Contents lists available at ScienceDirect

Digital Signal Processing

www.elsevier.com/locate/dsp

A structured review of sparse fast Fourier transform algorithms

Elias Rajaby [*], Sayed Masoud Sayedi

Department of Electrical and Computer Engineering, Isfahan University of Technology, Isfahan 84156-83111, Iran

ARTICLE INFO

Article history:
Available online 19 January 2022

Keywords:
Discrete Fourier transforms
Sparse signals
Sparse fast Fourier transform
Big data

ABSTRACT

Discrete Fourier transform (DFT) implementation requires high computational resources and time; a computational complexity of order $O(N^2)$ for a signal of size N. Fast Fourier transform (FFT) algorithm, that uses butterfly structures, has a computational complexity of $O(Nlog(N))$, a value much less than $O(N^2)$. However, in recent years by introducing big data in many applications, FFT calculations still impose serious challenges in terms of computational complexity, time requirement, and energy consumption. Involved data in many of these applications are sparse in the spectral domain. For these data by using Sparse Fast Fourier Transform (SFFT) algorithms with a sub-linear computational and sampling complexity, the problem of computational complexity of Fourier transform has been reduced substantially. Different algorithms and hardware implementations have been introduced and developed for SFFT calculations. This paper presents a categorized review of SFFT, highlights the differences of its various algorithms and implementations, and also reviews the current use of SFFT in different applications.

1. Introduction

Discrete Fourier transform (DFT) and acquisition of signals' frequency components is one of the most important and widely used operations in digital signal processing. Performing conventional DFT algorithms for a signal of size N needs $O(N^2)$ operations. By introducing Fast Fourier Transform (FFT) and using butterfly structures to eliminate duplicate mathematical operations, computational complexity was reduced to $O(Nlog(N))$. Due to its much lower complexity, FFT has become a common method in most DFT computations, and it has encouraged and resulted in many works to implement the algorithm on the hardware [1] and [2].

In recent years, by introducing big data in many applications (for example [3], [4]), using FFT algorithm for such data still imposes much computational complexity, processing time, and energy use. To mitigate the problem many efforts have been done to implement the algorithm more efficiently by customizing it for different applications and also by using different hardware platforms [5–7].

In some applications most of the Fourier coefficients of the involved signals are zeroes or negligible, in other words, in frequency domain signals' values are sparse [8–11]. Based on this characteristic, the Sparse Fast Fourier Transform (SFFT) algorithm, with low computational complexity and low processing time, has been introduced for DFT calculations of such signals. The method can be used in both cases of having exact K sparse signals where only at most $K(<<N)$ non-zero frequency coefficients exist, and general sparse signals where only at most K significant coefficients exist and other coefficients are zero or very small. A typical value of K for N=2^{22} is smaller than 2^{10} for such signals [12] and [13]. SFFT is used in a variety of applications such as global positioning [14], wideband spectrum sensing [15], radar systems [13], medical spectroscopy [16], computational photography [17], and speech processing [18].

SFFT significantly improves the processing speed of Fourier transform of big data by calculating only a small portion of frequency values, which results in sub-linear time complexity. Sublinear-time Fourier Transform has been widely investigated in the past three decades and various algorithms with different applied conditions have been proposed for its implementation. For example, the algorithm presented in [19] has a complexity of $O(K\log^a(N)\ \ log(N/K))$, in which parameter $'a'$ is a number greater than 2. Following this work, some attempts have been made to reduce the complexity more. SFFT algorithm was first proposed in [20] with a complexity of $O(log(N)\sqrt{KNlog(N)})$, and after that, many more SFFT algorithms with some modifications and improvements were proposed. The algorithms, based on their calculation approaches, can be grouped into two different categories; deterministic algorithms and non-deterministic algorithms. Deterministic algorithms converge to the correct answer and usually do not need many parameter settings, but have a high computational load. Non-deterministic algorithms converge to the correct

[*] Corresponding author.
E-mail address: e.rajaby@ec.iut.ac.ir (E. Rajaby).

https://doi.org/10.1016/j.dsp.2022.103403
1051-2004/© 2022 Elsevier Inc. All rights reserved.

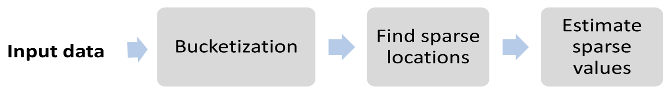


Fig. 1. Three main building blocks of SFFT algorithms.

answer with a high probability and need more parameter settings, but with a less computational load.

The main aim of this paper is to review different SFFT algorithms and highlight their differences. Considering the previous review presented in [12], here recent works are also reviewed and to show more clearly the differences, the algorithms are categorized based on their processing approaches. The rest of the paper is organized as follows. The basic theory of SFFT algorithms is presented in section 2. In section 3 different SFFT algorithms, are introduced and compared. In section 4 some important applications of SFFT are reviewed. And finally, section 5 concludes the paper. In Appendix, a detailed numerical example of basic SFFT is provided.

2. Basics of SFFT algorithms

Discrete Fourier transform of $\boldsymbol{x}$, a vector of size N, is $\hat{\boldsymbol{X}}$, another vector of size N, expressed as:

$$\hat{X}_l = \sum_{i=0}^{N-1} e^{\frac{-2\pi lij}{N}} x_i, \quad l \epsilon \{0, 1, \ldots, N-1\}, \quad j = \sqrt{-1} \tag{1}$$

If $\hat{\boldsymbol{X}}$ is sparse and has only K $(<< N)$ significant elements, then Sparse Fast Fourier Transform (SFFT) algorithm can be used for its calculation. SFFT algorithm by using the sparsity characteristic of the signal makes the required number of data samples much less than N and calculates $\hat{\boldsymbol{X}}$ with much less computational complexity compared to that of FFT algorithm.

In general, as shown in Fig. 1 performing SFFT algorithm has three main steps of bucketization, finding sparse locations, and estimation of sparse values. The steps are as follows:

2.1. Bucketization

In the bucketization step input signal is downsampled in such a way that each Fourier value of the resulting signal, which is called bucketized signal, is the sum of several Fourier values of the input signal (through an aliasing process). The aim is to have buckets with at most one significant frequency in each bucket. Two main methods of bucketization are:

1) Adjacent Fourier values are summed up; as shown in Fig. 2-a. If needed, the signal spectrum has been uniformly rearranged beforehand. This method is known as indirect sub-sampling in the frequency domain.
2) Periodic Fourier values are summed up; as shown in Fig. 2-b. This method is known as the direct sub-sampling in the time or spatial domain.

2.2. Detection of sparse locations

At this step by analyzing the frequencies of each bucket, the locations of sparse frequencies are detected. Two main methods of detection are:

1) Detection based on bucket Phase: According to the shift property of Fourier transform if a signal is shifted by a specific value in the spatial domain, its phase in the corresponding bucket's frequency domain is also changed by a specific value as expressed in (2). By assumption that at most one significant sparse frequency exists in each bucket, the phase change information can be used to detect locations of significant frequencies.

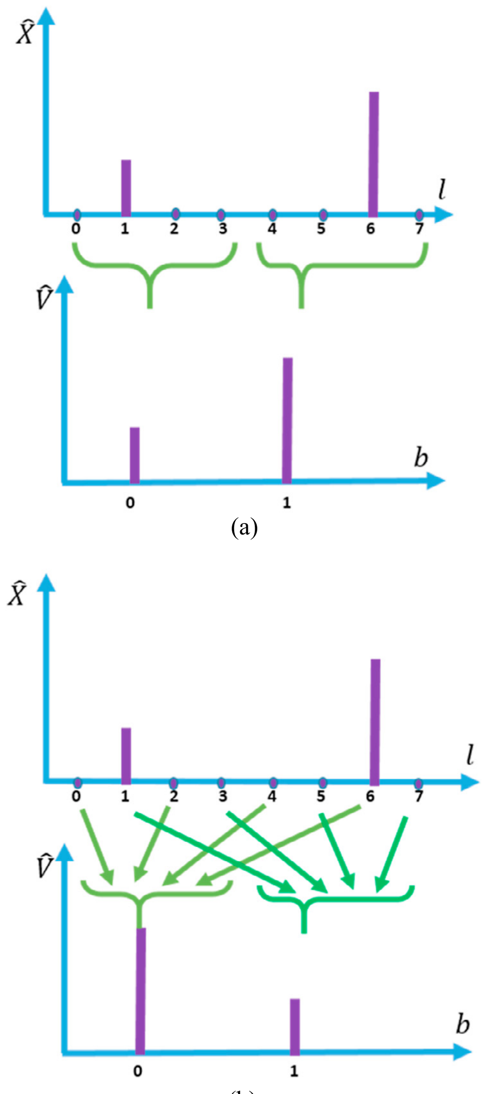
$$x_1(n) = x(n-m) \xrightarrow{DFT} \hat{X}_1(l) = e^{\frac{-j2\pi ml}{N}} \hat{X}(l) \tag{2}$$

2) Detection based on bucket number: In this method, by using an iteration procedure all frequencies located in a bucket will receive a vote. At the end, all frequencies with high votes are selected as significant frequencies. The set of frequencies located in each bucket is determined from the previous bucketization step. For example for the direct sub-sampling method, the set of frequencies in the bth bucket is given by:

$$F_b = \{f \mid f = \frac{N}{D} * i + b, i = 0, 1, \ldots, D-1\} \tag{3}$$

where D is the sub-sampling rate.

Above two methods and also two other methods, namely, the syndrome decoding method and pruning method are explained in more detail in section 3.



(b)

Fig. 2. Signal $\boldsymbol{x}$ is considered to have only 2 nonzero frequencies, and $\hat{\boldsymbol{v}}$ is its bucketized signal driven by (a) summation of adjacent frequency or, (b) periodic summation of frequencies.

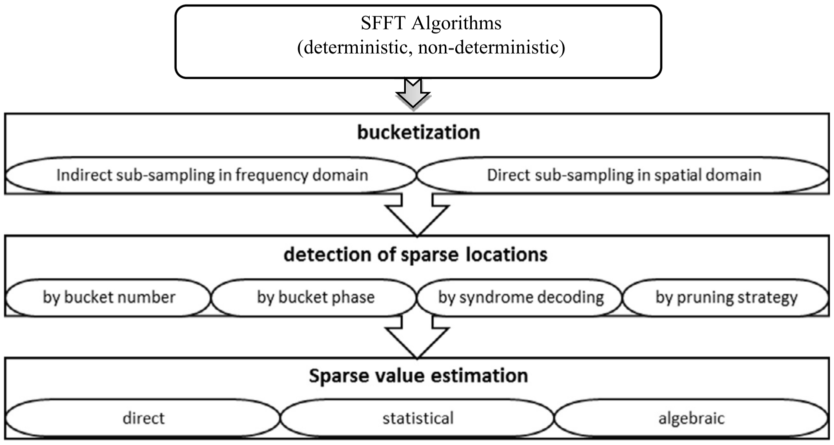


Fig. 3. Different methods used in the main steps of SFFT algorithms.

