CHALMERS
UNIVERSITY OF TECHNOLOGY

FPGA-Based Wordlength Optimization for DSP

FPGA-Based Wordlength Optimization for DSP

Chenhao Yang, Jinsheng Bian, Erik Börjeson, and Per Larsson-Edefors
Chalmers University of Technology
Gothenburg, Sweden

***Abstract*—Fixed-point representations are commonly used in DSP designs to efficiently use hardware resources. It is, however, a challenge to determine an appropriate fractional wordlength of all signals in order to reach a good balance between accuracy and hardware cost. Extensive simulations can be used to characterize a design and perform wordlength optimization (WLO), however, this tends to be slow when the DSP design is complex. FPGA emulation, which allows data to be streamed in hardware, is significantly faster than software simulation. We introduce an FPGA-accelerated WLO framework which utilizes a new WLO algorithm based on a tree-structured Parzen estimator developed for higher convergence speed. This framework is evaluated for three DSP designs; two finite-impulse response filters and one phase recovery design. The results show that our new WLO framework reduces the DSP accuracy evaluation time by a factor of 300–500 over simulation.**

I. Introduction

Increasing transistor density enables more functionalities to be implemented on application-specific integrated circuits (ASICs) and field-programmable gate arrays (FPGAs). Digital signal processing (DSP) applications on such platforms usually involve casting floating-point operations into fixed-point operations in order to respect area/resource and power budgets. However, using fixed point leads to a reduction in the quality of results due to lower accuracy, which means that the length of the integer and fractional parts of each fixed-point word need to be adjusted so that the performance specification is satisfied. This is especially important in ASICs whose logic gate structure is designed to match the signal resolution at the bit level.

The optimization problem we are facing is to find the best wordlength configuration for all signals so that the DSP design meets the accuracy specification at the minimum hardware cost. Similar to other optimization problems, the cost is expected to be minimal with the constraint of given lower-bound accuracy. Consequently, the *wordlength optimization (WLO)* problem can be described as:

$$C_{min}(W) \text{ subject to } \lambda(W) \geq \lambda_{min} \tag{1}$$

where W, C and λ represents the wordlength configuration, the cost (such as area or power), and the accuracy, respectively.

There are many WLO methods—both analytical and simulation based—proposed in the literature [1]. Analytical approaches solve a cost function that empirically or theoretically models systems with low complexities, based on approximations of accuracy, hardware resources, and power dissipation, whereas the simulation-based approaches conduct iterative search using simulations. The latter approach is more accurate and various algorithms have emerged, for instance, the greedy deterministic algorithms Min+1 and Max-1 [2] which vary the wordlength in steps of 1 bit. Nguyen et al. combined the methods above with Tabu search, which enables searches in both descending and ascending directions [3]. General optimization algorithms have also been used, such as in [4] where simulated annealing was modified to fit WLO problems.

The machine-learning-based optimization method of Bayesian optimization (BO) is widely used in problems where the loss function is a black box [5]. In the framework of WLO, BO methods can estimate the loss function based on the given data (e.g. estimated area from logic synthesis) and find an optimum value by updating the predicted loss value iteratively. The BO method mainly consists of two parts: i) A surrogate model, such as tree-structured Parzen estimator (TPE), to estimate the loss function. ii) An acquisition function to select the next query point during simulation. BO has been applied to WLO problems [6] and Ha et al. [7] later proposed a resource-constrained BO scheme to maximize computing accuracy with a restriction on resources.

Improving the search efficiency of the algorithms can reduce the number of redundant and time-consuming logic synthesis runs, therefore accelerating the overall optimization process. But regardless of this, for complex DSP designs, simulation-based approaches also run into problems with long run times. To this end, FPGA-based acceleration for the purpose of wordlength optimization of DSP designs was recently proposed [8], however, this WLO system is incomplete and no quantitative run-time gains are provided.

To enable fast WLO of parameterized complex DSP designs, we present a novel wordlength optimization approach which uses FPGA acceleration orchestrated from a computer running a variant of TPE, which has faster convergence speed. We will compare the new TPE approach with two reference algorithms in terms of convergence speed, and evaluate the overall FPGA-accelerated WLO framework on three DSP designs. In addition, the possibility of further speeding up the optimization by batch evaluation is discussed in the results.

