René Schmidt, Stephan Blokzyl, and Wolfram Hardt

A highly scalable FPGA implementation for cross-correlation with up-sampling support

Abstract: Cross-correlation is a fundamental algorithm which is used in many fields of digital signal processing. One important application is sound source localization. For this purpose, actual research shows a significant improvement of localization accuracy when using up-sampling-based algorithms. These state-of-the-art approaches are challenging for embedded realizations as up-sampling in combination with high sample rates and measurement window lengths leads to huge amounts of data to be processed and stored.

This chapter presents hardware acceleration techniques for cross-correlation with up-sampling support. The work introduces a novel processing architecture which addresses large-scale memory consumption compared to conventional solutions. The novel design introduces a highly scalable field programmable gate array realization with linear memory complexity and an interpolation factor-independent runtime.

Keywords: mathematics algorithms, digital signal processing, splines, field programmable gate arrays

1 Introduction

Cross-correlation algorithms are widely used in various fields of digital signal processing. Typical applications comprise image processing [1], audio signal processing [2], impedance spectroscopy [3] and sound source localization [2]. Especially for sound source localization, approaches like time difference of arrival (TDOA) are used, a technique to calculate the position of a sound source by computing the delay between incoming sound signals of spatially divided microphones [4]. The cross-correlation is used to calculate the audio signal lag (signal time difference). There are two common models for cross-correlation. The general definition is

$$u(t) \otimes v(t) = \int u^*(\tau)\, v(\tau + t) d\tau. \tag{1}$$

Since digital signals are discrete and real-valued, the definition (1) can be written

$$u(t) \otimes v(t) = \sum u(\tau) v(\tau + t). \tag{2}$$

René Schmidt, Stephan Blokzyl, Wolfram Hardt, Chemnitz University of Technology, Faculty of Computer Science, Professorship of Computer Engineering, Chemnitz, Germany, e-mail: rene.schmidt@informatik.tu-chemnitz.de

https://doi.org/10.1515/9783110558920-009

Alternatively, the cross-correlation can be calculated by using convolution, given by

$$u(t) * v(t) = \int u(\tau)v(\tau + t)d\tau. \tag{3}$$

An equivalent description is

$$u(t) * v(t) = \mathcal{F}^{-1}(\mathcal{F}(u(t)) * \mathcal{F}(v(t))). \tag{4}$$

The calculation (4) has recently been used in micro-controller-based realizations, since the fast Fourier transform (FFT) $\mathcal{F}$ can be implemented efficiently, in contrast to the traditional estimation (2), which has a runtime of $O(n\tau_{\max}) = O(n^2)$ [5]. FFT calculation is also appropriate for field programmable gate array (FPGA)-based solutions [6]. Hence, predesigned FFT IP cores are resource-intensive and restricted in the number of samples (compare Section 3).

Two major properties specify TDOA localization accuracy. One is the base distance between the microphones, which is proportional to the measurement precision. Small base distances cause high measurement uncertainty [7]. However, the base distance cannot always be adapted to a length which enables the required measurement accuracy. The second feature is the sampling rate. The measurement error reduces with a higher sampling rate, but sample rate incrementation leads to a higher data volume to be processed. With a measurement interval of 20 ms and a default microphone sample rate of 44.1 kHz, 882 values have to be correlated each measurement cycle. This amount of data is not a problem, but with increasing sample rate the computational effort raises significantly, which is challenging for embedded devices.

The sample rate can be increased by using faster ADCs or by generating additional artificial data with up-sampling. Kan has presented an up-sampling method to improve localization accuracy [2]. The challenge can be illustrated with the following example. An up-sampling rate of 64 together with a sample rate of 44.1 kHz leads to an increased sample rate of approximately 2.8 MHz and 56,448 data samples to be computed. The restriction of, e. g., the Xilinx reference FFT IP core to 65,536 data samples [8] allows a maximum sample rate of approximately 3.3 MHz. Higher sample rates cannot be realized without measurement window reduction.

This chapter is structured as follows. Section 2 introduces related work and applications and validates state-of-the-art cross-correlation methodologies with respect to embedded realizations. Section 3 gives a review of an FPGA-based reference design that is based on common cross-correlation as presented in standard literature. Section 4 presents a novel architecture for a highly scalable FPGA implementation of up-sampling supporting cross-correlation. Finally, Sections 5 and 6 conclude the chapter with the results and benefits of the proposed solution.

