Fixed-point realisation of fast nonlinear Fourier transform algorithm for FPGA implementation of optical data processing

Vasylchenkova, Anastasiia, Salnikov, Dmytro, Karaman, Dmytro, Vasylchenkov, Oleg, Prilepskiy, Jaroslaw

Anastasiia Vasylchenkova, Dmytro Salnikov, Dmytro Karaman, Oleg G. Vasylchenkov, Jaroslaw E. Prilepskiy, "Fixed-point realisation of fast nonlinear Fourier transform algorithm for FPGA implementation of optical data processing," Proc. SPIE 11770, Nonlinear Optics and Applications XII, 1177016 (18 April 2021); doi: 10.1117/12.2588735

Event: SPIE Optics + Optoelectronics, 2021, Online Only

Fixed-point Realisation of Fast Nonlinear Fourier Transform Algorithm for FPGA Implementation of Optical Data Processing

Anastasiia Vasylchenkova[a], Dmytro Salnikov[b], Dmytro Karaman[b], Oleg G. Vasylchenkov[b], and Jaroslaw E. Prilepsky[c]

[a]Optical Networks Group, University College London, London WC1E 7JE, UK
[b]National Technical University "Kharkiv Polytechnic Institute", Kharkiv 61002, Ukraine
[c]Aston Institute of Photonic Technologies, Aston University, Birmingham B4 7ET, UK

ABSTRACT

The nonlinear Fourier transform (NFT) based signal processing has attracted considerable attention as a promising tool for fibre nonlinearity mitigation in optical transmission. However, the mathematical complexity of NFT algorithms and the noticeable distinction of the latter from the "conventional" (Fourier-based) methods make it difficult to adapt this approach for practical applications. In our work, we demonstrate a hardware implementation of the fast direct NFT operation: it is used to map the optical signal onto its nonlinear Fourier spectrum, i.e. to demodulate the data. The main component of the algorithm is the matrix-multiplier unit, implemented on field-programmable gate arrays (FPGA) and used in our study for the estimation of required hardware resources. To design the best performing implementation in limited resources, we carry out the processing accuracy analysis to estimate the optimal bit width. The fast NFT algorithm that we analyse, is based on the FFT, which leads to the $\mathrm{O}(N \log_2^2 N)$ method's complexity for the signal consisting of N samples. Our analysis revealed the significant demand in DSP blocks on the used board, which is caused by the complex-valued matrix operations and FFTs. Nevertheless, it seems to be possible to utilise further the parallelisation of our NFT-processing implementation for the more efficient NFT hardware realisation.

Keywords: FPGA, nonlinear Fourier transform, digital signal processing

1. INTRODUCTION

Nonlinear effects in an optical fibre are widely considered as one of the most important factors degrading the performance of high-rate optical transmission systems.[1–3] Thus, there has been proposed a plethora of various approaches to mitigate these impairments, e.g. the digital back-propagation (DBP),[4–6] and, more recently, machine learning (ML) based methods[7] and NFT usage.[8,9] Such techniques are typically based on the pre- and/or post-compensation of the signal in the digital domain. However, the common challenge for the digital methods used for nonlinearity mitigation is the considerable numerical complexity of the algorithms (especially when we aim to get a high improvement of performance), and the intolerance to numerical processing error.[10] Therefore, effective real-time processing requires significant problem-specific hardware designing efforts as we have to account for the fixed-point arithmetic features,[11] and also address the parallel processing flexibility and efficient memory allocation.

The NFT-based transmission approach is different from the other nonlinearity mitigation methods because it inherently implies the constrictive use of the fibre nonlinearity, and the signals are specifically generated via the inverse NFT operation to be fibre nonlinearity-tolerant.[9] Being a natural extension of the soliton-based eigenvalue communication,[12] the NFT optical transmission approach implies the use of the so-called nonlinear Fourier (NF) modes associated with the time-domain signal, as data carriers. These modes stay de-coupled along the evolution, propagating in an efficiently linear manner down the optical fibre in the presence of Kerr

Further author information:
A.V.: E-mail: a.vasylchenkova@ucl.ac.uk
J.E.P.: E-mail: y.prylepskiy1@aston.ac.uk

Nonlinear Optics and Applications XII, edited by Mario Bertolotti, Anatoly V. Zayats, Alexei M. Zheltikov, Proc. of SPIE Vol. 11770, 1177016 · © 2021 SPIE
CCC code: 0277-786X/21/$21 · doi: 10.1117/12.2588735

nonlinearity.[8,9] But similar to the DBP-type methods, the signal processing based on the NFT algorithms suffers from the large numerical complexity, power-sensitive numerical errors,[13] and complicated signal-dependent noise affecting our NF information carriers along their evolution.[10,13–15]

We note that up to the moment, in addition to a multitude of theoretical studies,[9,15] there exists a number of experimental demonstrations of NFT-based transmission systems,[16] including recently introduced efficacious b-modulation method,[17,18] that incorporates the continuous NF spectrum part. In fact, the latter approach provided the highest data rate for the NFT-based systems.[17,19] However, the common practice in the experiments on NFT-based transmission systems is to perform the off-line processing to (de)modulate the data from optical signals. Thus, in our study here we address the question of how to build up the hardware implementation of the NFT processing, aiming on-line processing.