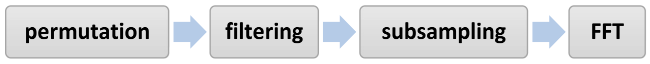


Fig. 4. Indirect frequency subsampling.

2.3. *Estimation of sparse values*

At this step, the Fourier values of sparse frequencies are extracted by using buckets values. The challenge is to extract values as close as possible to the real values. Two main methods used at this step are:

1) Using direct values: here the values of the buckets with significant frequencies are considered as the sparse values. If it is needed, the phase changes of the buckets caused by the time shifts are compensated.
2) Using statistical values: here combination of both mode and mean values of the buckets is used to calculate sparse values. This method has better accuracy and robustness to the noise compared to the first one.

The above two methods and also a method based on algebraic approach are discussed in detail in section 3.

3. SFFT algorithms

SFFT algorithms, as shown in Fig. 3 can be categorized based on the methods used in the performing of the main steps of the algorithms. Different methods and corresponding algorithms are explained in this section.

3.1. *Bucketization algorithms*

3.1.1. *Indirect sub-sampling in the frequency domain*

In the bucketization step, significant frequencies of the signal are placed in different individual buckets. In some SFFT algorithms, the method of indirect sub-sampling in the frequency domain is used to perform this step [13], [16], [20–29]. This method as shown in Fig. 4 includes four steps of permutation, filtering, down-sampling, and FFT.

Permutation step uniforms the distribution of the spectrum to prevent the presence of more than one frequency in each bucket. Figs. 5 (a) and (b) show a sample signal in spatial and frequency domains respectively. In Fig. 5 (b) two significant frequencies of the signal are close to each other. Figs. 5 (c) and (d) show the signal after the permutation step in spatial and frequency domains respectively. Fig. 5 (d) shows how the significant frequencies are separated. The permutation is performed by using (4) in the spatial domain, and the result in the frequency domain is as expressed in (5). Parameters a, b and σ in the equations are permutation parameters.

$$x'_i = x_{\sigma(i-a)\,mod\,N} . e^{\frac{-2\pi ibj}{N}} \quad a, b, \sigma \epsilon \mathbb{Z}^+ \tag{4}$$

$$\hat{X}'_{\sigma(i-b)\,mod\,N} = \hat{X}_i . e^{\frac{-2\pi iaj}{N}} \tag{5}$$

Performing filtering step after permutation step extends the values of significant frequencies to their adjacent frequencies. This prevents the elimination of some significant frequencies at the following frequency downsampling step. Filtering is done by convolving the permuted signal with a window function in Fourier domain, which is equivalent to multiplying the permuted signal by the window function in the spatial domain. Fig. 6 shows a sample 8 point window function in both spatial and Fourier domains, and Fig. 7 shows the result of applying this window on signal x' in Fig. 5.

Ideally, the window function in Fourier domain is a box-car function with a passband size equal to bucket size (equal to N/B; where N is the signal length and B is the total number of buckets) to avoid loss of significant frequencies. However, since the box-car function in frequency domain is equivalent to Sinc function in spatial domain with many significant coefficients, it leads to a high computational cost. To decrease computational cost the window's values are approximated. For a window function like box-car in spatial domain that has many significant elements in Fourier domain, this approximation mixes the significant frequencies in the bucket (so-called leakage) and results in an inaccuracy of the algorithm. It should be noted that this leakage is different from the leakage phenomenon occurs during sampling process. Design and selection of appropriate window function have been the subject of some researches [21], [23], [30], [31].

After filtering step, the signal is down-sampled simply by aliasing in the spatial domain as expressed in (6), in which B is the total number of the buckets and $\boldsymbol{v}$ is the bucketized signal.

$$v_i = \sum_{n=0}^{\frac{N}{B}-1} y_{i+Bn} \xrightarrow{DFT} \hat{V}_b = \hat{Y}_{b\frac{N}{B}} \quad i \in \{0, 1, \ldots, B-1\} \tag{6}$$

In some algorithms, downsampling is carried out without applying (6) and performed by just eliminating the tail of zero-signal members [24]. This reduces the computations but requires an appropriate window function. As mentioned before if the permuted signal is downsampled without applying a proper window, some significant Fourier values will be lost. For example, Fig. 8 shows

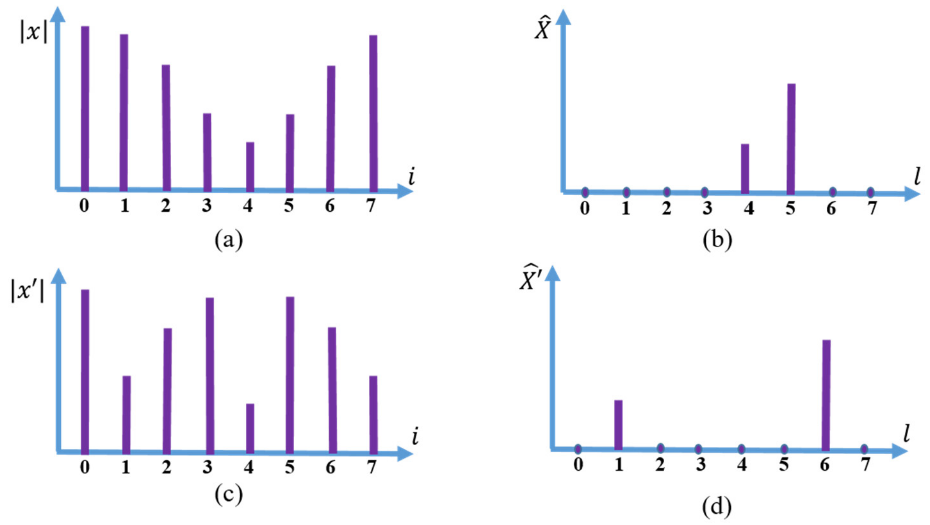


Fig. 5. Permutation step: (a) and (b) input signal in spatial and frequency domain respectively; (c) and (d) permuted signal in spatial and frequency domain respectively. The signals in Fourier domain are real and in spatial domain are complex. Absolute values are shown in spatial domain.

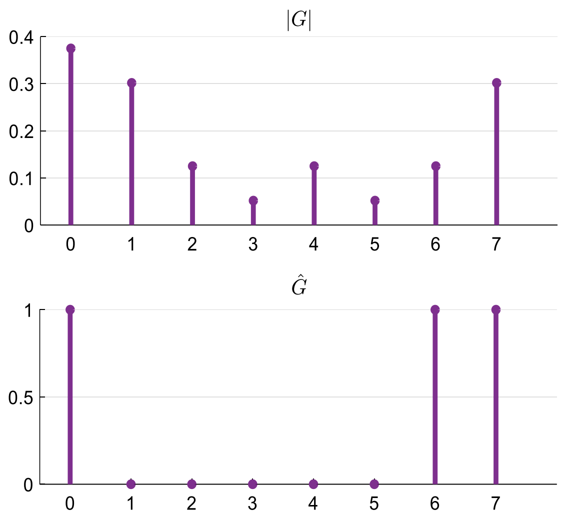


Fig. 6. An 8 point window function in spatial (G) and Fourier ($\hat{G}$) domains.

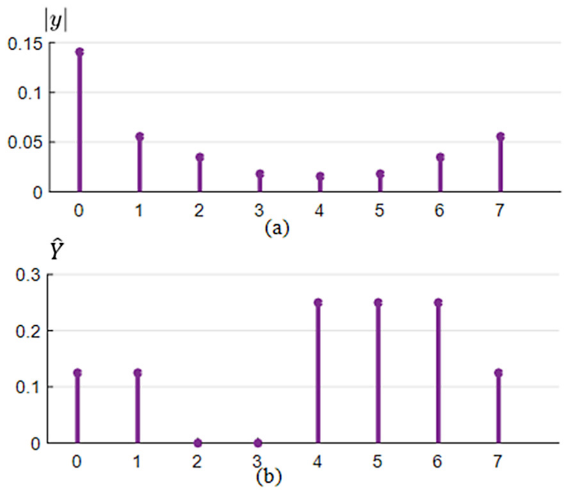


Fig. 7. Filtering step: signal y is the result of multiplication of permuted signal $\boldsymbol{x}'$ in Fig. 5 and window function G in Fig. 6.

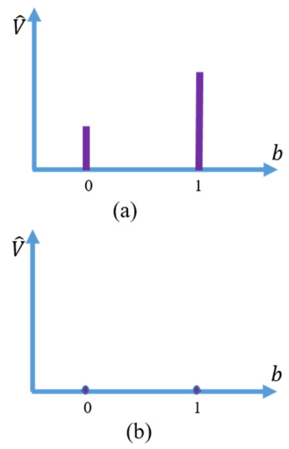


Fig. 8. The results of bucketization of signal $\boldsymbol{x}$ shown in Fig. 5, for two cases of: (a) applying filter, and (b) not applying filter.

the results of the bucketization step on signal $\boldsymbol{x}$ in Fig. 5 for two cases of applying and not applying the filter. After the downsampling step, to obtain the values of packets, B point FFT is applied to the bucketized signal. There is a low probability that after the permutation step still some significant frequencies lie next to each other. This causes the presence of more than one significant value in some buckets (collision) that leads to accuracy degradation. High levels of collision can lead to wrong results. Such algorithms, therefore, are called probabilistic or nondeterministic algorithms. Low levels of collision can be compensated by applying some techniques that are discussed later.

The indirect sub-sampling bucketization approach compared to the direct sub-sampling method has more computational load. In some works, the permutation step of this approach was modified to reduce computational load [32].

3.1.2. Direct sub-sampling in the spatial domain

Signal aliasing in Fourier domain can be performed simply by periodic summation of Fourier coefficients as expressed in (7). This method which was first used in [14], besides having a comparatively simple implementation, has the advantage of no leakage of buckets on each other. Different SFFT algorithms have used this approach, and it is mostly used in hardware implementations of

SFFT [15], [33], [34], and [35]. In these works, bucketization step is reiterated with different shifts and subsampling rates.

$$v_i = x_{i\frac{N}{B}} \xRightarrow{DFT} \hat{V}_b = \sum_{n=0}^{\frac{N}{B}-1} \hat{X}_{b+Bn} \quad i, b \in \{0, 1, \dots, B-1\} \tag{7}$$

In some works, downsampling is performed in digital domain [14] and [33]. In [33] to solve the collision problem, signal is subsampled with two co-prime sampling rates. It works based on the fact that if two frequencies collide in one of the bucketized signals, they cannot collide in another bucketized signal. In some works, implementation is done in analog domain [15] and [34]. Here low-speed ADCs are used and by controlling the switching times of the ADCs, different time shifts and sampling rates are obtained.