2 Related work

There is a considerable amount of literature discussing versatile applications of cross-correlation. The proposed approaches apply software-based cross-correlation specified and optimized for different measurement applications. Venable presented a work in 2006 using cross-correlation for the estimation of peptide molecular weights from tandem mass spectra [9]. With the tool SEQUEST, a cross-correlation-based template matching approach is used for spectrogram comparison [10].

In the field of image exploitation, the fast normalized cross-correlation has been established as novel template matching (TM) method [1, 11]. The TM-based cross-correlation has been used for medical applications like motion recognition in blood flow imaging [12] and frontotemporal dementia [13] and for quality assurance in industrial production processes like defect detection of electronic devices [14]. Other applications use cross-correlation to calculate time differences between two related signals, like sound source localization methods [4], as mentioned above.

The listed algorithms are realized in software and cause disadvantageous high computational costs [5], which is challenging in embedded applications. Therefore, FPGA technology is applied to accelerate computation with the help of parallel processing strategies. FPGA-based cross-correlation architectures, as proposed in the literature, can be classified into two main categories: solutions using discrete cross-correlation and methods applying FFT. Approaches from the first category solve equation (2) and implement a parallel or serial architecture [15] as depicted in Fig. 1. These architectures are used in the field of orthogonal frequency division multiplexing [16] and other high-performance applications [17]. The serial structure provides a high processing speed, since each sample X needs to be a processed one. However, resource consumption rises linearly with the amount of samples per measurement window. As each measurement sample requires a single multiplier for cross-correlation, e. g., 20 samples consume 20 and 20,000 samples consume 20,000 multipliers. The relation between the measurement sample and multipliers is linear. With 220 DSP cores for multiplication, as provided by state-of-the-art FPGAs,[1] the proposed implementation is not feasible for measurement scenarios with high sample rates or large-scale measurement windows.

In contrast to the serial approach, the parallel architecture saves resources, since the calculation needs to be executed for each possible lag [15]. However, the parallel architecture consumes a big amount of multipliers to realize cross-correlation for high sample rates. As the count of multipliers is limited in FPGAs, both serial and parallel solutions are not feasible for high sample rates and large measurement windows. Thus, state-of-the-art FPGA-based solutions [6, 18] use FFT-based cross-correlation realization with convolution (see equation (4)). Fig. 2 shows the appropriate hardware

1 E. g., Xilinx Zynq®-7000 All Programmable SoC XC7Z020-CLG484.

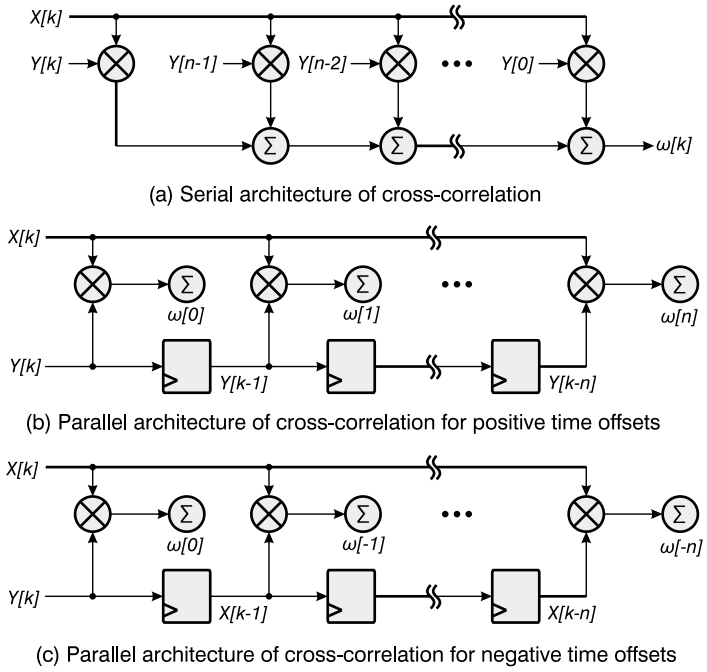


Fig. 1: Comparison of common FPGA architectures for direct calculation of discrete cross-correlation. (a) The stepwise, serial realization of cross-correlation. All X-samples have to be processed for the final cross-correlation result. (b) and (c) The highly parallel architecture for cross-correlation which provides a result update every clock cycle.

