional part. In the second phase, the bit widths are reduced linearly until they saturate at different values for each QLF. In the fine-tuning phase, there is a small increase in bit width again, since they get fixed to the next highest integer. The corresponding BER of the CNN is shown in Fig. 6. It can be seen that the BER is reduced continuously for all QLFs until training iteration 4000. Starting from this iteration, the low bit width starts to sacrifice the communication performance of the CNNs.

Especially for a QLF of 0.5 and 0.005, the BER increases dramatically. For a QLF of 0.005 and 0.0005 there is only a slight increase in BER. However, this increase can be compensated in the fine-tuning phase, where the quantized model nearly achieves the same BER as the full precision model. As a result of the quantization-aware training, our FPGA hardware architecture is based on a model with around 13 bits for weights and 10 bits for activations with approximately the same communication performance as the full precision model.

5 Hardware Architecture

In this section, the FPGA hardware architecture of our CNN-based equalizer is presented. We mainly focus on the high-throughput requirements of the optical fiber channel, but also show how the architecture can be applied to other application scenarios.

5.1 High-Throughput Architecture

The main target of our hardware implementation is to increase the throughput to meet the requirements of the 40GBd optical communication channel. To satisfy the strict throughput requirements, it is essential to use the FPGA's resources efficiently by increasing the utilization. Thus, the aim of our hardware architecture is to boost parallelism on all implementation levels. All those different levels of parallelism are illustrated in Fig. 7. The first level of parallelism is based on our streaming hardware architecture, where each of the L layers is implemented as an individual hardware instance. This way, the data is processed in a pipelined fashion, where each layer corresponds to a separate pipeline stage. Thus each layer can start its operation as soon as the first inputs are received which increases the throughput and the utilization of the available resources.

The core of our hardware architecture is a custom convolutional layer that is also optimized for high parallelism. The convolution operation can be described by:

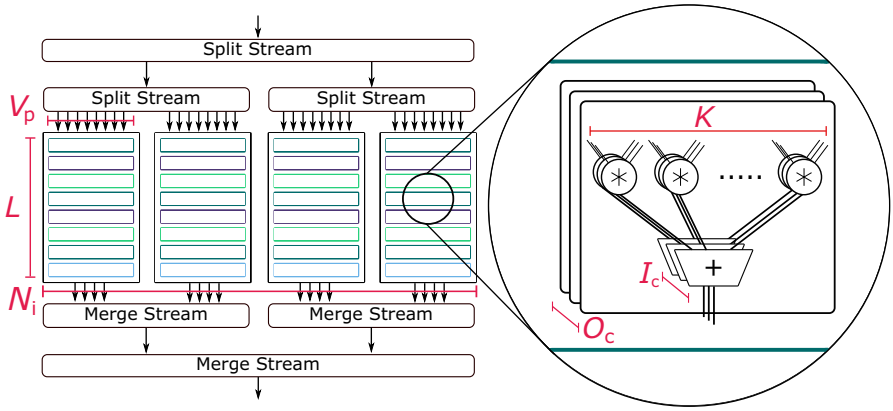


Fig. 7 Applied levels of parallelism of our hardware architecture illustrated for four instances

$$y_{o,j} = \sum_{i=0}^{I_c} \sum_{k=-\frac{K-1}{2}}^{\frac{K-1}{2}} x_{i,j+k} \cdot w_{i,o,k} \quad \forall o \in O_c \,, \tag{2}$$

where x is the input, y the output, w the kernel, K the kernel size, I_c the number of input channels and O_c the number of output channels. From (2), it can be seen that the convolutional layer offers multiple possibilities to apply spatial parallelism: on the level of input channels I_c, on the level of output channels O_c and on the kernel level K. Our hardware architecture of the convolutional layer exploits all of those parallelization options to achieve maximal throughput, as shown on the right of Fig. 7. Thus, it can achieve a throughput of one symbol per clock cycle. Since all of our layers are pipelined, this is also the throughput achieved by one hardware instance of the CNN. Another level of parallelism that is exploited by our implementation is the number of CNN instances N_i. We place and connect multiple instances of the CNN in one design to further boost the throughput. Therefore, the input is split into multiple streams, so each instance operates on a subset of the input sequence and produces a subset of the output sequence. Splitting and merging the sequence introduces further challenges, as explained later in Sect. 6.

In Fig. 7, we can see that increasing parallelism is a major objective across multiple design layers: starting from the topology, where V_p symbols are calculated in parallel, over the pipelined layers L and the parallelism with respect to K, O_c and I_c in the convolutional layer, up to the number of hardware instances N_i. This way several symbols are processed in parallel which is essential to reach the required throughput. In particular, the maximal throughput T_max in Gbit/s of our implementation is given as

$$T_\mathrm{max} = N_\mathrm{i} \cdot V_\mathrm{p} \cdot f_\mathrm{clk} \,.$$

We refer to maximal throughput here, as this throughput is only the theoretical upper limit, as explained later in Sec. 6.