II. Design of WLO Framework

Our proposed FPGA-accelerated WLO framework consists of a WLO program running on a computer, an FPGA, and a server with ASIC synthesis tools. The FPGA is used to emulate the target DSP design, evaluate the accuracy of the output with different wordlength configurations, and send the evaluation result to the computer. The synthesis server generates area estimation results, which represent the cost in the loss function. The WLO program conducts the optimization process based on received area and accuracy evaluation results.

A. Hardware Aspects

The block diagram for our WLO framework is shown in Fig. 1. The `Control Unit` receives wordlength configurations for the DSP design from the PC and sends the accuracy evaluation result back. The `Bit Switch (BS)` modules control the wordlengths of the data path, where the unused bits are set to 0 using AND operations. In the block marked DSP design, BS modules are integrated to change the wordlengths for internal signals of the DSP design. The block diagram shows an example emulation structure; the actual implementation will vary according to the actual requirements. The `Accuracy Evaluator` collects output data from the DSP design and calculates the accuracy (using metrics like mean square error, MSE, or bit error rate, BER) of the output.

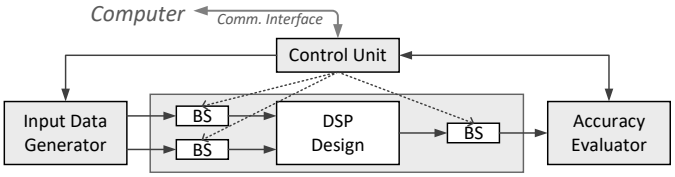

Fig. 1. Hardware part of our FPGA-accelerated WLO framework.

B. Algorithm Aspects

We will here explain aspects of the WLO algorithm including how it is modified for faster convergence.

a) Surrogate model: To reduce calculations when estimating the posterior, the tree-structured Parzen estimator proposed in [9] is used as a new surrogate model to estimate the likelihood. It is defined as

$$p(x|y) = \begin{cases} l(x) & y < y^* \\ g(x) & y \geq y^* \end{cases} \tag{2}$$

where the probability distribution of observed data is divided into two sub-distributions, $l(x)$ where the loss value is less than y^* and $g(x)$ where the loss value is greater than y^*, and y^* is the dividing criteria defined by $p(y < y^*) = \gamma$, and γ is the given cumulative probability.

The Gaussian mixture model (GMM) of $\mathcal{D}^{(g)}$ (i.e. $g(x)$) is constructed by the evenly weighted sum of multivariate Gaussian distributions. Conversely, GMM of $\mathcal{D}^{(l)}$ (i.e. $l(x)$) is constructed by the unevenly weighted sum of multivariate Gaussian distributions. The weights are

$$w = \frac{2(N-n)}{N(N+1)}, \quad n = 0, 1, \ldots, N-1 \tag{3}$$

where N is the number of observations in $\mathcal{D}^{(l)}$, and observations in $\mathcal{D}^{(l)}$ are ordered from best to worst and indexed from 0 to $N-1$.

b) Acquisition function: Applying the expected improvement (EI) to TPE [9], the next query point corresponds to the position of the maximum value of $\frac{l(x)}{g(x)}$. However, the number of values in the GMMs within the search space increases exponentially with increasing dimensions, resulting in high demands for computing resources. Instead, sampling is used to approximate the maximum [10]; the result is subsequently improved by applying stochastic gradient descent-ascent (SGDA) [11].

c) Loss Function: In [6], [7], the cost metric is the energy estimated by the number of operations on each operator. However, the estimated energy may vary with different input data. To mitigate this influence, we use the area after synthesis.

Here, metrics are combined into the loss function by multiplication, where better accuracy and area always lead to a lower loss value. Since synthesis may take a long time, a range for the accuracy metric is set to avoid performing synthesis of unreasonable wordlength configurations. The loss function is described as

$$loss = \begin{cases} |accu - target| \times area & LB \leq accu \leq UB \\ |accu - target| \times C & \text{otherwise} \end{cases} \tag{4}$$

where $accu$ is the accuracy metric, $target$ is the optimization target for the accuracy and, $area$ is the area from synthesis. Additionally, LB (lower bound) and UB (upper bound) define the expected range of the accuracy, whereas C is a large constant (larger than the area when the accuracy is in the defined range).