Some nonlinearity mitigation techniques have been demonstrated in hardware, showcasing a finely tuned trade-off between accuracy and the number of operations on ASIC,[4–6] CMOS[20] and on silicon photonics devices.[21] At the same time, the study of practical implementation of NFT-based algorithms for real-time processing is still largely fragmentary, mainly because the numerical NFT operation accuracy is simultaneously affected by many factors, including the chosen NF modes modulation type, signal's power and sampling rate, to mention the most important. Noticeable efforts have been exerted towards implementing the soliton modes modulator on silicon photonic devices.[22–24] Yet the full transmission system requires also a demodulator part, performing a rather complicated operation of identifying either the solitonic eigenvalues at the receiver side. Also, the aforementioned approach does not give the improved data rate values, and requires significant further development. Overall, we mention that there is still a lack of evidence for the applicability of a general NFT algorithm for real-time optical data processing, and our paper contributes to the understanding of this issue.

In this work, for the first time, we demonstrate a hardware implementation of the fast direct NFT operation,[25] which is realised at Rx to map the optical signal to its NF spectrum (i.e. to demodulate the data). We mainly focus on the two largest challenges of the NFT algorithm hardware design. We investigate the accuracy penalty caused by a finite number of bits in the fixed-point number representation. On the base of this analysis, we identify the optimal bit width to use in the further design stage. Then, we emulate the algorithm functioning on FPGA using VHSIC Hardware Description Language (VHDL). We build the (meta)matrix multiplier, which is a key elementary component of the overall NFT algorithm. The emulation results provide us with information about performance and resource consumption.

2. FAST NFT PROCESSING

Signal propagation in the optical fibre under the assumption of ideally compensated loss is modelled by the nonlinear Schrodinger equation (NLS):[1]

$$i\frac{\partial q}{\partial z} - \frac{\beta_2}{2}\frac{\partial^2 q}{\partial t^2} + \gamma |q|^2 q = 0, \tag{1}$$

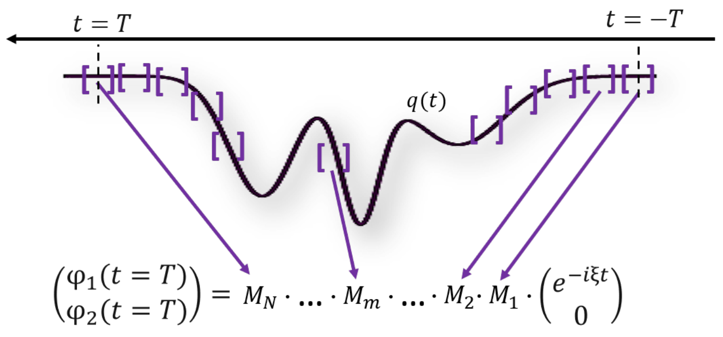
where $q(t,z)$ is a slow-varying envelope of the optical signal, z and t are distance along the fibre and retarded time the co-moving reference frame, respectively. Coefficients β_2 and γ are responsible for the effects of chromatic dispersion and Kerr nonlinearity. The correctness of the NLS for signal dynamics in the fibre is a main requirement for the NFT approach eligibility.

The direct NFT, i.e. the mapping between the signal in the optical domain and its NF spectrum, is performed by a solution of the so-called Zakharov-Shabat scattering problem,[26]

$$\begin{aligned} &\frac{d\varphi_1}{dt} = -i\xi\varphi_1(t) + q(t)\varphi_2(t), \qquad \frac{d\varphi_2}{dt} = -q^*(t)\varphi_1(t) + i\xi\varphi_2(t), \\ &\varphi_1(t\to -\infty) \to e^{-i\xi t}, \qquad \varphi_2(t \to -\infty) \to 0, \end{aligned} \tag{2}$$

where the signal to process, $q(t)$, acts as an effective scattering potential for two incident waves described by auxiliary functions $\varphi_{1,2}(t)$. Spectral parameter ξ acts as an analogous of frequency and metrics for the NF domain, where non-interacting NF components are defined. Auxiliary scattering coefficients are given as

$$a(\xi) = \lim_{t\to\infty} \varphi_1(t) e^{i\xi t}, \qquad b(\xi) = \lim_{t\to\infty} \varphi_2(t) e^{-i\xi t}, \tag{3}$$



$t = T$
$t = -T$
$q(t)$
$\begin{pmatrix}\varphi_1(t=T)\\ \varphi_2(t=T)\end{pmatrix} = M_N \cdot \ldots \cdot M_m \cdot \ldots \cdot M_2 \cdot M_1 \cdot \begin{pmatrix} e^{-i\xi t} \\ 0 \end{pmatrix}$

Figure 1. Principal scheme the transfer-matrix approach for solution of the Zakharov-Shabat system.

and dispersive NF modes correspond to the so-called continuous spectrum $r(\xi) = b(\xi)/a(\xi)$, while discrete (solitonic) spectrum is defined as scattering problem eigenvalues, i.e. zeros of scattering function $a(\xi)$ for $\Im\xi > 0$.

We note here that for communication purposes can be used both continuous spectrum $r(\xi)$, and scattering function $b(\xi)$. As $r(\xi)$ is a natural generalisation of the conventional Fourier spectrum, its modulation is known as the conventional NFT approach, or r-modulation. Recently introduced b-modulation is a more advanced technique, providing direct control over the resulting signal duration.[17] In this work, as a proof of concept, we do not consider solitonic components of the signal, assuming that only continuous ($r(\xi)$ or $b(\xi)$) spectral components are modulated at the Tx.

The numerical solution of the Zakharov-Shabat system is usually performed by the so-called transfer-matrix method.[9, 25] In this approach, after signal equidistant sampling on the reasonable support interval, $-T \leq t \leq T$, $q(t_m) = q_m$, each signal sample is used to build a corresponding trasfer matrix M_m. Each matrix effectively emulates elementary interaction between waves $\varphi_{1,2}(t)$ and potential $q(t)$ within a step, starting with the incident waves from Eq. (2) and resulting in their scattered versions (see Fig. 1 for the pictorial representation of the transfer-matrix method).