5.2 Flexibility of Hardware Architecture

As described in Sect. 5.1, our implementation is well-suited for high-throughput scenarios. Additionally, due to the flexible design of our hardware architecture, it can also be applied to applications that operate at lower data rates. For those applications, reducing the power consumption or targeting low-cost FPGAs might be more relevant than increasing throughput. To also satisfy those requirements, our hardware architecture allows for variable DOPs based on the parallelization levels presented in Sect. 5.1. One example of a channel with a limited maximal throughput is the Proakis-B magnetic recording channel as described in Sect. 2.2. In the following, we show how our hardware architecture is able to adapt to less strict throughput requirements. As an implementation platform, we select the low-cost FPGA Xilinx XC7S25-1CSGA324.

For each hardware instance, parallelism can be applied to the input channels $\mathrm{DOP_I}$, the output channels $\mathrm{DOP_O}$, and to the kernel $\mathrm{DOP_K}$. The final DOP is then given as:

$\mathrm{DOP} = \mathrm{DOP_I} \cdot \mathrm{DOP_O} \cdot \mathrm{DOP_K}$. However, the individual DOPs are constrained by the hardware architecture as

$$\begin{aligned} \mathrm{I_c} &\equiv 0 \bmod \mathrm{DOP_I}, \\ \mathrm{O_c} &\equiv 0 \bmod \mathrm{DOP_O}, \\ \mathrm{DOP_K} &\in \{1, \mathrm{K}\}. \end{aligned}$$

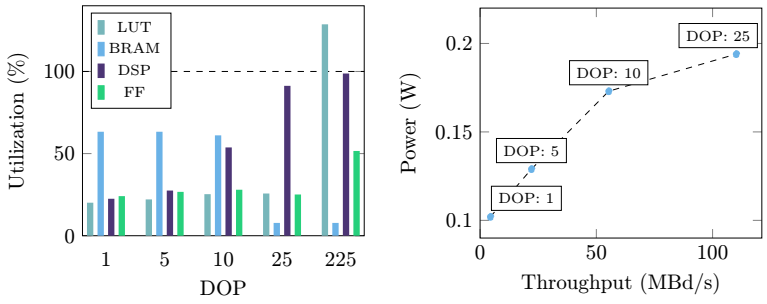
For our CNN topology this results in $\mathrm{DOP} \in \{1, 5, 10, 25, 225\}$.

In Fig. 8a the resourceutilization on the target FPGA for different DOPs is shown. It can be seen that primarily more LUTs and DSPs are required with higher DOP since more MAC operationsare performed in parallel. For a DOP of 255, all available DSPs are used, thus theMAC operations are implemented using LUTs which increases the LUT utilizationabove 100 %. Further, Vivado HLS implements the trainable parameters using blockrandom access memories (BRAMs) for the smaller DOPs and uses LUT resources asstorage for the larger DOPs. To summarize, Fig. 8a shows that our hardware architecture is also well suited for low-cost FPGAs as it can be adapted to exploit theavailable hardware resources.

In Fig. 8b we show how the DOP influences the dynamic power consumption andthe throughput of the implementation. A lower DOP results in fewer MAC operationsper clock cycle leading to lower throughput and lower power consumption. It canbe seen that one instance of the CNN on the XC7S25-1CSGA324 can be adjustedto achieve a throughput in the range of 4 Mbit·s^{-1}to 110 Mbit·s^{-1} while the powerranges from 0.1W to 0.2W.

5.3 Stream Partitioning

As explained in Sect. 5.1, for the high-throughput scenario our architecture provides a high level of parallelism, especially with respect to the number of CNN hardware instances. Splitting a stream of input symbols across those instances is not straightforward, since the ISI of the channel introduces an interdependence between consecutive symbols. Thus each CNN instance needs to operate on a continuous sequence of



(a) Resource utilization after synthesis.

(b) Dynamic power consumption vs. throughput

Fig. 8 Resource utilization and power consumption vs. throughput on the Xilinx XC7S25-1CSGA324 for different DOPs

input symbols. Therefore, we design a hardware module for splitting the input stream (split stream module (SSM)) and a hardware module for merging the output streams (merge stream module (MSM)), as shown in Fig. 9.

Each SSM takes an input stream and splits it into two streams of equal length. Multiple of those modules are arranged hierarchically to feed data to every CNN instance in a round-robin fashion. Due to the interdependence between consecutive symbols, splitting the input stream results in an increased BER at the border region of each sequence. Thus, the overlap generate module (OGM) adds an overlap to each sub-sequence. This way, the BER is approximately constant for the complete stream. Afterwards, the MSMs combine the divided sequences into one output stream. Then, the overlap is discarded by the overlap remove module (ORM). We arrange the SSMs and MSMs in hierarchical fashion instead of just implementing one module which operates on N_i streams. This improves the routability and timing of the design. Using only one module greatly increases the length of the paths from SSM and MSM to each of the CNN instances. In combination with regions of high congestion, this results in an enormous net delay, limiting the achievable clock frequency. By introducing multiple SSMs and MSMs, the critical paths are shortened, which increases the achievable clock frequency and therefore the obtainable throughput.