III. Results

In this section, the faster convergence speed of our new TPE variant is first demonstrated with benchmarks. After verifying the consistency of accuracy evaluation results from simulations and FPGA emulations, we apply the new TPE variant in the WLO framework and evaluate and contrast evaluations based on simulation and FPGA acceleration.

A. Evaluation of WLO Algorithms

Table I shows the benchmarks selected to evaluate the speed of convergence and the quality of solution of WLO algorithms.

TABLE I
Equations of benchmarks.

Benchmark	Equation
Rastrigin	$\sum_{i=0}^{N-1} \left(\frac{x_i}{3}\right)^2 - 10\cos(2\pi x_i/3) + 10N$
Rosenbrock	$\sum_{i=0}^{N-2} 100\,(x_{i+1} - x_i)^2 + (1 - x_i)^2$
Sphere	$\sum_{i=0}^{N-1} x_i^2$
Styblinski-Tang	$\sum_{i=0}^{N-1} \frac{(x_i/3)^4 - 16(x_i/3)^2 + 5(x_i/3)}{2}$

Table II presents the result of the algorithm evaluation, in which our TPE variant is called TPE-WLO and in which references include Watanabe's TPE [10], [12] and Optuna developed by Akiba et al. [13], [14]. In the comparison, the search space is set to integers from -16 to 16 across dimensions of 5D, 10D, 15D, and 30D.

TABLE II
Speedups of Watanabe's TPE and TPE-WLO. The four digits in each function represent the speedups in 5D, 10D, 15D, and 30D. "-" means no better solution than the baseline.

	Sphere	Rastrigin
Baseline (Optuna)	1.0, 1.0, 1.0, 1.0	1.0, 1.0, 1.0, 1.0
Watanabe's TPE	4.2, 6.5, 7.9, 5.5	1.5, 1.3, 1.4, 2.5
TPE-WLO	5.6, 7.5, 12.0, 16.7	3.0, 5.6, 7.3, 14.3
	Rosenbrock	Styblinski-Tang
Baseline (Optuna)	1.0, 1.0, 1.0, 1.0	1.0, 1.0, 1.0, 1.0
Watanabe's TPE	2.8, 5.7, 6.1, 5.4	-, 1.6, 1.4, 1.4
TPE-WLO	4.7, 12.0, 6.5, 17.6	-, 1.5, 1.8, 2.8

The maximum number of evaluations is set to 400, and the objective values of 300 evaluations by Optuna are selected as

baseline. The speedup is calculated through 300 divided by the minimum number of evaluations taken by an algorithm to achieve equal or better results. To reduce the impact of random algorithm features, the results represent an average of 20 runs. With the exception of the 10D Styblinski-Tang function, our TPE-WLO algorithm provides faster convergence speed over Watanabe's TPE and the Optuna baseline.

B. Evaluation of WLO Frameworks

The target DSP designs in this evaluation are one Viterbi-Viterbi phase recovery DSP [15] (called VV DSP) using QPSK partitioning, and two finite-impulse response (FIR) filters. We select these as examples of two important classes of DSP: Heterogeneous DSP structures that combine estimation and filtering, and homogeneous filter structures. To further exploit the hardware resources of FPGA and speed up the optimization process, we investigate *batch evaluation* for two FIR filters. Batch evaluation allows parallel emulation of DSP designs on FPGAs. In the following, the notation *Batch 1* means there is only one wordlength configuration of the DSP design under emulation, and *Batch 2* means two wordlength configurations of the DSP design are emulated. In all these tests, simulations are carried out in Modelsim 20.1.1 on a laptop running Debian 12 with Intel Core i5-11300H and 16GB RAM. The synthesis is finished in Cadence Genus on a server running Red Hat Enterprise Server 7.9 with Intel Xeon Gold 6226R and 756GB RAM. We use the open-source ASAP7 regular VT cell library [16] for synthesis.

a) Phase recovery DSP: Fig. 2 shows the FPGA-accelerated system (based on the open-source CHOICE project [17]) used to evaluate the phase recovery DSP. On FPGA, this DSP unit is driven by data created via a transmitter, containing a random number generator and a modulator, and a synthetic channel, in which synthetic additive white Gaussian noise and phase noise are added. At the output, after demodulation, the analysis unit evaluates the accuracy in terms of BER.