The numerically evaluated auxiliary scattering functions are therefore given as:

$$\begin{pmatrix} a(\xi) \\ b(\xi) \end{pmatrix} = \begin{pmatrix} e^{i\xi T} \\ e^{-i\xi T} \end{pmatrix} \cdot \prod_{m=1}^{N} M_m \cdot \begin{pmatrix} e^{i\xi T} \\ 0 \end{pmatrix}. \tag{4}$$

The widely used fast NFT algorithm is based on the fast Fourier transform (FFT) principles, which leads to the overall method's complexity $O(N \log_2^2 N)$ for the optical signal $q(t)$ sampled on N samples with Δt discretisation step.[25] In the main part of the algorithm, each entry of the transfer matrix is a polynomial of a variable $\zeta = i\exp(i\xi\Delta t)$. We refer to these objects from the hardware perspective as 'meta-matrices' in the following text. Each signal sample determines polynomials' coefficients for a meta-matrix `M` as:

$$M_m = \frac{1}{\sqrt{1+|q_m|^2\Delta t^2}} \frac{1}{\zeta} \begin{pmatrix} \zeta^2 & \Delta t q_m \zeta \\ -\Delta t q_m^* \zeta & 1 \end{pmatrix} \mapsto \hat{\mathtt{M}}[\mathtt{m}] = \frac{1}{\sqrt{1+|q_m|^2\Delta t^2}} \begin{pmatrix} [1,0,0] & [0, \mathtt{\Delta tq}[\mathtt{m}], 0] \\ [0, -\mathtt{\Delta tq}[\mathtt{m}]^*, 0] & [0,0,1] \end{pmatrix}, \tag{5}$$

here polynomial coefficients are ordered from the highest degree to a free term, i.e. $\zeta^2 - 2\zeta + 3 \mapsto$ `[1, -2, 3]`. We note here that the preceding factor $\frac{1}{\zeta}$ is multiplied trivially, so there is no need to introduce it in the matrix multiplication, and it has a minimal impact on the resulting computational complexity of the fast NFT algorithm.[25]

Then, all meta-matrices are multiplied pair-wise, each time increasing the degree of polynomials in matrix elements up to $2N$. In the final stage, the chirp-z transform is used for the simultaneous evaluation of the polynomials on the grid of nonlinear frequency ξ. The latter has a minor contribution to the total computational complexity of the NFT operation.

The main component of the algorithm is the meta-matrices multiplier,[25] so in this work, we emulate this unit on FPGA and use emulation results to estimate the required hardware resources. To design the best performing implementation in limited hardware resources, we perform preliminarily accuracy analysis to get an optimal bit width. This result significantly reduces the gap to the practical application of NFT-based optical transmission.

To analyse the device's performance on realistic examples, we generated a set of "received" waveforms, using the setup and signal parameters from one of the best-performing experiment on NFT transmission,[16] and from the most recent b-modulation NFT transmission system having the highest data rate.[18] Here we note that these approaches use different NF spectrum modulations: the so-called r-modulation and b-modulation.

3. COMMON HARDWARE PLATFORMS AND FIXED POINT ARITHMETIC ADVANTAGES

Currently, computationally heavy algorithms can be implemented using the FPGAs and specialised processor architectures, which include the computing units for common mathematical operations. The realisation of an algorithm using such blocks allows achieving a significant performance improvement.[27,28] It includes implementation of DSP methods for filtering, pre- and post-processing of optical signals.[7,29,30]

Modern FPGAs provide blocks that are optimised for fixed-point calculations, which include the multipliers of different bit width and configurable logic elements. Some FPGAs include the special blocks suitable for floating-point computations. However, at the moment there are no solutions that have a sufficiently large number of such blocks, required to synthesise an arbitrary complex algorithm[31] . Thus, on most FPGAs on the market, the implementation of floating-point algorithms is computationally and resource expensive in terms of the number of logic cells required (see e.g.,[32] for the schematics of the module for the addition of two IEEE754 floating-point numbers).

3.1 Fixed-point implementation with Python

Usually, the overflow of numbers during the calculations seriously reduces the accuracy of the algorithm. This phenomenon is one of the most undesirable in the fixed-point calculations since you can get a result that significantly differs from the floating-point counterpart. As a consequence, minor errors can remain in the hardware for a long time and become hardly detectable, especially for a considerable number of calculations.

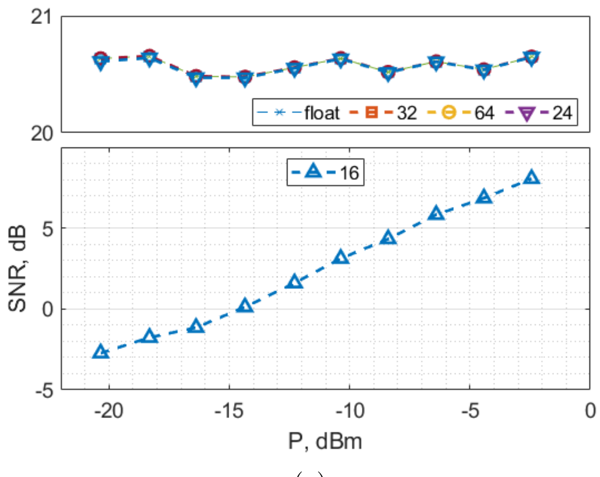
As an initial step of the implementation process of an algorithm for hardware platforms, we need a reference model that can be used to test and debug the fixed-point algorithm. The floating-point models are not suitable for being used as a reference since the comparison of numbers in different formats inevitably leads to errors. The overall error increases with the growth of operations number. Therefore, it is not possible to understand whether a given section of code contains an error or the difference in the results is caused by the quantization. So, as a part of the implementation, one has to convert the floating-point algorithm into a fixed-point form.