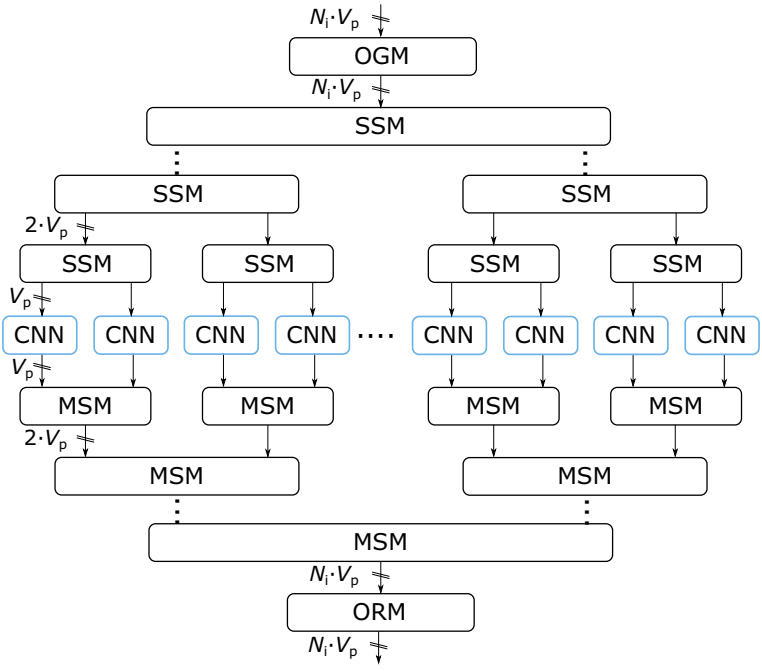


Fig. 9 Partitioning of the input sequence across multiple CNN instances

6 Sequence Length Optimization

As described in Sect. 5.3, each input sequence is divided into multiple sub-sequences of length ℓ_{inst} which are forwarded to the individual instances. However, choosing an optimal ℓ_{inst} under given application constraints is not straightforward. On the one side, an overlap of symbols is added at the beginning and the end of each sub-sequence. From this point of view, maximizing ℓ_{inst} results in the highest throughput since the overall overlap is minimized. On the other side, increasing ℓ_{inst} also increases the latency of the equalizer, which may violate low-latency constraints in applications like high-frequency trading or telemedicine. Thus it is important to choose ℓ_{inst} carefully, as described in the following.

6.1 Timing Model

To optimize ℓ_{inst}, we perform a detailed timing analysis of our hardware architecture. First, we analyze how many overlap symbols need to be added at the beginning and end of each sequence to compensate for the BER increase. For a CNN, the receptive field corresponds to the input symbols taken into account to predict each output. Thus, at the beginning and end of each sequence, half of the receptive field needs to be added as overlap. Based on the formula presented in [21], the number of overlap symbols for our network topology is calculated as

$$o_{\mathrm{sym}} = \frac{(K-1)\cdot(1+V_{\mathrm{p}}\cdot(L-1))}{2}\ .$$

However, this overlap needs to be added before the first SSM by the OGM, where the stream has a width of $N_{\mathrm{i}}\cdot V_p$ and has to be dividable by N_{os}, which equals to 2 in our case. Thus, the actual overlap can be calculated as

$$o_{\mathrm{act}} = \mathrm{nextEven}\left(\left\lceil \frac{o_{\mathrm{sym}}}{V_{\mathrm{p}}\cdot N_{\mathrm{i}}} \right\rceil\right)\cdot V_{\mathrm{p}}\cdot N_{\mathrm{i}}\ .$$

Therefore, the actual sequence length that needs to be processed including overlap is given as

$$\ell_{\mathrm{ol}} = \ell_{\mathrm{inst}} + 2\cdot o_{\mathrm{act}}\ .$$

As a second step, we analyze how ℓ_{ol} influences the time to fill the pipeline t_{init}, afterwards, we explain how this affects the latency of each symbol.

In Fig. 10, we show how ℓ_{ol} and therefore ℓ_{inst} impact the total time t to process one sequence. This time can be split into t_{init} and t_{p} and is illustrated for four instances. Since the width of the output streams of an SSM is half the width of the input stream and the sequences of length $\ell_{\mathrm{ol}}/V_{\mathrm{p}}$ are written alternately to each output, the writing to the second output stream only starts after $\ell_{\mathrm{ol}}/(2\cdot V_{\mathrm{p}})$ clock cycles. A similar behavior can be observed for each stage of the hierarchically arranged SSMs.

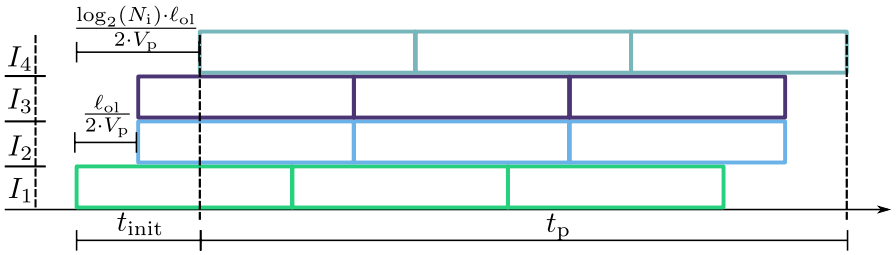


Fig. 10 Illustration of the processing time t_p and the time for filling the pipeline t_init for four CNN instances