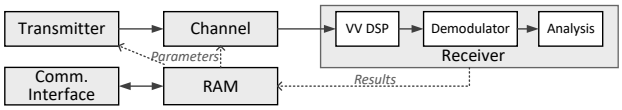

Fig. 2. FPGA-acceleration framework for VV DSP.

We perform two different evaluations: one based on the FPGA acceleration outlined above and another based on simulation, where all data generation for simulation is running in Python, but VV DSP is running in the logic simulator (Modelsim). There are some minor differences in how data bits are generated in the two evaluations, but the result differences are insignificant. In these evaluations, we use 1.5 million bits per run, the accuracy target BER is set to 0.012, and the accuracy range of BER is set to $[0.0118,\ 0.0122]$. The maximum number of evaluations is set to 100 including 16 initial points. There are five signals to be optimized (see [15]): the magnitude and partitioned output signals can take on values from 2 to 8 bits, the 2^{nd}-power and the 4^{th}-power output values from 2 to 12 bits, and the phase output values from 2 to 10 bits.

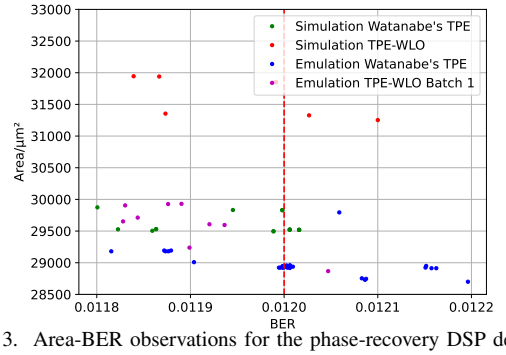

Fig. 3. Area-BER observations for the phase-recovery DSP design.

Fig. 3 shows the observation distribution for area from synthesis versus BER from simulation and FPGA emulation without batch evaluation, while Fig. 4 shows the trend of the minimum loss function value as the number of evaluations grows. In both simulation and FPGA-accelerated frameworks, Watanabe's TPE provides a faster convergence speed and better search results. Due to the input data for FPGA emulations and software simulations being slightly different, the final results also differ.

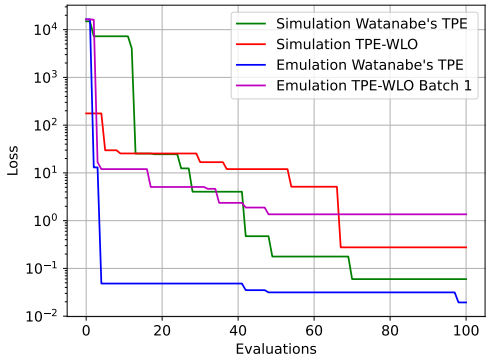

Fig. 4. Loss function trend for VV DSP.

The observations of Watanabe's TPE concentrate around $\text{BER} = 0.012$ as the loss function suggests. However, the observations of TPE-WLO do not show the explicit accumulation around the target due to worse exploration. In the simulation-based tests, the total time spent is 3,612 seconds for Watanabe's TPE and 3,207 seconds for TPE-WLO. It should be noted that in these tests, when the same wordlength configurations occur twice or more (and a netlist already exists) or the accuracy result is not in the desired range, the synthesis process is skipped altogether. In simulation, for example, there are in total 16 synthesis calls of which 7 are skipped for Watanabe's TPE and 7 synthesis calls of which 1 is skipped for TPE-WLO. In the FPGA-accelerated framework, the total evaluation time is 3,180 seconds for Watanabe's TPE and 2,687 seconds for TPE-WLO.

b) FIR Filters: The other two DSP designs used are FIR filters with 15 and 30 coefficients (14^{th}- and 29^{th}-order filters) in a transposed structure. The metrics used for optimization are

mean square error (MSE) and area. The signals to be optimized are the outputs of the tap multipliers, which consume the most area. In the test of 14^{th}-order FIR filter, the initial points and total evaluations are set to 16 and 250, respectively. The accuracy target MSE is $4.5 \cdot 10^{-5}$, and the desired accuracy range of MSE is set to $[3 \cdot 10^{-5},\ 6 \cdot 10^{-5}]$. The 15 tap multipliers have their output signals varying from 0 to 16 bits.