A two-dimensional SFFT, with a spatial domain subsampling approach was presented in [17]. It works based on a projection slice theorem in which 2D data are projected into some different 1D lines. The FFT of the lines provides different sums of coefficients in the spectrum.

In [36] the N-point signal is first downsampled to one point. If the point has a significant value then the signal is down-sampled to two points to find the distribution of coefficients in these two points. This approach of doubling the sample size continues in the following iterations (i.e., the length of bucketized signals in consecutive iterations of the algorithm are 1, 2, 4 ...) until all significant coefficients are detected.

3.2. Detection of sparse locations

3.2.1. Using bucket number

In some algorithms to obtain locations of significant frequencies, the numbers (indices) of the buckets with significant frequencies are used [15], [16], [17], [19], [20], [24–29], [37]. In this approach, for the bucketization step, the method of indirect subsampling in the frequency domain is mostly used.

In some algorithms like [20] numbers of large value buckets are saved and then permutation and downsampling are applied on all frequency locations (i.e., $0, 1, \dots, N-1$). By this, the frequencies that result in significant bucket numbers are extracted. For a simple case of permutation in spatial domain, by using (4) and also using the ideal window of Fig. 6, it can be mathematically expressed as:

$$\left\lfloor \left((\sigma (l-b) + w_2) \ mod\ N \right) \frac{B}{N} \right\rfloor = b_i \tag{8}$$

in which b_i is the index of the ith significant bucket, l is the frequency location, b and σ are permutation parameters as expressed in (4), and w_2 is the higher cut-off frequency of the window function. Ideally, the passband of the filter is equal to bucket size ($w_2 - w_1 + 1 = \frac{N}{B}$). The extracted frequencies that satisfy (8) are entered into a numerical set. Considering the union of sets, highly repeated frequencies are recognized as significant frequencies. This algorithm is used in some hardware implementation of SFFT [28], [38], [39]. Fig. 9 illustrates the algorithm used in [20]. The value estimation block in the Figure will be discussed later in the value estimation section.

In some algorithms like [29] to detect sparse locations the reversibility property of the permutation is used. Here, by using a reverse map equation instead of (8), locations of frequencies in a certain bucket are directly determined as expressed in (9), in which σ^{-1} is the modular inverse of σ, and $j \in \{0, 1, \dots, \frac{N}{B} - 1\}$.

$$l = \left((\frac{N}{B} b_i + j - w_2)\sigma^{-1} + b \right) mod\ N \tag{9}$$

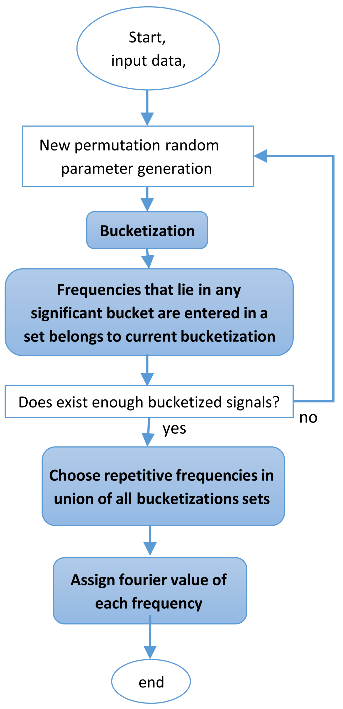

Start, input data,
New permutation random parameter generation
Bucketization
Frequencies that lie in any significant bucket are entered in a set belongs to current bucketization
Does exist enough bucketized signals?
yes
no
Choose repetitive frequencies in union of all bucketizations sets
Assign fourier value of each frequency
end

Fig. 9. The SFFT algorithm proposed in [20].

In [36] bucketization is performed by using direct downsampling at different sampling rates. In each iteration, DFT of the down-sampled signal determines the value of a bucket. All the frequencies that lie in a bucket with a negligible value are considered frequencies with zero Fourier values (this is a drawback because significant frequencies may cancel out each other), and through this, locations of significant frequencies are revealed too. At jth iteration of the algorithm, if b_i is the ith bucket with zero value, then the set of frequencies that lie in this bucket is found by (10). The algorithm used in [36] is shown in Fig. 10. How the Fourier value of the detected frequencies is obtained will be discussed later in the value estimation section.

$$\left\{ l \mid l = k * 2^j + b_i, k \in \left\{ 0, 1, \dots, \frac{N}{2^j} - 1 \right\} \right\}, \left(downsampling\ rate = \frac{N}{2^j} \right) \tag{10}$$

3.2.2. Using bucket phase

The technique of using phase difference values for locating sparse frequencies was first used in [21] and later in some other works like [22], [23], and [16]. In this method, by the assumption of ideal window function, bucketized signal is obtained by using (11) and (12), in which signal $\boldsymbol{x'}$ denotes signal $\boldsymbol{x}$ with m points shift, m is the amount of time shift and l_{i_b} denotes ith frequency in bth bucket. By the assumption that $\boldsymbol{x}$ is sparse in Fourier domain and at most one significant frequency exists in each bucket, (12) is simplified to (13), in which l_b is the significant frequency in bth bucket. By using these data, location of significant frequency can be calculated by (14).

$$\hat{\boldsymbol{V}} = \left(\hat{V}_b = \hat{X}_{l_{0_b}} + \hat{X}_{l_{1_b}} + \dots + \hat{X}_{l_{(\frac{N}{B}-1)_b}} \right)_{b=0}^{B-1} \tag{11}$$

$$\hat{\boldsymbol{V}}' = \left(\hat{V}'_b = \hat{X}'_{l_{0_b}} + \hat{X}'_{l_{1_b}} + \dots + \hat{X}'_{l_{(\frac{N}{B}-1)_b}} \right)_{b=0}^{B-1}$$

$$= \left(e^{\frac{2\pi m l_{0_b} j}{N}} \hat{X}_{l_{0_b}} + e^{\frac{2\pi m l_{1_b} j}{N}} \hat{X}_{l_{1_b}} + \dots + e^{\frac{2\pi m l_{(\frac{N}{B}-1)_b} j}{N}} \hat{X}_{l_{(\frac{N}{B}-1)_b}} \right)_{b=0}^{B-1} \quad (12)$$

$$\hat{\boldsymbol{V}}' = \left(\hat{V}'_b = e^{\frac{2\pi m l_b j}{N}} \hat{X}_{l_b} \right)_{b=0}^{B-1} \quad (13)$$

$$l_b = \frac{N}{2\pi m} \text{phase} \left(\frac{\hat{V}'(b)}{\hat{V}(b)} \right) \quad (14)$$

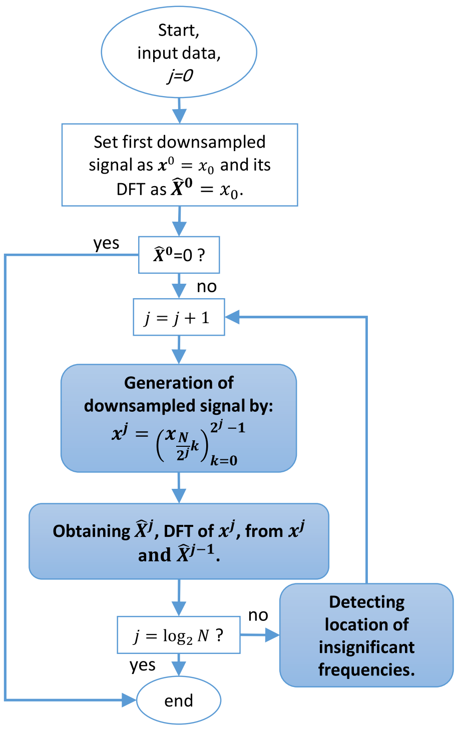

Start, input data, j=0
Set first downsampled signal as $x^0 = x_0$ and its DFT as $\hat{X}^0 = x_0$.
$\hat{X}^0$=0 ?
yes
no
$j = j + 1$
Generation of downsampled signal by: $x^j = \left(x_{\frac{N}{2^j}k}\right)_{k=0}^{2^j-1}$
Obtaining $\hat{X}^j$, DFT of x^j, from x^j and $\hat{X}^{j-1}$.
$j = \log_2 N$?
no
yes
Detecting location of insignificant frequencies.
end

Fig. 10. Block diagram of [36].

Two different algorithms have been proposed in [21] for the two cases of exact and general sparse signals. The algorithms are mainly different in the location detection step. The exact sparse algorithm uses a mathematical expression similar to (14). For the general sparse case, the direct use of (14) for different time shifts leads to different answers (sparse locations). So, phase difference approach has been combined with a search method to find a location in a bucket, which has minimum variance from various outputs of (14) for different time shifts. For each bucket, the entire frequency locations are checked, and the best location is saved. Each search iteration is limited to a specific distance from the selected location in previous iteration. The search is repeated until the selected locations converge to a specific frequency. Fig. 11 shows the main parts of the proposed SFFT algorithm, especially the location detection part for the general sparse case. As the figure shows the algorithm has several nested loops, causing high computational load. This algorithm which is noise-robust is used in [16], [25], [38], [40].

The frequency detection algorithm proposed in [34] is similar to the one presented in [21] (for the case of general sparse signals) with some modifications to improve the run time of the algorithm. In [21] as depicted in Fig. 11, the bucketization step is repeated many times, and in [34] the number of needed bucketizations has been reduced by half.

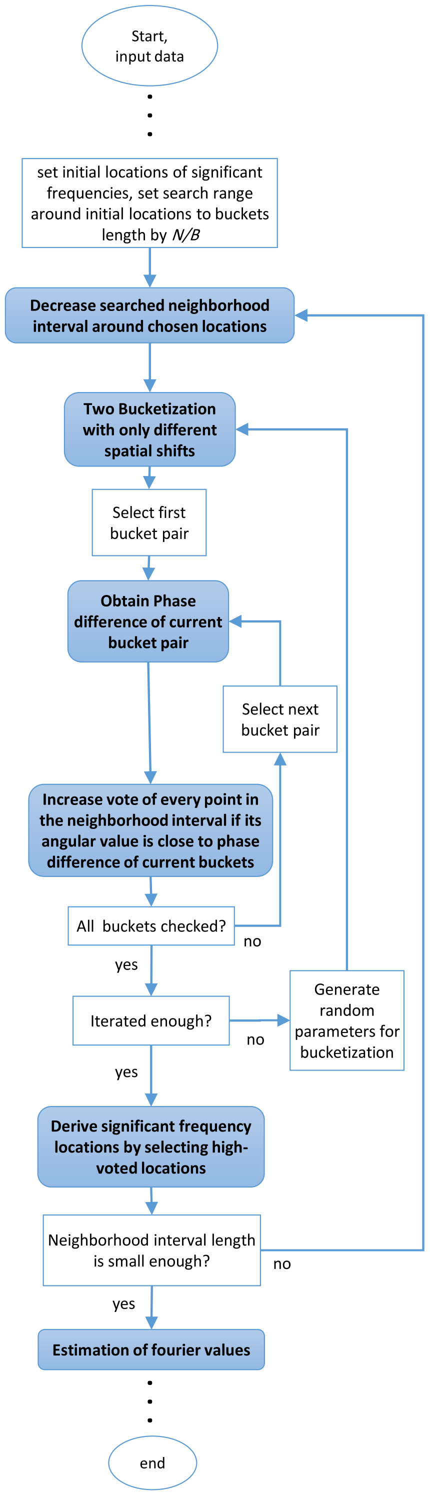

Start, input data
set initial locations of significant frequencies, set search range around initial locations to buckets length by N/B
Decrease searched neighborhood interval around chosen locations
Two Bucketization with only different spatial shifts
Select first bucket pair
Obtain Phase difference of current bucket pair
Select next bucket pair
Increase vote of every point in the neighborhood interval if its angular value is close to phase difference of current buckets
All buckets checked?
no
yes
Iterated enough?
no
Generate random parameters for bucketization
yes
Derive significant frequency locations by selecting high-voted locations
Neighborhood interval length is small enough?
no
yes
Estimation of fourier values
end

Fig. 11. The main components of the algorithm presented in [21] for the general sparse signals.