Therefore, t_init, corresponding to the time where the last CNN instance starts processing, is given by

$$t_\mathrm{init} = \log_2(N_\mathrm{i}) \cdot \frac{\ell_\mathrm{ol}}{2 \cdot V_\mathrm{p} \cdot f_\mathrm{clk}} \,.$$

Thus, it can be seen that t_init increases linearly with ℓ_ol and ℓ_inst. Is it important to determine t_init, as it directly influences the latency to process one symbol λ_sym, which is given as the sum of the latency for splitting λ_spl, processing λ_pro and merging λ_mer. As the CNN is fully parallelized and there is no stalling in the merging of streams, λ_pro and λ_mer are neglectable. In contrast, for splitting, a stream of higher width is converted into streams of lower width, which results in stalling and increased latency. The maximum symbol latency can therefore be approximated as the time t_init to fill the pipeline:

$$\lambda_\mathrm{sym} \approx t_\mathrm{init} = \frac{\log_2(N_\mathrm{i}) \cdot \ell_\mathrm{ol}}{2 \cdot V_\mathrm{p} \cdot f_\mathrm{clk}} = \frac{\log_2(N_\mathrm{i}) \cdot (\ell_\mathrm{inst} + 2 \cdot o_\mathrm{act})}{2 \cdot V_\mathrm{p} \cdot f_\mathrm{clk}} \,. \tag{3}$$

From (3), it can be seen that higher ℓ_inst negatively impacts the symbol latency λ_sym. From this point of view, it would be beneficial to set ℓ_inst as small as possible.

However, since o_act is fixed and is added to each sub-sequence of length ℓ_inst, the total number of symbols to process by the CNN instances grows with shorter ℓ_inst. This is directly reflected in the processing time for one sequence of length ℓ_in, which is calculated as

$$t_p = \frac{\ell_\mathrm{in}}{\ell_\mathrm{inst} \cdot N_\mathrm{i}} \cdot \frac{\ell_\mathrm{inst} + 2 \cdot o_\mathrm{act}}{V_\mathrm{p} \cdot f_\mathrm{clk}} = \frac{\ell_\mathrm{in}}{N_\mathrm{i} \cdot V_\mathrm{p} \cdot f_\mathrm{clk}} \cdot \left(1 + \frac{2 \cdot o_\mathrm{act}}{\ell_\mathrm{inst}}\right) .$$

The processing time is inversely proportional to the net throughput:

$$T_\mathrm{net} = \frac{\ell_\mathrm{in}}{t_p} = \frac{N_\mathrm{i} \cdot V_\mathrm{p} \cdot f_\mathrm{clk}}{1 + \frac{2 \cdot o_\mathrm{act}}{\ell_\mathrm{inst}}} \,. \tag{4}$$

Thus, the net throughput grows with larger ℓ_{inst}. In summary, both the symbol latency λ_{sym} and the throughput T_{net} increase with ℓ_{inst}, therefore a trade-off exists when optimizing for throughput and latency.

6.2 Optimization Framework

Based on the equations presented in the previous section, we propose a framework to select the best ℓ_{inst} for the given application requirements. In our case, the throughput is a hard constraint that needs to be satisfied, while latency is an objective we want to minimize. Thus, the framework selects the minimal ℓ_{inst} which satisfies the throughput requirements.

The framework is shown in Fig. 11. Part of the framework, in particular the lookup table which maps the required throughput T_{req} to the optimal sub-sequence length ℓ_{inst}, is implemented as a module on the FPGA. This way, the best ℓ_{inst} can be selected during runtime individually for each sequence to process. The lookup table is provided by a lookup-table-generator based on the timing model presented in Sec. 6.1. The timing model is derived from a detailed analysis of the hardware architecture and therefore introduced hardware awareness to the LUT-Generator. This information is fed back to the hardware by selecting ℓ_{inst}. This way, our framework is also based on a cross-layer design methodology.

7 Results

In the following, the results of our timing model, our high-throughput (HT) FPGA implementation, and our low-power (LP FPGA implementation are presented. The main goal of our high-throughput hardware implementation is to achieve the throughput required for equalizing the 40GBd optical communication channel. As the channel data is captured with an upsampling factor of $N_{\text{os}} = 2$ at the receiver, this corresponds to 80 Gsamples/s at the input of our equalizer. To allow for such high data rate, we choose the least complex CNN model still satisfying our BER requirements in combination with multiple levels of parallelism of our FPGA architecture. Furthermore, we make use of our timing model and framework to estimate the number of instances needed to achieve the required throughput. As a hardware platform for the high throughput channel, we select the high-performance FPGA Xilinx XCVU13-P with a huge amount of available resources, to allow for extremely high parallelism. In contrast, for the LP FPGA implementation of the equalizer for the

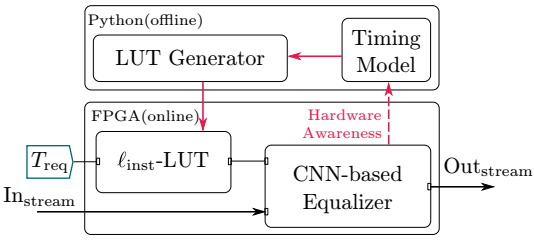


Fig. 11 Illustration of framework used to optimize sequence length per instance ℓ_{inst}

magnetic recording channel, the Xilinx XC7S25-1CSGA324 FPGA is used. For both implementations, Vitis HLS in combination with Vivado 2022.2 is used.