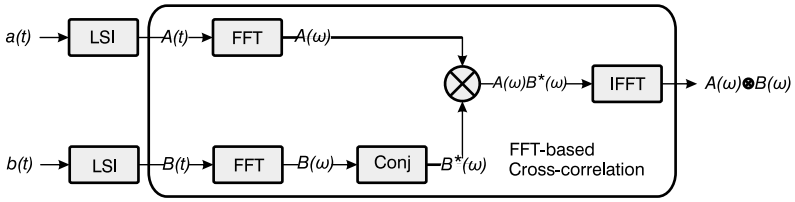


Fig. 2: FFT-based cross-correlation architecture with linear spline interpolation-based preprocessing.

implementation applying parallel FFT calculation. Various IP cores for efficient, high-performance FFT computation are available, easy to use, and optimized for specific FPGA devices, which makes them comfortable for modern FPGA designs.

A literature survey and manifold application scenarios prove the cross-correlation is a potential method for a wide range of commercial, industrial and research fields. The high computational effort and significant computational resource utilization represent the major drawbacks of state-of-the-art methodologies, especially for embedded realizations. The following analysis discusses standard FPGA-based cross-

correlation in detail and derives an approach facing specific resource limitations of FPGA technology while providing higher computational performance compared to conventional cross-correlation approaches.

3 Reference design

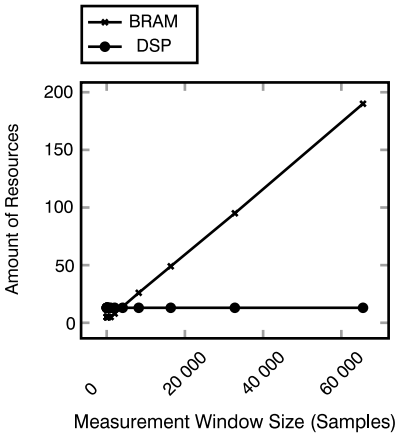
The FPGA-based hardware-accelerated design for cross-correlation, depicted in Fig. 2, has been implemented on an evaluation board with Xilinx Zynq®-7000 SoC to compare and assess different realization alternatives and implementation techniques. Two parallel FFT modules simultaneously transform two time domain input signals $A(t)$ and $B(t)$ from spatially separated microphones into a frequency domain. The design substitutes time domain cross-correlation with multiplication in the frequency domain of signal A and the complex conjugated of signal B.

To support the up-sampling methodology by Kan, which improves the measurement accuracy (compare Section 1), both signal $a(t)$ and signal $b(t)$ have to be interpolated to $A(t)$ and $B(t)$ prior to FFT. A common interpolation method is, e. g., linear spline interpolation (LSI). The inverse fast Fourier transform (IFFT) converts the multiplication result back to the time domain, which represents the cross-correlation result. The most prominent peak of the cross-correlation result characterizes the time offset between signal A and signal B.

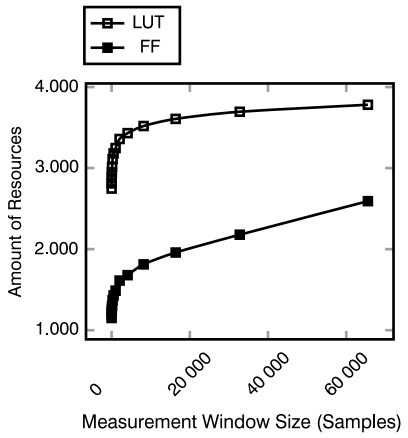
The two FFTs have been implemented as dual-channel FFT, which provides the best resource optimization provided by the vendor Xilinx. Fig. 3 (a) shows the appropriate utilization of Block RAM (BRAM) and DSP resources in relation to the measurement window length. The number of DSP slices is constant and independent of the measurement window size. The applied Radix-2 FFT algorithm [19] has a fixed structure and does not change with measurement window variation. The BRAM count increases linearly and is proportional to the measurement window length. The BRAM components are used as data buffer only and not applied for calculation. The memory increases by expanding the measurement window length and vice versa.