The sparse locations detection algorithm used in [33] is similar to the one in [21] for the case of exact sparse signals. Here, to avoid wrong detection results caused by collision problem, absolute values of buckets are checked to be equal for different shifts. If they are not, a second bucketization step with a co-prime sampling rate is performed to detect significant frequency locations. This algorithm does not work well for noisy or general sparse signals [26].

In [34] a new location detection approach was proposed to overcome the noise problem. The idea is to use the mean direction of unit vectors in consecutive bucketized signals. The method can be expressed as in (15), where P is the total number of ADCs in the design and $\hat{V}_b^k$ is the value of bth bucket of bucketized signal resulting from kth ADC. The use of ADCs is mentioned before in the direct subsampling section. All consecutive ADCs pairs (i.e. kth and k+1th ones) have equal time shifts. By using this detection approach, the SFFT algorithm in [34] has a lower computational load compared to that of the general sparse case algorithm presented in [21].

$$l_b = \frac{N}{2\pi m} phase \left(mean \left(\left\{ \frac{\left|\hat{V}_b^k\right| \hat{V}_b^{k+1}}{\left|\hat{V}_b^{k+1}\right| \hat{V}_b^k} \right\}_{k=0}^{P} \right) \right) \tag{15}$$

3.2.3. Using syndrome decoding

The syndrome decoding method introduced in [41], [42] was used in [22] for sparse location detection. In general, for some equidistant spatial domain samples we have the following expressions:

$$\begin{aligned} x_0 &= \frac{1}{N}\sum_{l=0}^{N-1} \hat{X}_l \\ x_1 &= \frac{1}{N}\sum_{l=0}^{N-1} \hat{X}_l e^{\frac{2\pi l j}{N}} \\ &\vdots \\ x_{t-1} &= \frac{1}{N}\sum_{l=0}^{N-1} \hat{X}_l e^{\frac{2\pi (t-1) l j}{N}} \end{aligned} \tag{16}$$

that can be expressed in the form of BCH[1] code as follows:

$$\begin{aligned} s_0 &= a_0 z_0^0 + a_1 z_1^0 + \dots + a_{m-1} z_{m-1}^0 \\ s_1 &= a_0 z_0^1 + a_1 z_1^1 + \dots + a_{m-1} z_{m-1}^1 \\ &\vdots \\ s_{t-1} &= a_0 z_0^{t-1} + a_1 z_1^{t-1} + \dots + a_{m-1} z_{m-1}^{t-1} \end{aligned} \tag{17}$$

in which s_i's, which are called syndromes, are known values, and z_i^k *and* $a_i's$ are unknown. $z_0, z_1, \dots, z_{m-1}$ are the roots of polynomial $P(z)$ in (18), in which $c_0, c_1, \dots, c_{m-1}$ can be extracted from syndromes by some simple matrix operations [42]. By having $z_0, z_1, \dots, z_{m-1}$ locations of significant frequencies are specified.

$$P(z) = z^m + c_{m-1} z^{m-1} + \dots + c_1 z^1 + c_0 \tag{18}$$

Using spatial domain samples leads to the presence of all significant frequencies in (17), and as a result, a high value of m that is equal to K. This makes solving equations (17) and (18) very complicated. In [43] to mitigate the problem instead of solving these equations with a high value of m, multiple of them with smaller m are considered by using the FFT of the downsampled signal. It reduces computational complexity, especially in the root-finding procedure. In this method, as expressed in (19) the input data is shifted by a specific value, downsampled, and then FFT is performed.

[1] Bose–Chaudhuri–Hocquenghem.

$$\begin{aligned} \hat{X}_l^{d,0} &= \frac{1}{d}\sum_{i=0}^{d-1} \left(\hat{X}_{l+i*\frac{N}{d}}\right) \\ \hat{X}_l^{d,1} &= \frac{1}{d}\sum_{i=0}^{d-1} \left(\hat{X}_{l+i*\frac{N}{d}}\right) e^{\frac{2\pi\left(l+i*\frac{N}{d}\right)j}{N}} \\ \hat{X}_l^{d,2} &= \frac{1}{d}\sum_{i=0}^{d-1} \left(\hat{X}_{l+i*\frac{N}{d}}\right) e^{\frac{4\pi\left(l+i*\frac{N}{d}\right)j}{N}} \\ &\vdots \\ \hat{X}_l^{d,t-1} &= \frac{1}{d}\sum_{i=0}^{d-1} \left(\hat{X}_{l+i*\frac{N}{d}}\right) e^{\frac{2(t-1)\pi\left(l+i*\frac{N}{d}\right)j}{N}} \end{aligned} \tag{19}$$

In (19) $\hat{X}_l^{d,2}$ denotes the lth element of the FFT of the signal that is shifted by 2 and is down-sampled by factor d. Expression (19) has the form of BCH code as expressed in (17).

3.2.4. Using pruning strategy

This method was proposed in [43] for the general sparse signals. It is derived from a subspace-pursuit compressed sensing algorithm [44]. Here, like (19), by considering down-sampled frequencies, syndrome sentences are obtained but insignificant nonzero terms exist too. As a result, instead of (17) we have:

$$\begin{aligned} & s_0 = a_0 z_0^0 + a_1 z_1^0 + \dots + a_{m-1} z_{m-1}^0 + a'_0 z_0'^0 + \dots \\ & + a'_{m'-1} z_{m'-1}'^0 \\ & s_1 = a_0 z_0^1 + a_1 z_1^1 + \dots + a_{m-1} z_{m-1}^1 + a'_0 z_0'^1 + \dots \\ & + a'_{m'-1} z_{m'-1}'^1 \\ & \vdots \\ & s_{t-1} = a_0 z_0^t + a_1 z_1^t + \dots + a_{m-1} z_{m-1}^t + a'_0 z_0'^{t-1} + \dots \\ & + a'_{m'-1} z_{m'-1}'^{t-1} \end{aligned} \tag{20}$$

where $z_i^k, z_i'^k$, a_i and a_i''s are unknown, z_i^k and $z_i'^k$'s are related to the location of significant and insignificant frequencies, and a_i and a_i''s are related to the values of significant and insignificant frequencies. (20) can be rewritten in a matrix form as:

$$S_{t\times 1} = Z_{t\times d} A_{d\times 1} \tag{21}$$

where d=m+m'. By using $S_{t\times 1}$ and the singular value decomposition technique, some polynomials like (18) with different orders are created. Frequencies that minimize these polynomials are detected, other frequencies are marked as insignificant nonzero frequencies, and corresponding columns in $Z_{t\times d}$ are pruned. Each column in $Z_{t\times d}$ corresponds to one frequency.

3.3. Sparse value estimation

3.3.1. Direct approach

In this approach which is applied on exact sparse signals, the values of buckets from non-shifted input data are simply considered as the values of sparse frequencies [21], [22], and [15]. For example by the assumption that l_b is obtained from (14) sparse value is estimated as:

$$\hat{X}_{l_b} = \hat{V}_b \tag{22}$$

3.3.2. Statistical approach

In many SFFT algorithms especially in the general sparse cases, the median of the values of the bucket is considered as sparse

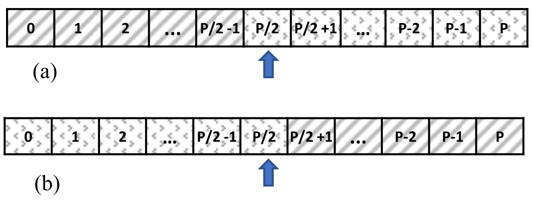


Fig. 12. Result of the median for the two worst cases when all erroneous values are (a) greater than, and (b) less than the correct value. Erroneous values are indicated in cross-hatched and the median is specified by an arrow.

value [20], [21], [23], [24], and [37]. In multiple bucketization operations, buckets with erroneous values (due to collision, noise, etc.) are in minority, and buckets with correct values are in majority. As shown in Fig. 12 median value gives an almost correct value of significant frequency. Before applying the median, buckets values are compensated for the effect of phase shift and also the effect of multiplication of filter coefficients. So we have:

$$\hat{X}_{l_b} = median\left(\left\{\hat{V}_b^k \frac{e^{\frac{2\pi l_b m_k j}{N}}}{\hat{G}_{h(l_b)}}\right\}_{k=0}^{P}\right) \tag{23}$$

where $h(l_b)$ denotes the location of filter coefficient multiplied by the l_bth frequency of the input signal. The median value is computed separately for the both real and imaginary parts. In some algorithms, for more assurance, the median value is checked to be more than a pre-specified threshold value, and if it is not, the l_bth frequency is considered as an insignificant frequency [21].

In [24], as shown in Fig. 13, the steps of location detection and value estimation are integrated. Here, all frequencies are entered into a Fourier value estimation stage. If the final assigned Fourier value is more than a specified threshold, then the frequency is added to the frequency elements of the estimated signal. In this algorithm, at each iteration step, the estimated signal is subtracted from the input signal. This increases the sparsity of the signal and improves the accuracy of the result.

In some works, the mean value of the buckets with significant frequencies, instead of the median value, is considered as the value of detected frequency [26]. In [31] using the mode/mean-based approach was proposed to overcome the degradation of median estimation in some conditions.

3.3.3. Algebraic approach

In [17], to estimate sparse values, a system of linear equations based on the location values of sparse frequencies and also some points of spatial domain is created and then by using a matrix pseudo inversion operation it is solved. In a location detection method based on the syndrome decoding [22], [43], first, the locations of significant frequencies (i.e. $z_0, z_1, \ \ldots, z_{m-1}$) are found, then based on that, a linear equation system like (17) is created, and then, equations are solved by using a matrix inversion approach. It is somewhat computationally expensive. Considering the structures of equations in (17), the matrix inversion operation can be simplified to Vandermonde matrix inversion operation, and by solving the equations, values of a_j which give the values of significant frequencies are obtained.