7.1 Timing Model Validation

In the following, we validate the correctness of our timing model by comparing it to real timing measurements. Moreover, we evaluate how many CNN instances are needed to achieve a throughput of 80 Gsamples/s with a clock frequency of 200MHz. In Fig. 12 we show how ℓ_{inst} influences the symbol latency λ_{sym} and the net throughput T_{net}. The blue stars are based on simulations of the hardware, while the black graphs correspond to our timing model. The horizontal lines of the throughput plot give the maximal theoretical throughput T_{max} as $\ell_{\mathrm{inst}} \to \infty$.

It can be seen that both the latency as well as the throughput increase with a higher number of instances N_{i}. While the latency grows linearly with the sequence length ℓ_{inst}, the throughput saturates for $\ell_{\mathrm{inst}} \to \infty$. The gap between T_{max} and T_{net} increases with N_{i} for a fixed ℓ_{inst}, which shows that it is necessary to select a larger ℓ_{inst} when increasing N_{i} to reduce the influence of the overlap symbols. Further, it is shown that our equations are close to the actual measurements. In particular, the difference between measurements and model is only around 6% for latency and 0.1% for throughput, which validates the accuracy of our timing model. Thus, based on our model, we can reliably predict that at least 64 instances are needed to achieve the required throughput of 80 Gsamples/s. In addition, it can be seen that ℓ_{inst} has a significant impact on T_{net}. Thus, ℓ_{inst} is an important factor to consider for increasing performance, which justifies the use of our framework to satisfy throughput requirements.

7.2 High-Throughput Implementation Results

Based on our timing model, we know that at least 64 instances are needed to achieve the required throughput. As a result of our detailed design space exploration and our advanced hardware architecture, we are actually able to place 64 parallel instances of the CNN model presented in Sect. 3.1 on the board. The instances are connected by

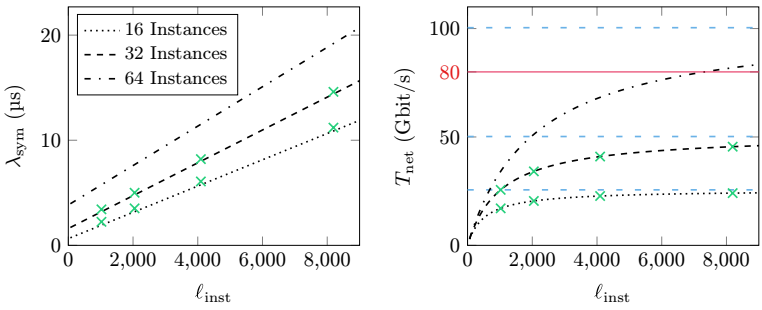


Fig. 12 Left: Plot of the sequence length ℓ_{inst} against the symbol latency λ_{sym}. Right: Plot of the sequence length ℓ_{inst} against the net throughput T_{net}. On the right, the maximal theoretical throughput is given by the horizontal dashed lines

63 SSMs and MSMs, respectively, to compose a common input and output stream. The maximal achievable clock frequency for the design is 200MHz. Thus, the maximal throughput is given by:

$$T_{\max} = N_i \cdot V_p \cdot f_{\text{clk}} = 64 \cdot 8 samples \cdot 200 MHz = 102 Gsamples/s \mathrel{\widehat{=}} 51 GBd\ .$$

Based on our framework presented in Sect. 6.2, the minimal ℓ_{inst} to achieve a net throughput of 80Gsym/s is determined, which is 7320. Choosing this sequence length results in the minimal symbol latency of only 17.5ms, while satisfying the throughput requirements.

In the following, we show the resource usage of our hardware architecture on the Xilinx XCVU13-P FPGA. In Table 1, we give the utilization of LUTs, flip-flop (FF), DSP, and BRAM after place and route. It can be seen that the resources with the highest utilization are DSPs and BRAMs. DSPs are utilized for the MAC operation in the convolutional layers, while the BRAMs are mainly used for splitting and merging the input streams. Increasing the number of instances further to achieve higher utilization results in routing congestion and a lower clock frequency, eventually reducing the achievable throughput.

7.3 Platform Comparison

In the following, we compare the throughput, latency, and power consumption achieved by our HT and LP FPGA implementations to other hardware platforms. As platforms we select the Nvidia RTX 2080 Ti high-performance GPU, the Nvidia AGX Xavier embedded GPU, and the Intel Core i9-9900KF high-performance CPU. For the CPU and the GPU, we increase the batch size and therefore the SPB up to the point where the throughput does not improve further. For the FPGAs, the SPB is fixed by the hardware architecture to 512 for the HT implementation and to 8 for the LP implementation. For fair comparison, for the GPUs we do not only evaluate the standard PyTorch implementation but also provide results for an optimized GPU implementation. This implementation is based on the Nvidia library TensorRT which builds a highly optimized model for faster inference. The optimizations performed by TensorRT include quantization, layer and tensor fusion, and kernel tuning.