The Radix-2 algorithm uses a fixed amount of flip-flops (FFs) and lookup tables (LUTs) for FFT. Hence, measurement window length variation results in a logarithmic relation of logic resource consumption (see Fig. 3 (b)). A larger measurement window size requires a larger address space and additional glue logic, but the growth rate for buffer control logic consumption reduces with the increasing buffer size.

As a result, the processing time increases linearly with the measurement window size (see Fig. 4). With a logic utilization less than 10 % on a Zedboard with Zynq®-7000 All Programmable SoC XC7Z020-CLG484, the method is very resource saving. The amount of FFs and LUTs is logarithmically related to the number of samples per measurement window to be processed. However, the amount of used memory (BRAM) is very high and the reference cross-correlation architecture cannot be realized on small,



(a) Linear dependency of BRAM utilization as a function of the sample count in the measurement window. The amount of applied DSP cores is independent of the amount of samples in the measurement window.



(b) Logarithmic dependency of both FF and LUT utilization as a function of the amount of samples per measurement window.

Fig. 3: Logic resources utilization for the reference design depicted in Fig. 2 on a Zedboard with Xilinx Zynq®-7000 All Programmable SoC XC7Z020-CLG484.

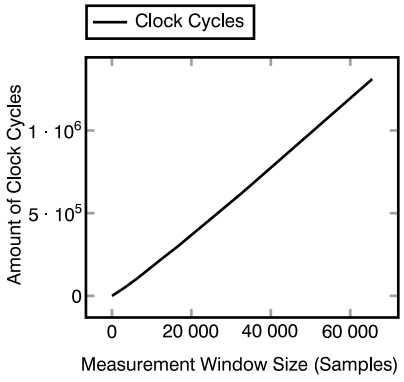


Fig. 4: Linearly rising FPGA processing time for the reference cross-correlation realization depicted in Fig. 2 as a function of the measurement window size.

low-budges FPGA devices with very limited BRAM resources. As the cross-correlation is just a single part of the overall solution, it shall not occupy the majority of BRAM resources of the host device.

Especially the storage of additional data caused by linear spline interpolation consumes a lot of BRAM resources. As the interpolation data are artificial and do not represent real measurements, a strategy to prevent the buffering of interpolation data will improve the BRAM utilization. While the reference FFT IP core buffers internally all data to process, the availability of interpolation data is essential for the convolution approach. Furthermore, the FFT core uses a measurement window size with a power of

two step width only. If fewer data are processed the remaining window share is filled with zeros, which causes unnecessary computational effort. The maximum lag can be determined for most of the TDOA use cases. Hence, it is not necessary to calculate the complete cross-correlation, which saves calculation time. The hereinafter highly scalable FPGA implementation addresses these disadvantages and faces the trade-off between resource limitation and measurement accuracy. The method uses fixed-point arithmetic and provides high up-sampling capabilities with linear spline calculation.

4 Novel acceleration architecture

The novel acceleration architecture for FPGA-based cross-correlation consists of 14 different components (see Fig. 5). The first two parts are two True Dual Port Block-RAMs (BRAM X, BRAM Y) which buffer sampled input data that have to be processed. The first port is assigned for filling the BRAM and the second port provides access to the connected processing modules. The subsequent component is the data access (DA) block. This module controls input data readout from the BRAMs and handles interfaces and address mapping.

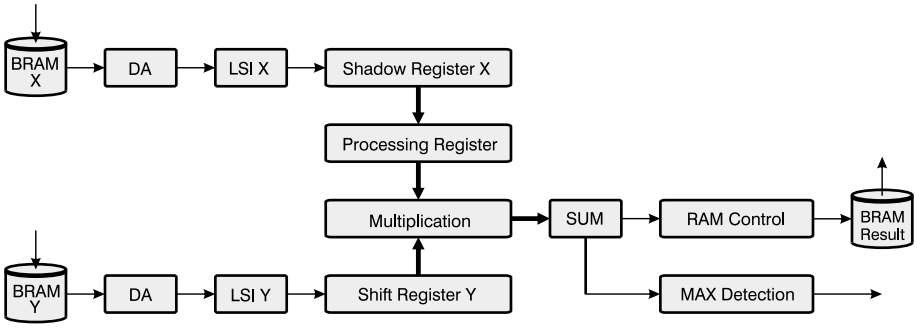


Fig. 5: Novel acceleration architecture for FPGA-based cross-correlation.