In the pruning-based algorithm [43], that was explained in the previous section, to estimate frequencies values, $Z_{t\times d}$ in (21) is first replaced by its pruned version Z^p to reduce computational complexity, and then it is solved by matrix inversion or the Gauss-Jordan method. The resulting $A_{d\times 1}$ contains the values of significant, and also a few insignificant frequencies.

Another algebraic approach was proposed in [36]. Here, by the assumption that $\boldsymbol{x}^{(j-1)}$ is the 2^{j-1}-point down-sampled of signal $\boldsymbol{x}$ and $\hat{\boldsymbol{X}}^{(j-1)}$ is its Fourier transform, at each algorithm iteration first half section of the Fourier transform of the down-sampled signal is obtained by solving the following linear system:

$$\boldsymbol{A}^{(j)}\tilde{\hat{\boldsymbol{X}}}_{\boldsymbol{0}}^{\boldsymbol{j}} = \frac{1}{2}(\boldsymbol{z}^{(j)} + \boldsymbol{A}^{(j)}\tilde{\hat{\boldsymbol{X}}}^{\boldsymbol{j-1}}) \tag{24}$$

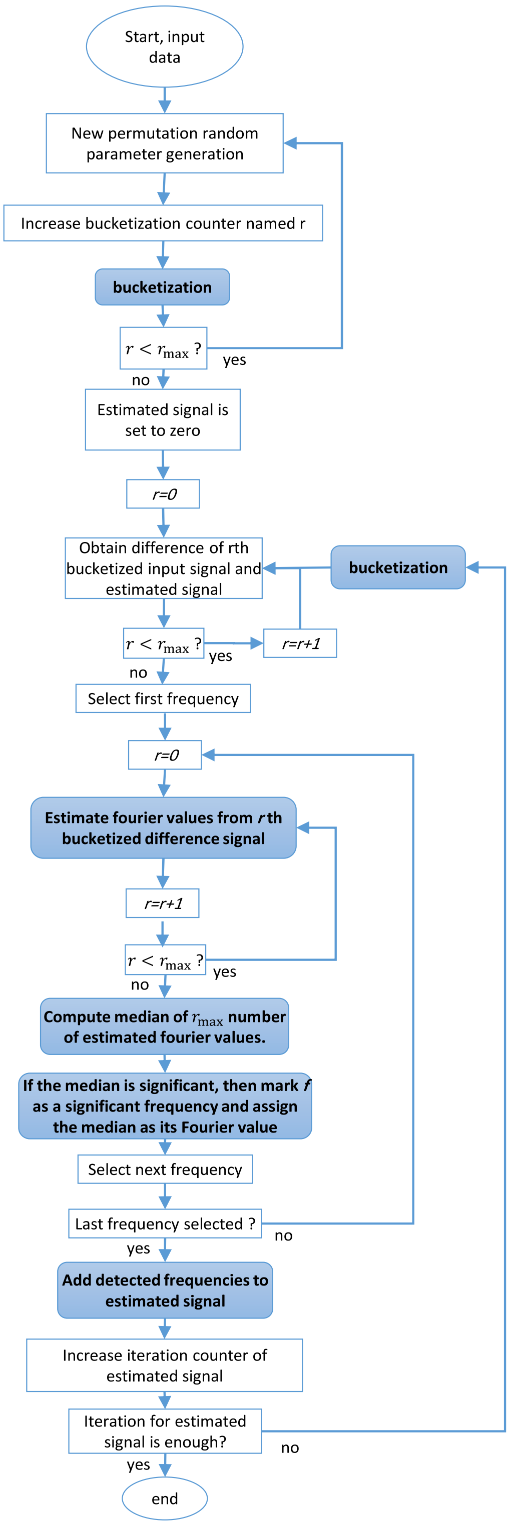

Start, input data
New permutation random parameter generation
Increase bucketization counter named r
bucketization
$r < r_{\max}$?
yes
no
Estimated signal is set to zero
r=0
Obtain difference of rth bucketized input signal and estimated signal
bucketization
$r < r_{\max}$?
yes
r=r+1
no
Select first frequency
r=0
Estimate fourier values from r th bucketized difference signal
r=r+1
$r < r_{\max}$?
yes
no
Compute median of $r_{\max}$ number of estimated fourier values.
If the median is significant, then mark f as a significant frequency and assign the median as its Fourier value
Select next frequency
Last frequency selected ?
no
yes
Add detected frequencies to estimated signal
Increase iteration counter of estimated signal
Iteration for estimated signal is enough?
no
yes
end

Fig. 13. SFFT algorithm of [24].

Table 1
Synthesis results of some different SFFT algorithms.

Ref. No.		Categorization			Hardware platform	Computation time	Maximum K	Recourse usage			
		Bucketization	Detection of sparse locations	Sparse value estimation				Registers	LUTs	RAM bits	DSP48Es
[33]		Direct subsampling	Bucket phase	Direct	ASIC (IBM's 45 nm SOI technology)	6.8 us ($N=3^6\times2^{10}$, K=N/1000)	3^6	–	–	–	–
[38]	Algorithms 1 and 2	Indirect subsampling	Bucket number	Statistical	Xeon E5-2660 CPU	100 ms ($N=2^{22}$, $K=2^8$)	2^{10}	–	–	–	–
	Algorithm 3	Indirect subsampling	Bucket phase	Statistical	Xeon E5-2660 CPU	0.5 ms ($N=2^{22}$, $K=2^8$)	2^{12}	–	–	–	–
[26]		Indirect subsampling	Bucket number	Statistical	FPGA XC6VLX240T	1.16 ms ($N=2^{20}$, $K=2^9$)	2^9	72,346	72,346	3,893,760	122
[34]		Direct subsampling	Bucket phase	Statistical	FPGA Stratix IV	546 us ($N\approx2^{18}$, $K\approx2^{12}$)	2^{12}	30,193	23,168	2,242,378	88
[40]		In-direct subsampling	Bucket number	Statistical	GPU	40 ms (K=1000, $N=2^{22}$)	2^{13}	–	–	–	–
[45]		Indirect subsampling	Bucket phase	Statistical	CPU	50 ms (K=1000, $N=2^{22}$)	2^{12}	–	–	–	–
[46]		Indirect subsampling	Bucket phase	Statistical	SOC FPGA 5CSEMA5F31C6	2.2 s ($K=2^{10}$, $N=2^{22}$)	2^{11}	11,346	7,654	2,629,734	8
[35]		Direct subsampling	Bucket number	Statistical	FPGA - XC6VLX240T	-	2^{12}	120,111	88,124	4,901,321	77

where $\hat{\boldsymbol{X}}_{\mathbf{0}}^{(\boldsymbol{j})}$ is the first half section of Fourier transform of $\boldsymbol{x}^{(j)}$, $\tilde{\hat{\boldsymbol{X}}}_{\mathbf{0}}^{\boldsymbol{j}}$ is significant elements of $\hat{\boldsymbol{X}}_{\mathbf{0}}^{(\boldsymbol{j})}$, $\boldsymbol{z}^{(\boldsymbol{j})}$ is consist of M_{j-1} points of signal $\boldsymbol{x}$ where M_{j-1} is the number of significant elements of $\hat{\boldsymbol{X}}^{(\boldsymbol{j-1})}$, $\tilde{\hat{\boldsymbol{X}}}^{\boldsymbol{j-1}}$ is consist of significant elements of $\hat{\boldsymbol{X}}^{(\boldsymbol{j-1})}$, and $\boldsymbol{A}^{(\boldsymbol{j})}$ is an $M_{j-1}\times M_{j-1}$ matrix. Entries of $\boldsymbol{A}^{(\boldsymbol{j})}$ are derived from locations of significant frequencies. After solving (24), $\hat{\boldsymbol{X}}_{\mathbf{1}}^{(\boldsymbol{j})}$, the second half section of Fourier transform, is obtained simply by subtraction of $\hat{\boldsymbol{X}}_{\mathbf{0}}^{(\boldsymbol{j})}$ from $\tilde{\hat{\boldsymbol{X}}}^{\boldsymbol{j-1}}$. Since the size of vectors and matrices of this system depends only on significant values of $\hat{\boldsymbol{X}}^{(\boldsymbol{j-1})}$ its computational load is reduced.

Use of algebraic approach is recommended for when the number of significant frequencies is high (i.e., for low sparsity degree) and the noise level is very low. For the high sparsity degrees, when the noise level is negligible using direct method and when the noise level is not low statistical approach is suggested.

At the end of this section, to better understand, the results of synthesizes of some prominent works according to different species of SFFT are expressed in Table 1. At the end of this section, in Table 1 the synthesize results of some different SFFT algorithms are presented.

4. Applications

SFFT has been utilized in different applications, and implemented on different platforms such as CPU, GPU, ASIC, and FPGA. The applications and platforms are discussed in this section.

4.1. Global navigation satellite system (GNSS)

GPS[2] receives a satellite signal and convolves it with the CDMA[3] code of the satellite. The output is a signal with a spike that its position in the signal provides information about the position of the

[2] Global positioning system.
[3] Code Division Multiple Access.

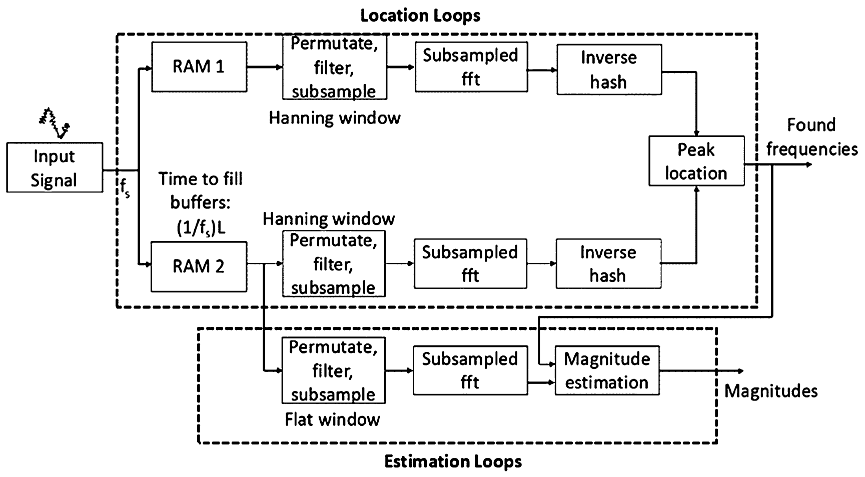


Fig. 14. Structure of implemented SFFT on FPGA for VSD [48].

system. In [14] to reduce the computation time, instead of direct convolution, FFTs of the received signal and CDMA code are multiplied, and then IFFT operation is performed to obtain output signal. Since the output signal is sparse, (inverse)FFT operation is replaced by (inverse)SFFT operation. This work was tested on a dataset collected from Europe and US Satellite GPS data. It achieved a reduction of about 2.2 times in the number of needed multiplications compared to that of the traditional FFT-based GPS synchronization algorithm. In another work, SFFT was utilized in the GNSS receiver [47]. The algorithm which was implemented on software achieved a run time of about 4 times faster than FFT-based GNSS receiver while maintaining robustness to the noise.