A fixed-point number can be represented with two parts: integer and fractional, so, commonly, the designers measure the dynamic range of the signal using a floating-point model (which does not suffer from overflowing), getting the required bit width of the integer part. The bit width of the fractional part is selected according to the desired accuracy and to the value of one bit in the form of a floating-point number. With this approach, one can find the bit width sufficient for computations.

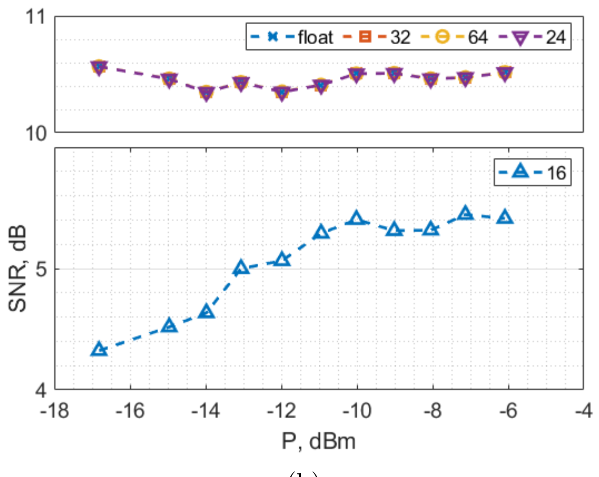
In this work, we use Python programming language is used for this initial testing stage. Python allows implementing the algorithm with functions that can be configured during the algorithm's runtime, so we can change the parameters without implementing several variants separately.

The key feature of Python in this context is the lack of a range for `Integer` data type. Thus, the overflow of the number is impossible, which allows modelling the algorithm with a different number of bits in the integer and fractional parts, and we can identify the sufficient number of bits in each part to achieve the required accuracy of the whole algorithm.

We implement the fixed-point variant of fast NFT[25] for different bit width on Python. Our goal is to compare the accuracy for the NF spectrum computation (the r spectral function for conventional NFT transmission,[16] and b spectral function for b-modulation approach[17]), where the spectra are found via the fixed-point realisation. The inverse NFT operation to generate the test signal uses the floating-point code. As a benchmark, we use the



(a)



(b)

Figure 2. (a) SNR vs signal power for different bit width of fixed-point, and the comparison with floating-point realization, for the conventional r-modulated system from.[16] (b) SNR vs signal power for different bit width of fixed-point, and the comparison with floating-point realisation, for the b-modulated system from.[17]

fast floating-point NFT, as there is no overflow to degrade the accuracy. As a measure of performance, we use the signal-to-noise ratio (SNR) between seed and back-to-back calculated NF spectra.

We compare the SNR for the range of signal powers, and for the range of bit width, see Fig. 2, to decide on the minimal sufficient bit width to build the optimal hardware implementation. It can be seen that for both testing examples 16 bit is not satisfactory, whilst stretching up from 32-bit width, we do not gain a significant change in terms of SNR.

4. MATRIX MULTIPLICATION EMULATION

We perform emulation of the principal component of the fast NFT algorithm: the multiplier of two meta-matrices. Fig. 3 provides explanations for the algorithm's architecture. Note that we have to deal here with rather untypical objects for FPGA implementation: operate with complex numbers, without a direct opportunity to split them into real and imaginary parts, and to process them in parallel. This feature naturally increases resource demand. Also, the meta-matrix third dimension (i.e. polynomial degree) increases in the consecutive algorithm layers, and so we cannot reuse the same hardware unit for all multiplications.

The most challenging part here is to find the balance between the performance and the FPGA resources required for the implementation. The fast NFT algorithm heavily relies on FFT-type processing, see Fig. 3. For this initial study, we used the embedded Xilinx FFT block,[28,33] focusing rather on higher-level algorithm optimizations.

Alternatively, one can reduce the number of FFT operations. Originally, each multiplication block converts the signal into the frequency domain and back (see Fig. 4). This scheme can be optimised if we calculate $4N$ high order FFTs at the beginning and 4 at the end of the entire multiplication cascade (factor 4 comes since the meta-matrices rank is $2{\times}2$), though for much longer processed sequences. Nevertheless, there is no guaranty that such an implementation would require fewer FPGA slices. Our estimations expect this approach will increase the overall resource consumption because of the considerable length of these FFTs, being the degree of resulting polynomials $2N$. However, we admit that in different algorithm's architecture such an FFT-domain approach can be beneficial.

At this point, we implement the basic meta-matrix multiplication block for Xilinx FPGA's using the VHDL language. We build the unit for a specific FFT length here, see Fig. 4, to scale them for an arbitrary value in further units realizations. In this example implementation, we use a 32-sample signal as a test study.