Fig. 5 shows that in both simulation and emulation, our TPE-WLO algorithm has a faster convergence speed in the beginning, but Watanabe's TPE can lead to a better solution in the following iterations.

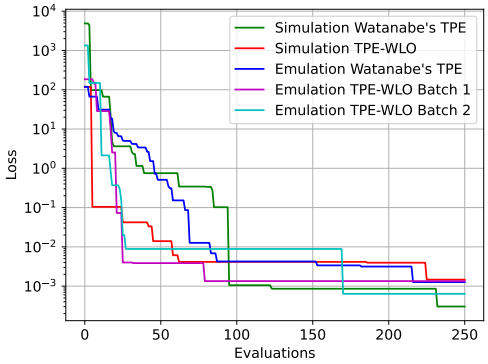

Fig. 5. Loss function trend for 14^{th}-order FIR filter.

The simulation-based framework with Watanabe's TPE and TPE-WLO takes 1,754 and 1,673 seconds, respectively. The FPGA-accelerated approach with Watanabe's TPE takes 1,056 seconds, while the TPE-WLO approach takes 1,129 and 894 seconds when the batch size is 1 and 2, respectively. The optimization process is expected to be faster when increasing the batch size, but the result does not explicitly confirm this expectation. This is because the random factors in the algorithm part leads to different numbers of synthesis runs, which create variations in the optimization time.

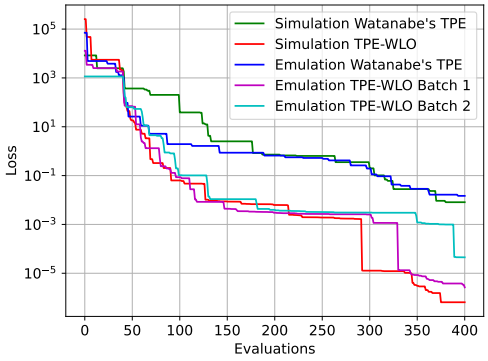

Fig. 6. Loss function trend for 29^{th}-order FIR filter.

In the test of the 29^{th}-order FIR filter, the initial points and total evaluations are set to 40 and 400, the accuracy range of MSE is set to $[5 \cdot 10^{-10},\ 5 \cdot 10^{-9}]$, and the target MSE is 10^{-9}. The wordlength of signals can vary from 0 to 24 bits. Watanabe's TPE fails to converge to the accuracy range in this case, as shown in Fig. 6, whereas TPE-WLO is able to converge to the accuracy range in both frameworks. However, the exploration time reduction due to larger batch size is more dramatic in this case. The time for the simulation-based approach with TPE-WLO is 14,115 seconds. In contrast, the time for the FPGA-accelerated framework is 5,785 and 484 seconds for a batch size of 1 and 2, respectively. The significant time difference is caused by the difference in the number of synthesis runs as mentioned above.

C. Speedup by FPGA Acceleration

The synthesis process we use to estimate area in some cases dominates the WLO run time above. To quantify the true speedup brought on by FPGA emulations for accuracy analysis, we perform the same tests as above but this time with a proxy cost metric: the proxy cost we use is the aggregated wordlength, i.e. the sum of all wordlengths that are being optimized, removing the time for synthesis.

TABLE III
TIME (SECONDS) SPENT ON SIMULATION-BASED AND THE FPGA-ACCELERATED FRAMEWORK WHEN A PROXY COST IS USED.

	Simulation-based		Batch size: 1		Batch size: 2	
	Sim	Total	Emu	Total	Emu	Total
VV DSP	819.01	820.92	1.68	49.87	-	-
14^{th}-order FIR	626.61	634.74	1.83	10.06	1.21	5.64
29^{th}-order FIR	1760.73	1782.43	3.49	25.36	2.45	13.48

The result is presented in Table III, where the total simulation time includes also the time for running the algorithm and finding the cost. The total emulation time is dominated by the communication time, neglecting the remote communication cost of VV DSP which is run on a remote FPGA causing longer total emulation time than others. The result demonstrates a reduction in the accuracy evaluation time of DSP designs by a factor of 504 for the 29^{th}-order FIR filter, 342 for the 14^{th}-order FIR filter, and 487 for the phase recovery DSP design, when the batch size is 1. It is clear that the time spent on accuracy evaluations can be significantly reduced by FPGA emulations. It is also clear that further time reductions would be possible with larger batch sizes, if the overhead of data transmissions is reduced.