7.3.1 Throughput

In Fig. 13, the throughput of the different platforms is compared. For similar SPB, the HT FPGA outperforms the CPU and all GPU implementations by orders of magnitude. For instance, the HT FPGA achieves a 4500 times higher throughput than the RTX TensorRT model for 400 SPB. The throughput of the LP FPGA is in the same order of magnitude as the AGX TensorRT model, the RTX PyTorch model,

Table 1 Utilization

LUT		FF		DSP		BRAM	
%	absolute	%	absolute	%	absolute	%	absolute
68.06	1176156	30.39	1050179	78.52	9648	78.79	2118

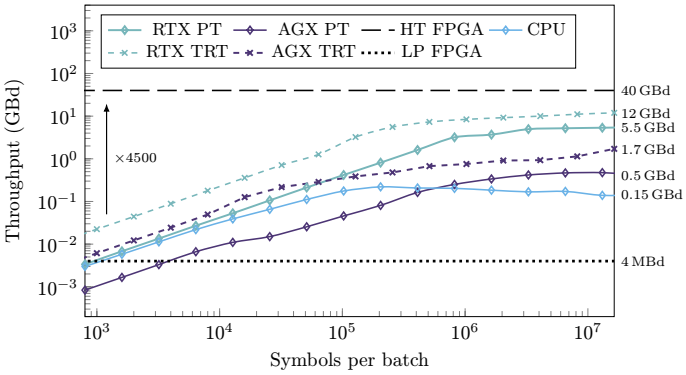


Fig. 13 Comparison of the throughput of the CNN-based equalizer, running on GPU, CPU and FPGA. PT refers to the PyTorch and TRT refers to the TensorRT models

and the CPU for SPB < 1000. However, for large SPB, the throughput of the high-performance platforms is much higher than that of the LP FPGA.

It can also be seen that TensorRT highly optimizes the throughput of the CNN models on GPU. Especially for low batch sizes, the throughput is around one order of magnitude higher as compared to the PyTorch model for both the RTX TensorRT model and the AGX Tensor RT model. For all implementations, the throughput increases linearly for low SPB and saturates for high SPB. The FPGA's throughput is constant over the whole range of SPB since the parallelization is fixed by the hardware architecture. Thus, parallelization does not increase with batch size but each batch is calculated sequentially on the FPGA. The highest throughput achieved by the conventional platforms is 12GBd by the RTX TensorRT implementation. However, even for high SPB, the FPGA outperforms the RTX TensorRT model by 10 times while the high-performance CPU is outperformed by more than two orders of magnitude.

7.3.2 Latency

In Fig. 14, the latency of the different implementations is compared. It can be seen that even for low SPB, the latency of the GPUs and CPU is more than one order of magnitude higher than that of the LP FPGA and around 5 × higher than that of the HT FPGAs. This gap increases for higher SPB up to a factor of 52 between the HT FPGA and the Nvidia AGX with the TensorRT model. As already seen in the throughput plot, TensorRT provides improved performance as compared to the default PyTorch model. Specifically, it decreased the latency of the models by around one order of magnitude.

7.3.3 Power

Besides throughput and latency, we compare the power of the different implementation approaches. For the FPGAs, the power is given by the Vivado power estimation tool, whereas for the Nvidia RTX GPU and the CPU we measure the power using

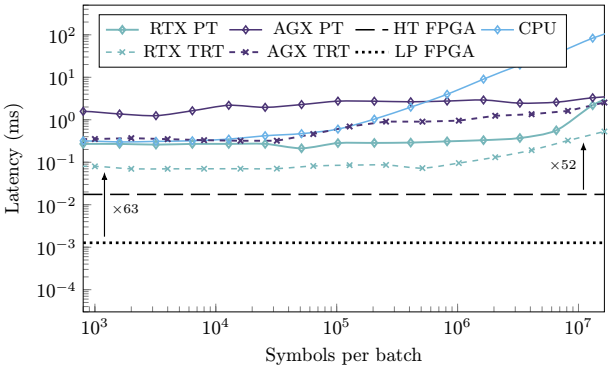


Fig. 14 Comparison of the latency of the CNN-based equalizer, running on GPU, CPU and FPGA. PT refers to the PyTorch and TRT refers to the TensorRT models

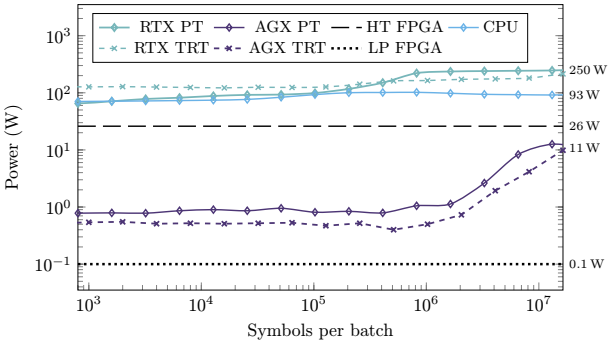


Fig. 15 Comparison of the power consumption of the CNN-based equalizer, running on GPU, CPU and FPGA. PT refers to the PyTorch and TRT refers to the TensorRT models

the PyJoules library [22]. For the Nvidia AGX platform, the power is obtained based on the jetson-stats package [23]. The results are shown in Fig. 15. It can be seen that the power consumption of the LP FPGA solution is orders of magnitude lower as compared to all other approaches. The AGX platform's power consumption lies in between the two FPGA approaches. Its power consumption is nearly constant up to 10^6 SPB, and from there on it starts to increase significantly. This is probably caused by under-utilization of the GPU for the small CNN topology for low batch sizes. The power consumption of the HT FPGA is around 2 × higher than that of the Nvidia AGX. For the high performance CPU and GPU, the power increases up to 93W and 250W respectively.