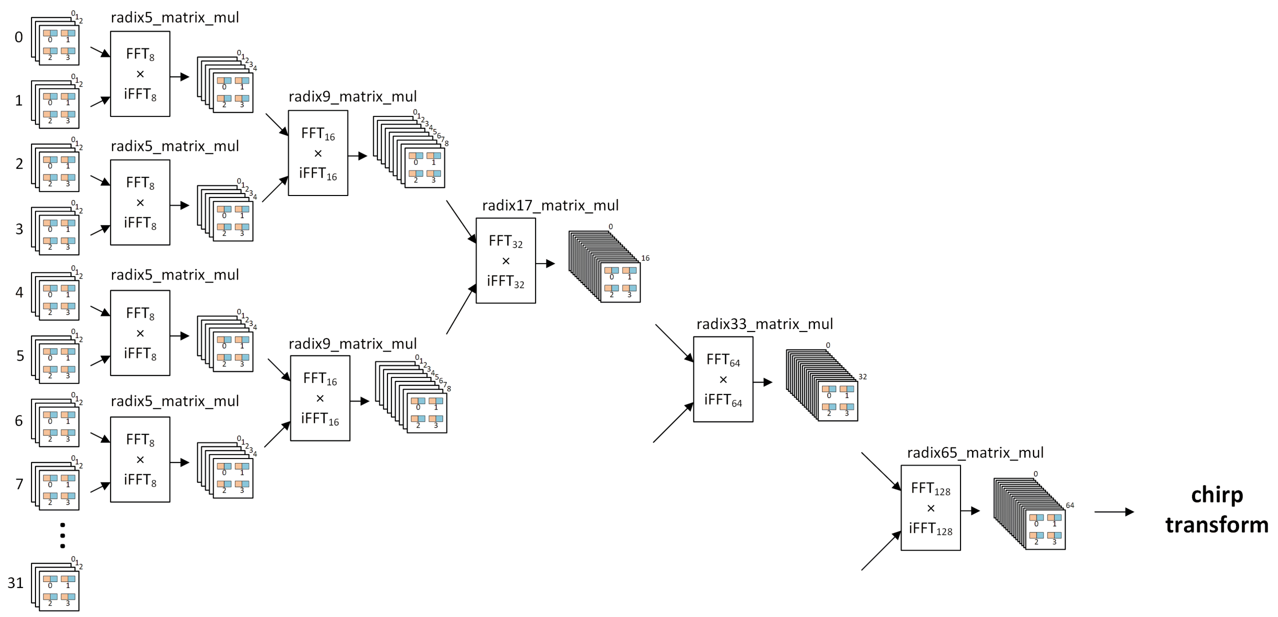


Figure 3. Principal scheme of the fast NFT algorithm, with designated meta-matrices and multipliers.

In Fig. 4, lengths of FFT blocks are motivated of the highest degree among resulting polynomials after the multiplication. At the first layer of matrix multiplication (designated as `radix5_matrix_mul` in Fig. 3) the highest polynomial degree is 2, so the maximum length of a meta-matrix third dimension is 3. After two meta-matrices multiplication, the respective parameters are 4 and 5. Because of the optimisation reasons, without loss of accuracy, the FFT block length should be a power of 2. Therefore, for the blocks, purposed the first-layer processing, the length of all FFT blocks is 8. The calculations of the FFT length of subsequent layers is similar, finally coming to the closest power of 2 from $2N + 1$ in the last layer.

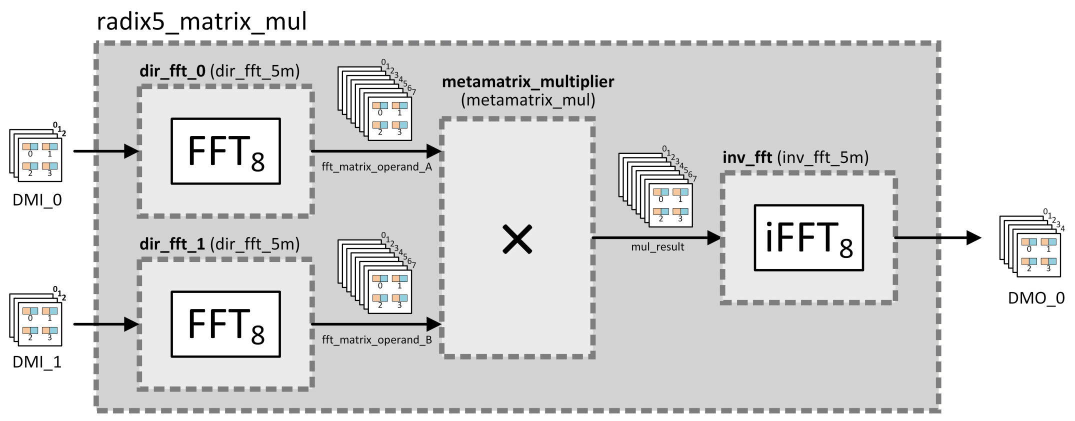


Figure 4. Scheme of two meta-matrices multiplier, featuring sub-sequential FFT operations and slice-wise multiplication.

There are different kind of resources (memory, input/output ports, etc.) which need to be considered during the architecture design process. As the largest part of the fast NFT extensively uses FFT operations of difference sequence length, the most demanded unit is the digital signal processing (DSP) block. After the consideration of the available commercial FPGAs, we use for our emulations one of the Xilinx Virtex 7 family devices, as this family is the most populated by DPS units.[27]

We also pay special attention to the complex-number arithmetics, as one of the non-trivial parts of fast NFT algorithms. The transfer-matrix method, and therefore, fast NFT algorithms, is not factorisable on real and