Next, the LSI block stores two subsequent values and calculates interpolation values depending on the interpolation factor ξ. The LSI block for x-values feeds a shadow register, which is organized as a shift register with a length of ξ. The interpolation block of the y-values feeds a second shift register with equal length ξ. The initialization value of LSI Y is for each interpolation step $X_n - \tau_{\max}$ and generates $k = 2\tau_{\max} + 1$ values, while the next ξ x-values are calculated simultaneously; $\tau_{\max}$ describes the maximum delay (in clock cycles) of the cross-correlation given by $\tau_{\max} = f(F_s, \xi)$ with sample rate F_s. After each processing step of all k values, the shadow register X is copied to the processing register. The ξ interpolation values in shift register Y are multiplied in parallel with all ξ interpolation values in the processing register, which results in the

products $\omega_i(\tau)$ with $\{i \in \mathbb{N} \mid 0 \leq i < \xi\}$. All multiplication results $\omega_i(\tau)$ are finally summed up to $\Omega(\tau)$ in the SUM module.

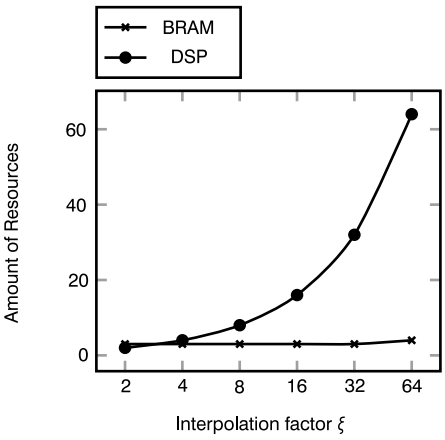
In reference to equation (2), the interim result from the previous processing step for X_{n-1} is prepared by the BRAM control block, transferred to the SUM block, and accumulated in the processing step of X_n. The BRAM control block stores the updated X_n in BRAM result and handles the storage access. The maximum detection index and the maximum detection value are stored by the MAX detection block after each summation step. The procedure is repeated for all $(N \cdot \xi)$ x-values, where N is the amount of data per window. Since the multiplication generates big numbers and the range limit of integers is reached soon, the whole processing is done with fixed-point arithmetic. The width of the input data is defined as 32 bits, with 16 bits for the integer part and 16 bits for the fractional part.

5 Results

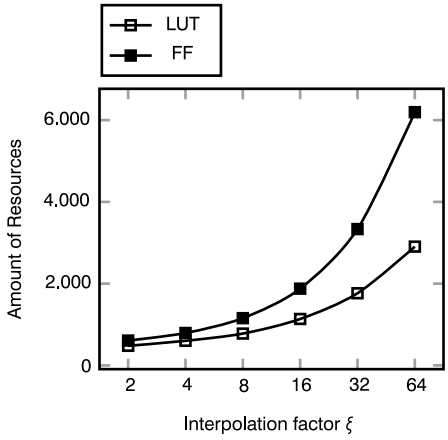
The algorithm calculates for each x-value ξ interpolation values and processes them in parallel. The computation is done $2\tau_{\max} + 1$ times, which leads to a runtime of $O(2N\tau_{\max} + 1)$, depending on the according sample rate. The runtime O is independent of the interpolation factor ξ since all interpolation values are calculated at the same point of time.

Logic resource consumption of the proposed design depends on the value of interpolation factor ξ. FF, LUT, DSP slice and BRAM utilization is proportional to ξ with linear hardware complexity as shown in Fig. 6. Furthermore, both FF and LUT consumption are affected by the measurement window size and grow logarithmically with expanding window size and vice versa. The number of required DSP slices is independent of the measurement window length as all interpolation steps are executed in parallel.