4.2. Variable speed drive (VSD)

VSDs improve the efficiency of electrical machines by compensating the disturbances like torque oscillations and mechanical vibrations [48]. To perform compensation, extraction of harmonic information is needed. FFT is commonly used to obtain this data. For real-time control, the FFT operation must be performed in parallel with some other control procedures, which could be difficult and expensive. Using sub-sampled FFT to decrease complexity leads to imprecise results. In [48] the signal of rotation speed was processed by SFFT to find its spectral components. To improve execution speed, frequency location detection and value estimation are performed in parallel (with different window functions for better performance) as depicted in Fig. 14. The SFFT which was implemented on a Virtex 6 FPGA consumed 30% fewer resources compared to that of a full-length implementation on the same FPGA.

4.3. GHz-wide sensing and decoding

The widespread use of wireless communication systems and the potential of approaching spectrum shortage have increased the need of multiple bands for dynamic spectrum sharing. In [15], by considering that the utilization of the spectrum is sparse in practice (2-5%), BigBand, an SFFT based technology, was proposed. It enables real-time GHz-wide spectrum sensing and reception by using low-power radios.

The key obstacle in capturing GHz of bandwidth in the real-time stems from the need for high-speed analog-to-digital converters, which are costly and power-hungry, besides having low bit resolutions [49], [50]. The compressive sensing technique requires random sampling of the signal which cannot be done simply by using standard low-speed ADCs. The process needs an analog

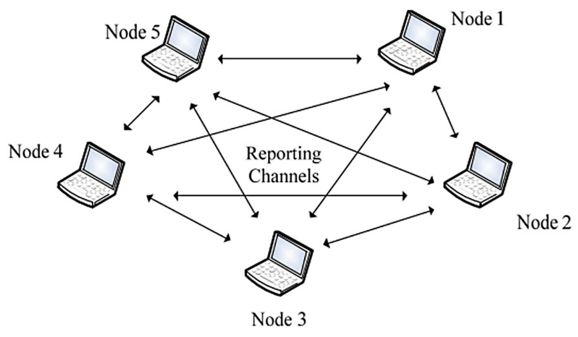


Fig. 15. The CWSS system with M nodes [34].

mixing at Nyquist rates and an expensive processing to recover original signal [51], [52]. Such a design is quite complex with a power consumption that could be comparable to the power consumption of an ADC working at Nyquist rate. BigBand needs only a few ADCs, that each samples at a rate much less than Nyquist rate. In [15], BigBand is implemented on USRP (Universal Software Radio Peripheral) software radios. Results show a high improvement in calculation speed compared to those of other spectrum sensing instruments. In [34], the noise robustness of SFFT-based spectrum sensing was improved. The proposed algorithm performs cooperative wideband spectrum sensing (CWSS) by using a network of connected nodes as shown in Fig. 15. Each node consists of a multi-coset sub-Nyquist sampling block (several ADCs for bucketization) and a recovery block (for frequency detection and estimation and also transmission). The sub-sampling rate of each node is relatively prime to those of other nodes, which helps collision resolution. Detected frequencies at each node are shared with other nodes. The algorithm has been implemented on a Stratix IV FPGA. For verification, a test multiband signal was contaminated with different noise levels. Results show a high improvement in noise robustness based on SRNR parameter.

4.4. Light field reconstruction

In 4D light field reconstruction, a four-variable function $L(u,v,x,y)$ is used to determine the color or grayscale information of the point (x,y) of the image from the viewpoint or camera coordinate (u,v). The image capturing from all (u,v) points, by using a 2D camera lens array and robotic moving cameras, requires expensive hardware and takes a long time to perform [17]. Some research was carried out to obtain needed data by using just a few images from different camera coordinates. 2D angular slice of the light spectrum $\hat{L}_{\omega_x,\omega_y}(\omega_u,\omega_v)$ is sparse. In [17] by using this characteristic, SFFT was used to reconstruct the 4D light field. In this work, the signal is first transferred from continuous space to discrete space, and then after frequency detection in discrete domain, an additional step is performed to locate sparse signals in the continuous domain by using a gradient search method. The algorithm that was implemented on a CPU platform, compared to other methods, resulted in a better light field reconstruction quality, despite requiring less number of samples.

4.5. Radar signal processing

Traditionally the radar signal is processed by a 3D FFT operation to detect the position and velocity of the objects. Almost always, the number of radar cells affected by a target is much less than the total number of cells. This means the radar signal is sparse. Various studies have been carried out to improve the speed and accuracy of radar signal processing by using the compressive sensing method. In [29] by using SFFT, the complexity of the process is reduced. In this work, the problems of unknown the sparsity level, noise, and off-grid significant frequencies are addressed too. The latter problem caused by the transferring of continuous radar signals to discrete space which can reduce signal sparsity. To overcome the problem, the SFFT algorithm in [20] has been modified by revising some of the algorithm's thresholds and adding a pre-permutation windowing stage. The new proposed algorithm, named RSFT (Robust Sparse Fourier Transform), provides much better target detection resolution compared to that of [20].

In another work in [13], SFFT is used in the classic RD (Range Doppler) algorithm of ISAR (Inverse Synthetic Aperture Radar) signals. This work that utilized the SFFT algorithm of [20] achieved half of the computing time of FFT-based processing of one-dimensional ISAR images.

4.6. Audio signal processing

The sparsity of some audio signals in the spectral domain has motivated many researches on the sparse processing of such signals. In [18] by using the SFFT algorithm in [20], only the position of significant frequencies and their corresponding values are sent through a transmission line. On the receiver side, the signal is reconstructed by placing the values on their specified frequencies, setting all other frequencies to zero, and performing IFFT. The algorithm was run on a Core i7 CPU. The results show that compared to other CS methods based on DWT, DCT, and DFT, performance of SFFT, in terms of time and accuracy, is much better.

In another study in [53], perceptual properties of the audio signal were used in SFFT-based compression. In the human auditory system, some frequencies are inaudible and some frequencies are more important. This characteristic is exploited by giving different weights to the frequencies, which leads to a higher degree of sparsity. Experimental results show improvement in the audio signal quality.

4.7. Magnetic resonance spectroscopy

2D Correlation Spectroscopy (COSY) is an effective method for unequivocal assignment of metabolites in localized regions of the body, which allows the detecting of new molecular biomarkers of the diseases. Despite its capability, the in vivo[4] use of traditional 2D COSY is not largely spread due to some problems, such as the limited number of samples, intrusive frequencies in the spectral domain, and considerable encoding time.

[4] In the living body of a plant or animal.

In [16], SFFT was used in the COSY processing algorithm. It was implemented in software and tested on three volunteers. Results show that SFFT-based COSY has reduced measurement time by around 72% compared with FFT-based and compressed sensing-based COSY algorithms, while achieving more robustness to truncation artifacts and noise.

In another work on the COSY algorithm, SFFT and ShMoCo[5] algorithms were combined [25]. It reduced both acquisition time and the disturbance caused by the scanned object's motion.

4.8. General-purpose hardware implementation

The first general-purpose ASIC implementation of SFFT was performed in IBM's 45 nm SOI technology [33]. It was capable of computing DFT of 746,496 ($2^{10}\times3^6$) point sparse data, a data with a degree of sparsity higher than 0.1% in the spectral domain.

The Bucketization step is performed by using a direct subsampling method and two FFT operations in the bucketization block. Sparse frequency detection is performed in the reconstruction block by using the phase change data. The phase change is caused by the time shifts applied to the 1st bucketization set. Collision is resolved by using the 2nd bucketization set, which is co-prime to the 1st set.

The hardware had about 40 times lower power consumption and 20 times higher throughput compared to some other low-power FFT works. However, it was sensitive to noise, especially in the frequency recovery block [26].

In [38] SFFT was implemented on the Xeon E5-2660 CPU platform. In this work, some techniques such as altering algorithm's loops order (to maximize cache memory usage), exploiting SIMD[6] instructions, and selecting optimal data format of complex numbers were applied in the C++ code. This work achieved more than 2 times speed improvement compared with a prototypical C++ implementation [55]. Another SFFT implementation, with the capability of processing a million point data with noise robustness, was proposed in [26]. Here, sparse frequency detection is performed based on the location information rather than phase difference information. By using a voting process, frequencies that are found in large buckets and received more votes are selected. In this work, the maximum selection unit that selects maximum value buckets, and the voter unit that determines the sparse frequency, were optimized for hardware efficiency. Also, in the voter unit, instead of recording the votes for all frequencies, a filtering structure was used to pass current frequencies of the large buckets to the next iteration step. At the end, only the frequencies that at all iterations have been present in large buckets remain. The proposed scheme was implemented on a Virtex-6 FPGA. Implementation parameters such as data type, subsampled FFT size, and the number of iterations, were considered in the hardware optimization to optimize resource consumption and speed. The achieved processing time is 1.16 ms, 85 times less than that of SFFT implementation on the Xeon E5-2660 CPU presented in [26]. The drawbacks of the system are its requirement for a high degree of sparsity and high resource usage.

In [28] implementation of SFFTon multi-core CPU and also GPU was studied. OpenMP and Pthread libraries were used to implement SFFT algorithm in a parallel structure on a multi-core CPU platform. Also, CUDA is used for the GPU case. The developed codes were tested on human voice signals. The Multi-core based implementation and the GPU implementation are, respectively, 3 times and about 10 times faster than the SFFT implementation presented in [38]. Also, the GPU framework was compared with

[5] Real-time motion correction and shim update [54].

[6] Single instruction multiple data.

the GPU-based FFT implementation presented in [56], which shows less run time for the SFFT structure. In [40] and [57] it is tried to optimize and accelerate the proposed GPU system in [28]. By using some techniques such as memory coalescing, minimizing data movement between CPU-GPU, and coarse-grained parallelism along with fine-grained parallelism, a significant improvement in run time was achieved. In [57] run time is 5 times faster than GPU-based FFT implementation in [7] and 38 times faster than implementation in [58].

An FPGA-based implementation of SFFT was proposed in [35]. It has improved the work presented in [26]. In comparison with [26], it is capable of working on different numbers of significant frequencies. In this work, the max select unit was replaced by a pipeline shift queue architecture. In the voter block, the filtering structure has been changed to a ticket counting structure. This work achieved an about 50% reduction in chip resource usage and 50% reduction in run time for N=2^{16}, in comparison with those presented in [26].