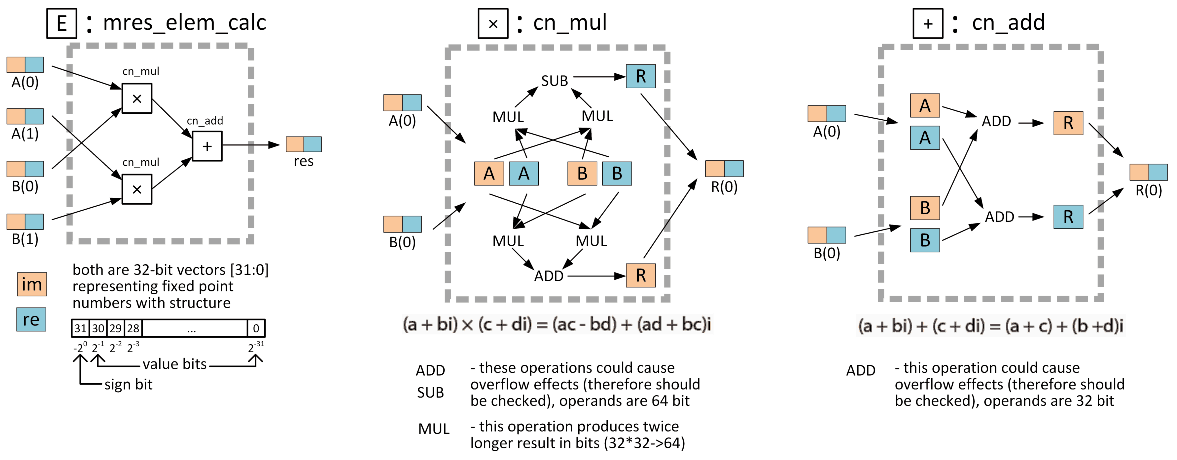

E : mres_elem_calc
A(0)
A(1)
B(0)
B(1)
cn_mul
cn_mul
cn_add
res
im
re
both are 32-bit vectors [31:0] representing fixed point numbers with structure
31 30 29 28 ... 0
-2^0 2^{-1} 2^{-2} 2^{-3} 2^{-31}
value bits
sign bit
× : cn_mul
SUB
MUL
MUL
R
A(0)
A A B B
B(0)
MUL
MUL
ADD
R
R(0)
(a + bi) × (c + di) = (ac - bd) + (ad + bc)i
ADD SUB - these operations could cause overflow effects (therefore should be checked), operands are 64 bit
MUL - this operation produces twice longer result in bits (32*32->64)
+ : cn_add
A(0)
A
A
ADD
R
B
B
ADD
R
B(0)
R(0)
(a + bi) + (c + di) = (a + c) + (b +d)i
ADD - this operation could cause overflow effects (therefore should be checked), operands are 32 bit

Figure 5. Schematics of complex numbers arithmetic operations blocks, with possible sources of overflow shown.

imaginary components. It is one of the consequences of the nonlinear property of NFT. As you can see in Fig. 5, fixed-point complex numbers multiplication is at risk of overflow. At some part of the design, it is required to save longer numbers (i.e., 64-bit numbers for the multiplication of two 32-bit ones).

We use the information of the emulated example for 32-sample signal, and then update it for the realistic length value in $N = 2^{12}$ samples. For the Virtex 7 FPGA, the emulated multiplier from Fig. 4, involved around 28% of the available DSP blocks, more noticeable using them in a multiplication of matrix elements (MxM part) rather than in FFTs. As we know that the FFT complexity for the sequence length k scales as $\log_2 k$, we extrapolate the DSP number and expect to use around 10 Xilinx Virtex 7 devices for the realisation of a full fast NFT algorithm. We also assume here that each identical meta-matrix multiplier block processes all layer input pairs in turn. This saves the on-board resources but requires longer operational time.

5. COMPUTATIONAL COMPLEXITY DISCUSSION

As can be seen from the presented estimate, the parallel implementation of the NTF on FPGAs requires a large number of hardware resources. Therefore, we need to find a reduction in resource consumption. Taking into account that the data appears at the input sequentially, it makes sense to consider changing the architecture of the delay line to reduce the number of required matrix multiplication blocks.

The architecture of the algorithms, considered in this paper, can be schematically given as in Fig. 6. Here delay line elements denoted as Z^{-1} and multiplication stage as sK for Kth layer of the multiplication.

For this 8-sample example, the overall multiplier consists of $\log_2 8 = 3$ layers. The number of multipliers decreases at each subsequent step. Such an approach indeed allows reducing the number of multipliers in comparison with sequential multiplication, resulting in the claimed complexity $O(N \log_2^2 N)$ in the contrast to $O(N^2)$. Nevertheless, we cannot insist that it is an optimal structure.

We suppose that it is possible to get rid of most multiplication blocks in the computing chain, reducing the required resources. The main trick here is to embed delay components between multiplication blocks, shaping the architecture in a more stream-like manner. It is a hardware-specific feature because of the sequential nature of inputs and limited memory allocation. Relocating the delay line blocks, the current architecture from Fig. 6, inherited from the floating-point fast NFT,[25] can be transformed to the structure given in Fig. 7. As a hardware-specific modification, it is suggested in this work in the context of NFT processing for the first time and therefore needed further investigation.

Estimating on the base of the example 8-sample signal, we lessen the number of required meta-matrix multiplication blocks from 7 to 3. Therefore, we expect a similar reduction in the number of FPGA logic cells required to implement a full-scale transform without a performance drop.