The BRAM utilization is linear and strongly linked to the runtime and changes with sample rate, measurement windows size and interpolation factor modifications (see Fig. 7). Window size and sample rate changes affect the two data input buffers (BRAM X/Y) and the BRAM for results, while a modification of ξ affects the result BRAM only. The growth of BRAM consumption is moderate as existing memory utilization rises up to 100 % before a new memory block will be activated. The proposed architecture uses significantly less BRAM resources compared to the FFT IP core-based reference design. E. g., a hardware realization with an interpolation factor of $\xi = 64$ requires only ˜60 % more FFs, ˜10 % more LUTs, and five times more DSP slices, but saves more than 97 % of BRAM resources for 65k samples (maximum of reference design). With 18 FFs (0.3 %) and 60 LUTs (2 %) more, the proposed design is able to process 262k samples (equal number of DSP slices and four times the BRAM consumption), which equals four times the performance compared to the reference design. The

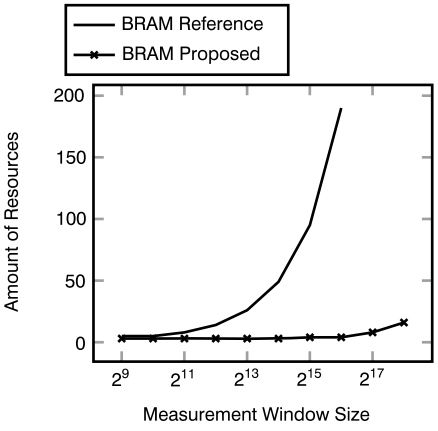


(a) Linear relationship between interpolation factor ξ and DSP slice consumption. BRAM utilization is independent of ξ.

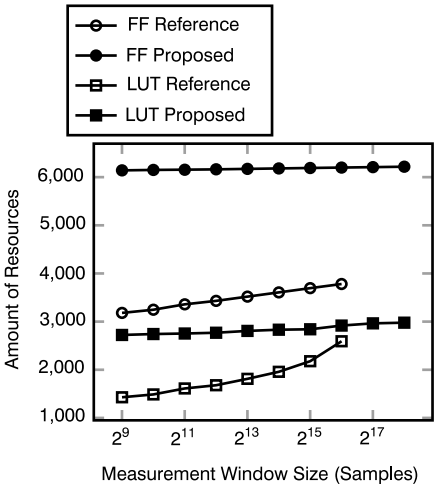


(b) Linear relationship between interpolation factor ξ and required FFs and LUTs.

Fig. 6: Influence of the interpolation factor ξ on resource utilization of the proposed acceleration architecture for FPGA-based cross-correlation.



(a) Comparison of BRAM utilization between the reference design and the proposed design.



(b) Comparison of FF and LUT utilization between the reference design and the proposed design.

Fig. 7: Comparison of resource consumption of the reference and proposed designs for interpolation factor $\xi = 64$.

proposed design does not store intermediate multiplication results, which leads to a significant reduction of BRAM resource consumption compared to the reference design. Furthermore, the measurement window length and sample rate do not affect the overall processing part of the design. FF and LUT consumption is nearly constant and rises minimally for glue logic only, which is negligible compared to the available FF and LUT resources on modern FPGA devices. The independence of logic resource consumption from different measurement configurations leads to a beneficial scalability of the overall architecture. Additionally, the novel method provides a measurement window configuration of arbitrary size, which is only limited by the available logic resources of the applied FPGA.

6 Conclusion and future work

The proposed FPGA design provides a highly scalable FPGA architecture for hardware-accelerated cross-correlation with up-sampling support. The novel approach has a runtime of $O(2N\tau_{\max})$ and a memory complexity of $O(2N+3\xi+2\tau_{\max})$. The up-sampling is realized by LSI and characterized by an interpolation factor ξ. The modification of the interpolation depth affects the consumption of DSP slices and FFs only and does not influence the overall processing time, which is solely coupled to window size and sample rate.

The novel approach enables arbitrary and unlimited measurement window lengths, which allows an extension of the maximum measurement window size compared to the reference design. The proposed solution requires significantly less BRAM and LUT resources for both high sample rates and large-scale measurement window configurations. The presented architecture occupies slightly more FFs, which is negligible with respect to the available FFs on modern FPGA devices.