4.9. Optical fiber communication

Orthogonal frequency division multiplexing is a suitable choice for optical access networks. However, due to its high sensitivity to synchronization errors, its computational complexity and hardware usage are high. To overcome the problem a quick synchronization algorithm based on the SFFT algorithm presented in [20] was proposed in [59]. The method is tested on 50 km standard single-mode fiber and it achieved a well accuracy with a relatively low complexity.

Identifying modulation format is an essential procedure on the receiver side of an optical network. In [60] a method was proposed to distinguish QPSK and 16/32/64QAM modulations by using SFFT, which led to a significant reduction in algorithm complexity.

4.10. Deformation metrology

Some algorithms that use Fourier analyses of high-resolution images have been developed for interference fringes analysis. These algorithms transform interference fringes into unwrapped phase maps. Integrating the SFFT algorithm of [20] for Fourier operations and Electronic Speckle Pattern Interferometry successfully resulted in a new faster system [61] and [62].

5. Conclusion

SFFT has superseded FFT in some signal processing applications. It is more complex than FFT, but in terms of hardware, it is less resource-consuming. In FFT, few simple blocks are repeated in large numbers, whereas in SFFT, a lower number of blocks with different mathematical operations is required. Compared to FFT, SFFT has a higher execution speed and lower implementation cost for the big data that are sparse in the frequency domain. In recent years, various algorithms to improve noise robustness, speed, and adaptability, have been proposed for SFFT implementation. Selecting an appropriate algorithm depends on the parameters such as the spectrum of input signals, noise level, and hardware platform. However, in general, for the applications in which speed is the priority, for the bucketization step using direct spatial domain down-sampling (due to its simplicity and no frequency leakage), for the frequency detection step using phase information, and for the value estimation step using a statistical approach, are more suitable. Also for the applications in which accuracy is the priority, for frequency detection step pruning strategy based on compressed sensing, and for value estimation step algebraic approach are more suitable.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix A

In this section an example of a numerical execution of a non-deterministic sparse fast Fourier transform is presented. The transform is applied on signal $\boldsymbol{x}$, an 8-point input data, that its value and its Fourier transform ($\mathbf{X_{FFT}}$) are shown in rows 1 and 2 of Table 2 respectively.

First, by using relationships (4) and (5) the input data is permuted. The result ($\boldsymbol{x'}$) is shown in row 3 of the table. Then, spatial windowing is performed. Window coefficients are shown in both spatial and frequency domains in rows 4 and 5 respectively. In this example, a simple rectangular window is used and the degree of signal sparsity in the Fourier domain is very low equal to 0.25, and all coefficients of the window $\boldsymbol{F}$ in the spatial domain are non-zero. In a real processing operation, both the number of signal points and the degree of sparsity are much higher and also, the used rectangular window is not ideal and the number of its zero coefficients is high. After multiplying the window, a normalizing constant number (here number 8) is multiplied to compensate for the signal energy. The resulting signal after windowing and energy compensation is shown in row 6. In the next step, the signal is downsampled from 8 points to 2 points and the FFT is applied to this signal. The resulting signal $\hat{\boldsymbol{V}}_{\mathbf{1}}$ of this process is shown in row 7. The first point of the signal $\hat{\boldsymbol{V}}_{\mathbf{1}}$ is equal to the sum of frequencies 1 to 3 of signal $\boldsymbol{x}'_{\mathbf{1}}$ and the second point is equal to the sum of frequencies 5 to 7 of signal $\boldsymbol{x}'_{\mathbf{1}}$. This process is called frequency bucketization. Since the initial value of the signal $\hat{\boldsymbol{X}}$ (the parameter that represents FFT of $\boldsymbol{x}$) is equal to zero, the signal $\hat{\boldsymbol{V}}_{\mathbf{update}}$ obtained from the convolution of signal $\hat{\boldsymbol{X}}$ with the window and downsampling it to two points, is also equal to zero. Signal $\hat{\boldsymbol{Z}}_{\mathbf{1}}$ is obtained from the subtraction of $\hat{\boldsymbol{V}}_{\mathbf{update}}$ from $\hat{\boldsymbol{V}}_{\mathbf{1}}$, as shown in row 10. $\hat{\boldsymbol{Z}}_{\mathbf{1}}$, which shows new information discovered in the Fourier domain of the signal in the first iteration, is stored in a memory. Afterward, permutation, windowing, and downsampling operations are repeated in the second iteration. In this iteration the parameter “a″ used in relationships (4) and (5) is different from the previous one. The resulting signal of the second permutation process is given in row 11 of Table 2. In row 12 the values after the multiplication of the spatial window and energy compensation coefficient are shown, and in row 13 the result of downsampling to two points and applying the fast Fourier transform is shown. Since $\hat{\boldsymbol{V}}_{\mathbf{update}}$ is zero at this point, the resulting signal $\hat{\boldsymbol{Z}}_{\mathbf{2}}$ is equal to $\hat{\boldsymbol{V}}_{\mathbf{2}}$, as its values are shown in row 14. This signal is also stored in memory.

Now it is time to run the frequency location identification phase. Since signals $\boldsymbol{x}'_{\mathbf{1}}$ and $\boldsymbol{x}'_{\mathbf{2}}$ differ only in terms of spatial displacement, their related signals $\hat{\boldsymbol{Z}}_{\mathbf{1}}$ and $\hat{\boldsymbol{Z}}_{\mathbf{2}}$ in the frequency domain, vary only in phase and their amplitude absolute values are equal. So based on the values of $\hat{\boldsymbol{Z}}_{\mathbf{1}}$ and $\hat{\boldsymbol{Z}}_{\mathbf{2}}$ and their phase difference value, the location of significant frequencies is identified by using relation (2). The value of m in (2), in current example is equal to 3. The identified locations are shown in row 15 of the table. Afterward, the amplitudes of the significant frequencies are calculated. The results are shown in row 16. Using the data obtained, $\hat{\boldsymbol{X}}$ the Fourier transform of the signal, which was initially equal to zero (row 8), is updated as shown in row 17. The values in this row, which are the result of the execution of the sparse fast Fourier transform algorithm, are equal to the values in row 2, the values of the Fourier transform of the signal. In the next iteration

Table 2
Signal values during an execution of a basic SFFT algorithm on an 8 point sample data.

Row	Signal point / Signal name	1	2	3	4	5	6	7	8
1	$\boldsymbol{x}$	0.3750000 + 0.000000i	-0.08838835 + 0.3383884i	-0.2500000 - 0.1250000i	0.08838835 - 0.1616116i	0.1250000 + 0.000000i	0.08838835 + 0.1616116i	-0.2500000 + 0.1250000i	-0.08838835 - 0.3383884i
2	$\mathbf{X}_{\mathbf{FFT}}$	0	0	2	1	0	0	0	0
3	$\boldsymbol{x}'_1$	0.3750000 + 0.000000i	0.08838835 - 0.1616116i	-0.2500000 + 0.1250000i	-0.08838835 + 0.3383884i	0.1250000 + 0.000000i	-0.08838835 - 0.3383884i	-0.2500000 - 0.1250000i	0.08838835 + 0.1616116i
4	$\boldsymbol{F}$	0.3750000 + 0.000000i	0.2133884 - 0.2133884i	0.000000 - 0.1250000i	0.03661165 + 0.03661165i	0.1250000 + 0.000000i	0.03661165 - 0.03661165i	0.000000 + 0.1250000i	0.2133884 + 0.2133884i
5	$\hat{\boldsymbol{F}}$	1	0	0	0	0	0	1	1
6	$\mathbf{y}_1$	1.125000 + 0.000000i	-0.1250000 - 0.4267767i	0.1250000 + 0.2500000i	-0.1250000 + 0.07322332i	0.1250000 + 0.000000i	-0.1250000 - 0.07322332i	0.1250000 - 0.2500000i	-0.1250000 + 0.4267767i
7	$\hat{\boldsymbol{V}}_1$	1.000000 + 0.000000i	2.000000 + 0.000000i						
8	$\hat{\mathbf{X}}$	0	0	0	0	0	0	0	0
9	$\hat{\boldsymbol{V}}_{\mathbf{update}}$	0	0						
10	$\hat{\boldsymbol{Z}}_1$	1.000000 + 0.000000i	2.000000 + 0.000000i						
11	$\boldsymbol{x}'_2$	0.08838835 + 0.1616116i	0.3750000 + 0.000000i	0.08838835 - 0.1616116i	-0.2500000 + 0.1250000i	-0.08838835 + 0.3383884i	0.1250000 + 0.000000i	-0.08838835 - 0.3383884i	-0.2500000 - 0.1250000i
12	$\mathbf{y}_2$	0.2651650 + 0.4848349i	0.6401651 - 0.6401651i	-0.1616116 - 0.08838835i	-0.1098350 - 0.03661165i	-0.08838835 + 0.3383884i	0.03661165 - 0.03661165i	0.3383884 - 0.08838835i	-0.2133884 - 0.6401651i
13	$\hat{\boldsymbol{V}}_2$	0.7071068 - 0.7071068i	0.000 + 2.000000i						
14	$\hat{\boldsymbol{Z}}_2$	0.7071068 - 0.7071068i	0.000 + 2.000000i						
15	k	3	4						
16	f_k	2	1						
17	$\hat{\mathbf{X}}$	0	0	2	1	0	0	0	0

of the algorithm, the values of $\hat{Z}_1$ and $\hat{Z}_2$ will be zero, indicating that there is no new information and that all significant frequencies have been identified, thus the algorithm is ended.