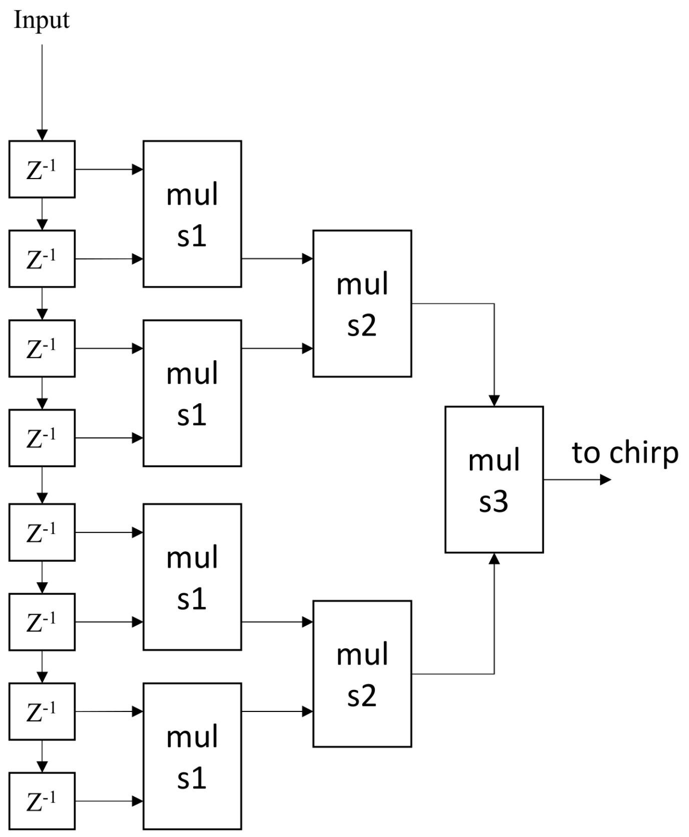


Figure 6. Current architecture with a delay line and parallel meta-matrices multipliers for 8 inputs.

6. CONCLUSIONS

In this work, we demonstrate the performance of the FPGA realisation of the fast NFT algorithm. We found the optimal bit width required for the fixed-point arithmetics. We also observed the significant demand in DSP blocks on the used board, which is caused by complex numbers and matrix operations, as well as FFTs. Nevertheless, for future work, it is possible to involve parallelisation, alternate the architecture and build the hardware performing the NFT operation, which should bring to the ability to process the optical signal with NFT methods on-line.

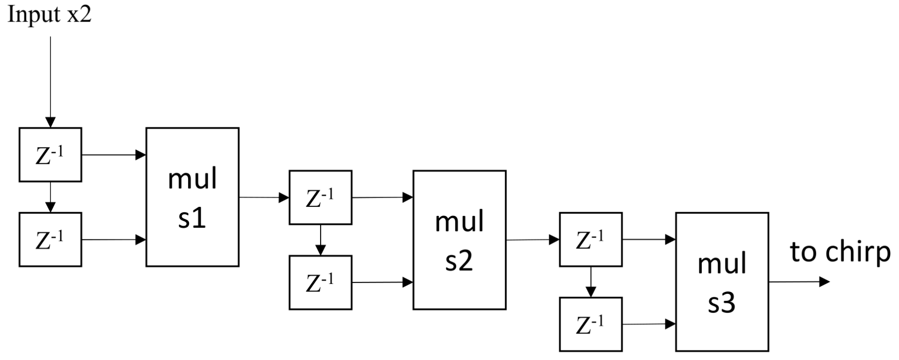


Figure 7. Alternative hardware-specific architecture with delay line and streaming meta-matrices multipliers for 8 inputs.

ACKNOWLEDGMENTS

The authors thank Erasmus+ mobility exchange programme. JEP acknowledges the Leverhulme Trust Research Programme Grant (RPG-2018-063). AV acknowledges the Leverhulme Trust Early Career Fellowship (ECF-2020-150).

REFERENCES

[1] Essiambre, R.-J., Gerhard K., Winzer, P. J., Foschini, G. J. and Goebel, B., "Capacity limits of optical fiber networks," Journal of Lightwave Technology 28(4), 662-701 (2010).

[2] Agrell, E., Karlsson M., Chraplyvy, A. R. *et al*, "Roadmap of optical communications," Journal of Optics, 18(6), 063002 (2016).

[3] Winzer, P. J., Neilson, D. T. and Chraplyvy, A. R., "Fiber-optic transmission and networking: the previous 20 and the next 20 years," Optics Express, 26(18), 24190-24239 (2018).

[4] Ranzini, S. M., Parahyba, V. E., Vilela, T. *et al*, "Digital back-propagation ASIC design for high-speed coherent optical system," Proc. 2015 SBMO/IEEE MTT-S International Microwave and Optoelectronics Conference (IMOC), 1-5 (2015).

[5] Fougstedt, C., Mazur, M., Svensson, L., Eliasson, H., Karlsson, M., and Larsson-Edefors, P., "Time-domain digital back propagation: Algorithm and finite-precision implementation aspects," Proc. Optical Fiber Communication Conference (OFC), W1G-4 (2017).

[6] Fougstedt, C., Svensson, L., Mazur, M., Karlsson, M., and Larsson-Edefors, P., "ASIC implementation of time-domain digital back propagation for coherent receivers," IEEE Photonics Technology Letters 30(13), 1179-1182 (2018).

[7] Giacoumidis, E., Lin, Y., Blott, M. and Barry, L. P., "Real-time machine learning based fiber-induced nonlinearity compensation in energy-efficient coherent optical networks," APL Photonics 5(4), 041301 (2020).

[8] Yousefi, M. I. and Kschischang, F. R., "Information Transmission Using the Nonlinear Fourier Transform, Parts I – III," IEEE Transactions on Information Theory 60, 4312 (2014).

[9] Turitsyn, S. K., Prilepsky, J. E., Le, S. T. *et al* "Nonlinear Fourier transform for optical data processing and transmission: advances and perspectives," Optica, 4(1), 307 (2017).