Future activities focus on the analysis of time restrictions to ensure real-time capabilities of the design. Furthermore, techniques for a significant reduction of DSP slices have to be developed.

Bibliography

[1] J. Lewis, “Fast normalized cross-correlation”, Vision Interface, vol. 10, pp. 120–123, 1995.

[2] Y. Kan, P. Wang, F. Zha, et al., “Passive acoustic source localization at a low sampling rate based on a five-element cross microphone array,” Sensors, vol. 15, no. 6, pp. 13326–13347, 2015.

[3] S. Gawad, T. Sun, N. G. Green, et al., “Impedance spectroscopy using maximum length sequences: application to single cell analysis”, Review of Scientific Instruments, vol. 78, no. 5, 054301, 2007.

[4] C. H. Knapp and G. C. Carter, "The generalized correlation method for estimation of time delay", IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. 24, no. 4, pp. 320–327, 1976.
[5] D. Lyon, "The discrete fourier transform, part 6: cross-correlation.", Journal of Object Technology, vol. 9, no. 2, pp. 17–22, 2010.
[6] D. Nguyen, P. Aarabi, and A. Sheikholeslami, "Real-time sound localization using field-programmable gate arrays", In: Acoustics, Speech, and Signal Processing, 2003. Proceedings. (ICASSP'03). 2003 IEEE International Conference on, IEEE, vol. 2, 2003, pp. II–573.
[7] R. Schmidt and W. Hardt, "Schallquellenlokalisierung zur sprecheridentifizierung fur multi-user dialogschnittstellen", In: Studierendensymposium Informatik 2016 der TU Chemnitz, TU Chemnitz, vol. 1, 2016, pp. 105110.
[8] L. I. P. Guide, "Fast fourier transform v9. 0".
[9] J. D. Venable, T. Xu, D. Cociorva, et al., "Cross-correlation algorithm for calculation of peptide molecular weight from tandem mass spectra", Analytical Chemistry, vol. 78, no. 6, pp. 1921–1929, 2006.
[10] J. K. Eng, B. Fischer, J. Grossmann, et al., "A fast sequest cross correlation algorithm", Journal of Proteome Research, vol. 7, no. 10, pp. 4598–4602, 2008.
[11] J.-C. Yoo and T. H. Han, "Fast normalized cross-correlation", Circuits, Systems, and Signal Processing, vol. 28, no. 6, pp. 819–843, 2009.
[12] J. Luo and E. E. Konofagou, "A fast normalized cross-correlation calculation method for motion estimation", IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control, vol. 57, no. 6, pp. 1347–1357, 2010.
[13] B. B. Avants, C. L. Epstein, M. Grossman, et al., "Symmetric diffeomorphic image registration with cross-correlation: evaluating automated labeling of elderly and neurodegenerative brain", Medical Image Analysis, vol. 12, no. 1, pp. 26–41, 2008.
[14] D.-M. Tsai and C.-T. Lin, "Fast normalized cross correlation for defect detection", Pattern Recognition Letters, vol. 24, no. 15, pp. 2625–2631, 2003.
[15] A. Fort, J.-W. Weijers, V. Derudder, et al., "A performance and complexity comparison of auto-correlation and cross-correlation for ofdm burst synchronization", In: Acoustics, Speech, and Signal Processing, 2003. Proceedings. (ICASSP'03). 2003 IEEE International Conference on, IEEE, vol. 2, 2003, pp. II–341.
[16] M. Chen, J. He, and L. Chen, "Real-time optical ofdm long-reach pon system over 100 km ssmf using a directly modulated dfb laser", Journal of Optical Communications and Networking, vol. 6, no. 1, pp. 18–25, 2014.
[17] B. Von Herzen, "Signal processing at 250 mHz using high-performance fpga's", IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 6, no. 2, pp. 238–246, 1998.
[18] J. Pang and J. Starzyk, "Fast direct gps signal acquisition using fpga", In: ECCTD, Krakow:[sn], vol. 287, 2003.
[19] S. Goedecker, "Fast radix 2, 3, 4, and 5 kernels for fast fourier transformations on computers with overlapping multiply-add instructions", SIAM Journal on Scientific Computing, vol. 18, no. 6, pp. 1605–1611, 1997.