7.3.4 Summary

In summary, the comparison shows that our flexible FPGA architecture is able to provide efficient solutions for low-power and high-throughput application scenarios. In particular, the HT FPGA achieves higher throughput than all other platforms, including the high-performance GPU, even for very high SPB. Moreover, the LP

FPGA provides much lower power consumption as the other platforms. Main reason for the superior performance of the FPGAs is probably the efficient CNN inference hardware architecture, which is perfectly adapted to the specific CNN topology. In contrast, the other platforms provide a more general architecture that can also be utilized for other algorithms and use cases.

To conclude, the results show that the FPGA as a platform, in combination with a hardware architecture based on optimizations across all design layers, can provide a promising solution for implementing ANN-based algorithms for communications.

8 Conclusion

In this work, we present the FPGA implementation of a high-throughput CNN-based equalizer for optical communications. The implementation is based on optimization across all design layers, beginning with an extensive design-space exploration of the CNN, followed by a detailed quantization analysis, and culminating in the design of an efficient hardware architecture. As a result, the high-throughput FPGA implementation of our equalizer achieves a BER around 4 × lower than that of a conventional linear equalizer while meeting the high-throughput requirements of a 40GBd communication channel. Moreover, we demonstrate the flexibility of our custom hardware architecture by successfully applying our approach to a magnetic recording with a focus on low cost and low power. Further, we present a framework that optimizes the sequence length per instance to reduce the equalizer's latency under given throughput constraints. Finally, we compare our hardware implementation to optimized CPU and GPU implementations and show that the HT FPGA achieves a throughput that is three orders of magnitude higher than that of the high-performance GPU for a similar batch size. Moreover, we demonstrate that the same hardware architecture can also be applied to a low-power scenario, where an embedded GPU is outperformed in terms of power consumption.

Author Contributions J.N.: design space exploration, hardware implementation, result evaluation, wrote the original draft. C.F., V.L., S.R.: assembled the experimental setup, performed the experiments to generate the dataset. C.F., V.L., L.S., S.R., N.W.: detailed draft review and editing. S.R., L.S., N.W.: idea and formulation of research goals, acquisition of funding.

Funding Open Access funding enabled and organized by Projekt DEAL. Rheinland-Pfälzische Technische Universität Kaiserslautern-Landau (6375); This work was carried out in the framework of the CELTIC-NEXT project AI-NET-ANTILLAS (C2019/3-3) and was funded by the German Federal Ministry of Education and Research (BMBF) under grant agreements 16KIS1316 and 16KIS1317 as well as under grant 16KISK004 (Open6GHuB).

Data Availability No datasets were generated or analysed during the current study.

Declarations

Conflict of interest The authors have no conflict of interest as defined by Springer, or other interests that might be perceived to influence the results and/or discussion reported in this paper.