[10] Civelli, S., Forestieri, E. and Secondini, M., "Why noise and dispersion may seriously hamper nonlinear frequency-division multiplexing," IEEE Photonics Technology Letters 29(16), 1332-1335 (2017).

[11] Sherborne, T., Banks, B., Semrau, D., Killey, R. I., Bayvel, P. and Lavery, D., "On the impact of fixed point hardware for optical fiber nonlinearity compensation algorithms," J. Lightwave Technol. 36(20), 5016-5022 (2018).

[12] Hasegawa, A. and Nyu, T., "Eigenvalue communication," J. Lightwave Technol. 11, 395 (1993).

[13] Pankratova, M., Vasylchenkova, A., Derevyanko, S. A., Chichkov, N. B. and Prilepsky, J. E., "Signal-noise interaction in optical-fiber communication systems employing nonlinear frequency-division multiplexing," Physical Review Applied 13(5), 054021 (2020).

[14] Derevyanko, S. A., Prilepsky, J. E. and Turitsyn, S. K., "Capacity estimates for optical transmission based on the nonlinear Fourier transform," Nat. Comms. 7, 12710 (2016).

[15] Civelli, S., Forestieri, E. and Secondini, M., "Mitigating the Impact of Noise on Nonlinear Frequency Division Multiplexing," Applied Sciences 10(24), 9099 (2020)

[16] Le, S. T., Aref, V. and Bülow, H., "Nonlinear signal multiplexing for communication beyond the Kerr nonlinearity limit," Nat. Photon. 11, 570 (2017).

[17] Wahls, S., "Generation of time-limited signals in the nonlinear Fourier domain via b-modulation," Proc. European Conference on Optical Communications (ECOC), W3C-6 (2017).

[18] Yangzhang, X., Aref, V., Le, S. T. *et al*, "Dual-Polarization Nonlinear Frequency-Division Multiplexed Transmission With b-Modulation", J. Lightwave Technol. 37(6), 1570 (2019).

[19] Derevyanko, S. A., Balogun, M., Aluf, O., Shepelsky, D. and Prilepsky, J. E., "Channel model and the achievable information rates of the optical nonlinear frequency division-multiplexed systems employing continuous b-modulation," Optics Express 29 (5), 6384-6406 (2021).

[20] Parahyba, V. E. S., Reis, J. D., Ranzini, S. M. *et al*, "Performance against implementation of digital backpropagation for high-speed coherent optical systems," Electronics Letters 51(14), 1094-1096 (2015).

[21] Moralis-Pegios, M., Terzenidis, N., Mourgias-Alexandris, G. and Vyrsokinos, K., "Silicon photonics towards disaggregation of resources in data centers," Applied Sciences 8(1), 83 (2017).

[22] Moscoso-Martir, A., Koch, J., Müller, J., Sharif Azadeh, S., Pachnicke, S. and Witzens, J., "Silicon Photonics DWDM NLFT Soliton Transmitter Implementation and Link Budget Assessment," Proc. European Conference on Integrated Optics (ECIO), 1-4 (2020).

[23] Koch, J., Moscoso-Martir, A., Müller, J., Pachnicke, S. and Witzens, J., "Silicon Photonics DWDM NLFT Soliton Transmitter," Proc. ITG Conference "Photonic Networks", 1-8 (2020).

[24] Koch, J., Müller, J., Moscoso-Martir, A., Witzens, J. and Pachnicke, S., "DWDM Soliton Transmission enabled by Integrated Photonics and Nonlinear Fourier Transform," Proc. ITG Conference "Photonic Networks", 1-8 (2019).

[25] Wahls, S., Chimmalgi, S. and Prins, P. J., "FNFT: A Software Library for Computing Nonlinear Fourier Transform," J. of Open Source Software 3(23), 597 (2018).

[26] Shabat, A. and Zakharov, V., "Exact theory of two-dimensional self-focusing and one-dimensional self-modulation of waves in nonlinear media," Soviet physics JETP 34(1), 62 (1972).

[27] "Backgrounder. Stratix 10: The Most Powerful, Most Efficient FPGA for Signal Processing," Altera 2015, <https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/backgrounder/stratix10-floating-point-backgrounder.pdf> (5 May 2020).

[28] Chahardahcherik, A., Kavian, Y. S., Strobel, O. and Rejeb, R., "Implementing FFT algorithms on FPGA," J. of Computer Science and Network Security 11(11), 148 (2011).

[29] Watts, P., Waegemans, R., Glick, M., Bayvel, P. and Killey, R., "An FPGA-based optical transmitter design using real-time DSP for advanced signal formats and electronic predistortion,". J. of Lightw. Technol. 25(10), 3089-3099 (2007).

[30] Zhang, J., Xiao, X., Yu, J., Wey, J.S., Huang, X. and Ma, Z., "Real-time FPGA demonstration of PAM-4 burst-mode all-digital clock and data recovery for single wavelength 50G PON application," Proc. Optical Fiber Communication Conference (OFC), M1B-7 (2018).

[31] Hemmert, K. S. and Underwood, K. D., "An analysis of the double-precision floating-point FFT on FPGAs," Proc. 13th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'05), 171-180 (2005).

[32] Bekkers, J., "In-depth: IEEE 754 Multiplication And Addition," 6 December 2011, <http://jasperbekkers.nl/posts/floating-point/> (5 May 2020)

[33] Saeed, A., Elbably, M., Abdelfadeel, G. and Eladawy, M. I., "Efficient FPGA implementation of FFT/IFFT processor," International Journal of circuits, systems and signal processing 3(3), 103-110 (2009).