IV. Conclusion

In this paper, we proposed a new variant of the TPE algoritm and developed an FPGA-accelerated WLO framework targeting at general DSP designs. The main contributions of the paper are the fast convergence speed of the algorithm in the benchmarks and the acceleration for the evaluation of DSP designs during wordlength optimization. The convergence speed was compared to two reference algorithms, viz. Watanabe's TPE and Optuna, whereas our WLO framework was evaluated using three different DSP designs and two accuracy evaluators. Through these tests, the new TPE variant demonstrated faster convergence into the desired accuracy range when the search space was large, consistent with the benchmark results. Additionally, the FPGA emulation can greatly speed up DSP evaluation.

REFERENCES

[1] H. Choi and W. Burleson, "Search-based wordlength optimization for VLSI/DSP synthesis," in *IEEE Workshop on VLSI Signal Processing*, 1994, pp. 198–207.

[2] M.-A. Cantin, Y. Savaria, D. Prodanos, and P. Lavoie, "An automatic word length determination method," in *IEEE Int. Symp. on Circuits and Systems (ISCAS)*, vol. 5, 2001, pp. 53–56 vol. 5.

[3] H.-N. Nguyen, D. Menard, and O. Sentieys, "Novel algorithms for word-length optimization," in *European Signal Processing Conf.*, 2011, pp. 1944–1948.

[4] D.-U. Lee, A. Gaffar, R. Cheung, O. Mencer, W. Luk, and G. Constantinides, "Accuracy-guaranteed bit-width optimization," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, no. 10, pp. 1990–2000, 2006.

[5] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. de Freitas, "Taking the human out of the loop: A review of Bayesian optimization," *Proc. of the IEEE*, vol. 104, no. 1, pp. 148–175, 2016.

[6] V.-P. Ha and O. Sentieys, "Leveraging Bayesian optimization to speed up automatic precision tuning," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2021, pp. 1542–1547.

[7] ——, "Maximizing computing accuracy on resource-constrained architectures," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2023.

[8] J. Hormigo and G. Caffarena, "FPGA acceleration of bit-true simulations for word-length optimization," in *IEEE Symp. on Computer Arithmetic (ARITH)*, 2021, pp. 119–122.

[9] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," in *Int. Conf. on Neural Information Processing Systems*, ser. NIPS'11, 2011, p. 2546–2554.

[10] S. Watanabe, "Tree-structured Parzen estimator: Understanding its algorithm components and their roles for better empirical performance," 2023. [Online]. Available: https://arxiv.org/abs/2304.11127

[11] H. Robbins and S. Monro, "A stochastic approximation method," *The Annals of Mathematical Statistics*, vol. 22, no. 3, pp. 400–407, 1951. [Online]. Available: https://doi.org/10.1214/aoms/1177729586

[12] The tree-structured Parzen estimator (TPE) implementation, 2023, https://github.com/nabenabe0928/tpe.

[13] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *ACM SIGKDD Int. Conf. on Knowledge Discovery & Data Mining*, 2019, p. 2623–2631.

[14] Optuna: A hyperparameter optimization framework, 2018, https://github.com/optuna/optuna.

[15] E. Börjeson and P. Larsson-Edefors, "Energy-efficient implementation of carrier phase recovery for higher-order modulation formats," *Journal of Lightwave Technology*, vol. 39, no. 2, pp. 505–510, 2021.

[16] V. Vashishtha, M. Vangala, and L. T. Clark, "ASAP7 predictive design kit development and cell design technology co-optimization: Invited paper," in *IEEE/ACM Int. Conf. on Computer-Aided Design (ICCAD)*, 2017, pp. 992–998.

[17] CHOICE - Chalmers Optical Fiber Channel Emulator, 2022, www.cse.chalmers.se/research/group/vlsi/choice.