References

1. Bayvel, P., Maher, R., Xu, T., Liga, G., Shevchenko, N.A., Lavery, D., Alvarado, A., Killey, R.I.: Maximizing the optical network capacity. Phil. Trans. R. Soc. Lond. A: Math. Phys. Eng. Sci. **374**(2062), 20140440 (2016)
2. Ahmet Yazar, H.A., Seda Dogan, T.: 6G vision: an ultra-flexible perspective. ITU J. Future and Evolving Tech. **1**, 121–140 (2020). https://doi.org/10.52953/IKVY9186
3. Viswanathan, H., Mogensen, P.E.: Communications in the 6G era. IEEE Access **8**, 57063–57074 (2020). https://doi.org/10.1109/ACCESS.2020.2981745
4. Zerguine, A., Shafi, A., Bettayeb, M.: Multilayer perceptron-based DFE with lattice structure. IEEE Trans. Neural Netw. **12**(3), 532–545 (2001)
5. Schaedler, M., Bluemm, C., Kuschnerov, M., Pittalà, F., Calabrò, S., Pachnicke, S.: Deep neural network equalization for optical short reach communication. Appl. Sci. **9**(21), 4675 (2019)
6. Ney, J., Hammoud, B., Dörner, S., Herrmann, M., Clausius, J., ten Brink, S., Wehn, N.: Efficient FPGA implementation of an ANN-based demapper using cross-layer analysis. Electronics **11**(7) (2022)
7. Khan, F.N., Fan, Q., Lu, C., Lau, A.P.T.: An optical communication's perspective on machine learning and its applications. J. Lightwave Technol. **37**(2), 493–516 (2019)
8. Amari, A., Dobre, O.A., Venkatesan, R., Kumar, O.S.S., Ciblat, P., Jaouën, Y.: A survey on fiber nonlinearity compensation for 400 Gb/s and beyond optical communication systems. IEEE Commn. Sur. Tutorials **19**(4), 3097–3113 (2017). https://doi.org/10.1109/COMST.2017.2719958
9. Kuon, I., Rose, J.: Quantifying and Exploring the Gap Between FPGAs and ASICs. Springer, New York (2010)
10. Freire, P.J., Srivallapanondh, S., Anderson, M., Spinnler, B., Bex, T., Eriksson, T.A., Napoli, A., Schairer, W., Costa, N., Blott, M., Turitsyn, S.K., Prilepsky, J.E.: Implementing neural network-based equalizers in a coherent optical transmission system using field-programmable gate arrays. J. Lightwave Technol. **41**(12), 3797–3815 (2023). https://doi.org/10.1109/JLT.2023.3272011
11. Kaneda, N., Chuang, C.-Y., Zhu, Z., Mahadevan, A., Farah, B., Bergman, K., Van Veen, D., Houtsma, V.: Fixed-point analysis and FPGA implementation of deep neural network based equalizers for high-speed PON. J. Lightwave Technol. **40**(7), 1972–1980 (2022)
12. Li, M., Zhang, W., Chen, Q., He, Z.: High-throughput hardware deployment of pruned neural network based nonlinear equalization for 100-Gbps short-reach optical interconnect. Opt. Lett. **46**(19), 4980–4983 (2021)
13. Huang, X., Zhang, D., Hu, X., Ye, C., Zhang, K.: Low-complexity recurrent neural network based equalizer with embedded parallelization for 100-Gbit/sλPON. J. Lightwave Technol. **40**(5), 1353–1359 (2022)
14. Ney, J., Lauinger, V., Schmalen, L., Wehn, N.: Unsupervised ANN-based equalizer and its trainable FPGA implementation. In: 2023 Joint European Conference on Networks and Communications & 6G Summit (EuCNC/6G Summit), pp. 60–65 (2023). https://doi.org/10.1109/EuCNC/6GSummit58263.2023.10188269
15. Ney, J., Füllner, C., Lauinger, V., Schmalen, L., Randel, S., Wehn, N.: From algorithm to implementation: Enabling high-throughput CNN-based equalization on FPGA for optical communications. In: Silvano, C., Pilato, C., Reichenbach, M. (eds.) Embedded Computer Systems: Architectures, Modeling, and Simulation, pp. 158–173. Springer, Cham (2023)

16. Owaki, S., Nakamura, M.: Equalization of optical nonlinear waveform distortion using neural-network based digital signal processing. In: 2016 21st OptoElectronics and Communications Conference (OECC) Held Jointly with 2016 International Conference on Photonics in Switching (PS), pp. 1–3 (2016)
17. Estaran, J., Rios-Mueller, R., Mestre, M.A., Jorge, F., Mardoyan, H., Konczykowska, A., Dupuy, J.-Y., Bigo, S.: Artificial neural networks for linear and non-linear impairment mitigation in high-baudrate IM/DD systems. In: ECOC 2016; 42nd European Conference on Optical Communication, pp. 1–3 (2016)
18. Eriksson, T.A., Bülow, H., Leven, A.: Applying neural networks in optical communication systems: possible pitfalls. IEEE Photonics Technol. Lett. **29**(23), 2091–2094 (2017). https://doi.org/10.1109/LPT.2017.2755663
19. Proakis, J.G., Salehi, M.: Digital Communications, 5th edn. McGraw-Hill Higher Education, New York (2008)
20. Nikolic, M., Hacene, G.B., Bannon, C., Lascorz, A.D., Courbariaux, M., Bengio, Y., Gripon, V., Moshovos, A.: Bitpruning: Learning bitlengths for aggressive and accurate quantization. (2020)
21. Araujo, A., Norris, W.D., Sim, J.: Computing receptive fields of convolutional neural networks. Distill (2019)
22. Belgaid, M.c., Rouvoy, R., Seinturier, L.: Pyjoules: Python library that measures python code snippets (2019). https://github.com/powerapi-ng/pyJoules
23. Bonghi, R.: jetson-stats. https://github.com/rbonghi/jetson_stats

Authors and Affiliations

Jonas Ney[1] · Christoph Füllner[2] · Vincent Lauinger[3] · Laurent Schmalen[3] · Sebastian Randel[2] · Norbert Wehn[1]

✉ Jonas Ney
jonas.ney@rptu.de

Christoph Füllner
christoph.fuellner@kit.edu

Vincent Lauinger
vincent.lauinger@kit.edu

Laurent Schmalen
laurent.schmalen@kit.edu

Sebastian Randel
sebastian.randel@kit.edu

Norbert Wehn
norbert.wehn@rptu.de

[1] Microelectronic Systems Design (EMS), RPTU Kaiserslautern-Landau, 67653 Kaiserslautern, Germany

[2] Institute of Photonics and Quantum Electronics (IPQ), KIT, 76131 Karlsruhe, Germany

[3] Communications Engineering Lab (CEL), KIT, 76131 Karlsruhe, Germany