References

[1] G. Bergland, Fast Fourier transform hardware implementations–a survey, IEEE Trans. Audio Electroacoust. 17 (2) (1969) 109–119.
[2] M. Garrido, A survey on pipelined FFT hardware architectures, J. Signal Process. Syst. (Jul. 2021) 1–20.
[3] International Telecommunication Union (ITU), Very high speed digital subscriber line transceivers 2 (VDSL2), 2019.
[4] F. Structure, Channel coding and modulation for a second generation digital terrestrial television broadcasting system (DVB-T2), European Standard, ETSI EN 302 (755) (2008) V1.
[5] G. Kang, W. Choi, J. Park, Embedded DRAM-based memory customization for low-cost FFT processor design, IEEE Trans. Very Large Scale Integr. Syst. 25 (12) (2017) 3484–3494.
[6] X.-Y. Shih, H.-R. Chou, Y.-Q. Liu, VLSI design and implementation of reconfigurable 46-mode combined-radix-based FFT hardware architecture for 3GPP-LTE applications, IEEE Trans. Circuits Syst. I, Regul. Pap. 65 (1) (2018) 118–129.
[7] K. Wieloch, K. Stokfiszewski, M. Yatsymirskyy, Effectiveness of partitioning strategies of fast Fourier transform in GPU implementations, in: 2017 12th International Scientific and Technical Conference on Computer Sciences and Information Technologies (CSIT), vol. 1, 2017, pp. 322–325.
[8] R. Baraniuk, P. Steeghs, Compressive radar imaging, in: 2007 IEEE Radar Conference, 2007, pp. 128–133.
[9] M. Lustig, D.L. Donoho, J.M. Santos, J.M. Pauly, Compressed sensing MRI, IEEE Signal Process. Mag. 25 (2) (2008) 72–82.
[10] T.T. Do, Y. Chen, D.T. Nguyen, N. Nguyen, L. Gan, T.D. Tran, Distributed compressed video sensing, in: 2009 43rd Annual Conference on Information Sciences and Systems, 2009, pp. 1–2.
[11] L. Balzano, R. Nowak, J. Ellenberg, Compressed sensing audio demonstration, website http://sunbeam.ece.wisc.edu/csaudio, 2010.
[12] A.C. Gilbert, P. Indyk, M. Iwen, L. Schmidt, Recent developments in the sparse Fourier transform: a compressed Fourier transform for big data, IEEE Signal Process. Mag. 31 (5) (2014) 91–100.
[13] Y.-Z. Gong, H. Tao, SFFT based ISAR imaging, in: 2nd Annual International Conference on Electronics, Electrical Engineering and Information Science (EEEIS 2016), 2017, pp. 35–42.
[14] H. Hassanieh, F. Adib, D. Katabi, P. Indyk, Faster GPS via the sparse Fourier transform, in: Proceedings of the 18th Annual International Conference on Mobile Computing and Networking, 2012, pp. 353–364.
[15] H. Hassanieh, L. Shi, O. Abari, E. Hamed, D. Katabi, GHz-wide sensing and decoding using the sparse Fourier transform, in: 33rd IEEE Conference on Computer Communications, IEEE INFOCOM 2014, 2014, pp. 2256–2264.
[16] L. Shi, O. Andronesi, H. Hassanieh, B. Ghazi, D. Katabi, E. Adalsteinsson, Mrs sparse-fft: reducing acquisition time and artifacts for in vivo 2d correlation spectroscopy, in: ISMRM13, Int. Society for Magnetic Resonance in Medicine Annual Meeting and Exhibition, 2013.
[17] L. Shi, H. Hassanieh, A. Davis, D. Katabi, F. Durand, Light field reconstruction using sparsity in the continuous Fourier domain, ACM Trans. Graph. 34 (1) (2014) 12.
[18] H.M. Kasem, M. El-Sabrouty, A comparative study of audio compression based on compressed sensing and sparse fast Fourier transform (SFFT): performance and challenges, in: IEEE International Symposium on Signal Processing and Information Technology, 2013, pp. 452–457.
[19] A.C. Gilbert, S. Muthukrishnan, M. Strauss, Improved time bounds for near-optimal sparse Fourier representations, in: Proceedings of SPIE - the International Society for Optical Engineering, vol. 5914, 2005, pp. 1–15.
[20] H. Hassanieh, P. Indyk, D. Katabi, E. Price, Simple and practical algorithm for sparse Fourier transform, SIAM J. Control Optim. (2012).
[21] H. Hassanieh, P. Indyk, D. Katabi, E. Price, Nearly optimal sparse Fourier transform, in: Proceedings of the Forty-Fourth Annual ACM Symposium on Theory of Computing, 2012, pp. 563–578.
[22] B. Ghazi, H. Hassanieh, P. Indyk, D. Katabi, E. Price, L. Shi, Sample-optimal average-case sparse Fourier transform in two dimensions, in: 2013 51st Annual Allerton Conference on Communication, Control, and Computing (Allerton), 2013, pp. 1258–1265.
[23] P. Indyk, M. Kapralov, E. Price, (Nearly) sample-optimal sparse Fourier transform, in: Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, 2014, pp. 480–499.
[24] P. Indyk, M. Kapralov, Sample-optimal Fourier sampling in any constant dimension, in: FOCS '14 Proceedings of the 2014 IEEE 55th Annual Symposium on Foundations of Computer Science, 2014, pp. 514–523.

[25] O.C. Andronesi, et al., Correlation chemical shift imaging with sparse-fft and real-time motion and shim correction, in: 55th Experimental Nuclear Magnetic Resonance Conference, 2014.
[26] A. Agarwal, H. Hassanieh, O. Abari, E. Hamed, D. Katabi, Arvind, High-throughput implementation of a million-point sparse Fourier transform, in: 2014 24th International Conference on Field Programmable Logic and Applications (FPL), 2014, pp. 1–6.
[27] S. Chen, X. Li, An input-adaptive algorithm for high performance sparse fast Fourier transform, in: International Workshop on Languages and Compilers for Parallel Computing, 2013, pp. 252–271.
[28] J. Hu, Z. Wang, Q. Qiu, W. Xiao, D.J. Lilja, Sparse fast Fourier transform on GPUs and multi-core CPUs, in: 2012 IEEE 24th International Symposium on Computer Architecture and High Performance Computing, 2012, pp. 83–91.
[29] S. Wang, V.M. Patel, A. Petropulu, The robust sparse Fourier transform (RSFT) and its application in radar signal processing, IEEE Trans. Aerosp. Electron. Syst. 53 (6) (2017) 2735–2755.
[30] B. Li, Z. Jiang, J. Chen, On performance of sparse fast Fourier transform algorithms using the flat window filter, IEEE Access 8 (2020) 79134–79146.
[31] G. Chen, S. Tsai, K. Yang, On performance of sparse fast Fourier transform and enhancement algorithm, IEEE Trans. Signal Process. 65 (21) (2017) 5716–5729.
[32] A. Rauh, G.R. Arce, Optimized spectrum permutation for the multidimensional sparse FFT, IEEE Trans. Signal Process. 65 (1) (Jan. 2017) 162–172.
[33] O. Abari, et al., 27.4 A 0.75-million-point Fourier-transform chip for frequency-sparse signals, in: 2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2014, pp. 458–459.
[34] A. Lopez-Parrado, J. Velasco-Medina, Cooperative wideband spectrum sensing based on sub-Nyquist sparse fast Fourier transform, IEEE Trans. Circuits Syst. II, Express Briefs 63 (1) (2016) 39–43.
[35] X. Pei, T. Shan, S. Liu, Y. Feng, An improved FPGA implementation of sparse fast Fourier transform, in: 6th International Conference on Advanced Materials and Computer Science, 2017.
[36] G. Plonka, K. Wannenwetsch, A.A.M. Cuyt, W. Lee, Deterministic sparse FFT for M-sparse vectors, Numer. Algorithms 78 (1) (2018) 133–159.
[37] S. Bittens, R. Zhang, M.A. Iwen, A deterministic sparse FFT for functions with structured Fourier sparsity, Adv. Comput. Math. 45 (2) (2019) 519–561.
[38] J. Schumacher, M. Puschel, High-performance sparse fast Fourier transforms, in: 2014 IEEE Workshop on Signal Processing Systems (SiPS), 2014, pp. 1–6.
[39] C. Wang, M. Araya-Polo, S. Chandrasekaran, A. St-Cyr, B. Chapman, D. Hohl, Parallel sparse FFT, in: Proceedings of the 3rd Workshop on Irregular Applications: Architectures and Algorithms, 2013, p. 10.
[40] C. Wang, S. Chandrasekaran, B. Chapman, cusFFT: a high-performance sparse fast Fourier transform algorithm on GPUs, in: 2016 IEEE International Parallel and Distributed Processing Symposium (IPDPS), 2016, pp. 963–972.
[41] J. Kung, G.-C. Rota, The theory of error-correcting codes: F.J. MacWilliams and N.J.A. Sloane, in two volumes, North-Holland, New York, 1977, 762 pp., Adv. Math. 32 (1) (1977) 81.
[42] G. Szegš, Orthogonal Polynomials, American Mathematical Society, 1939 [Online]. Available: https://books.google.com/books?id=RemVAwAAQBAJ.
[43] S.-H. Hsieh, C.-S. Lu, S.-C. Pei, Sparse fast Fourier transform for exactly and generally k-sparse signals by downsampling and sparse recovery, preprint, arXiv:1407.8315, 2014.
[44] W. Dai, O. Milenkovic, Subspace pursuit for compressive sensing signal reconstruction, IEEE Trans. Inf. Theory 55 (5) (2009) 2230–2249.
[45] A. López-Parrado, J.V. Medina, Efficient software implementation of the nearly optimal sparse fast Fourier transform for the noisy case, Ingeniería y Ciencia 11 (22) (2015) 73–94.
[46] A. López-Parrado, J. Velasco-Medina, SoC-FPGA implementation of the sparse fast Fourier transform algorithm, in: 2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS), Aug. 2017, pp. 120–123.
[47] D. Xu, S. Chen, F. Shen, G. Wang, Y. Zhang, A fast acquisition algorithm of GNSS receiver based on SFFT, in: 2017 Forum on Cooperative Positioning and Service (CPGPS), 2017, pp. 106–110.
[48] M. Pathmanathan, L. Peretti, Real-time signal frequency analysis in variable speed drives using the sparse fast Fourier transform (sFFT), in: 2017 IEEE International Conference on Industrial Technology (ICIT), Mar. 2017, pp. 1053–1058.
[49] B. Murmann, A/D converter trends: power dissipation, scaling and digitally assisted architectures, in: 2008 IEEE Custom Integrated Circuits Conference, Sep. 2008, pp. 105–112.
[50] Y.M. Greshishchev, et al., A 40GS/s 6b ADC in 65 nm CMOS, in: 2010 IEEE International Solid-State Circuits Conference - (ISSCC), Feb. 2010, pp. 390–391.
[51] O. Abari, F. Lim, F. Chen, V. Stojanovic, Why analog-to-information converters suffer in high-bandwidth sparse signal applications, IEEE Trans. Circuits Syst. 60 (9) (2013) 2273–2284.
[52] O. Abari, F. Chen, F. Lim, V. Stojanovic, Performance trade-offs and design limitations of analog-to-information converter front-ends, in: 2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2012, pp. 5309–5312.
[53] H. Kasem, M. Elsabrouty, Perceptual compressed sensing and perceptual sparse fast Fourier transform for audio signal compression, in: 2014 IEEE 15th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC), 2014, pp. 444–448.
[54] W. Bogner, et al., Real-time motion- and B0-correction for LASER-localized spiral-accelerated 3D-MRSI of the brain at 3T, NeuroImage 88 (2014) 22–31.
[55] D. Katabi, H. Hassanieh, P. Indyk, E. Price, M. Kapralov, L. Shi, SFFT: Sparse Fast Fourier Transform, https://groups.csail.mit.edu/netmit/sFFT/.
[56] CUFFT Library, CUFFT Library, https://docs.nvidia.com/cuda/pdf/CUFFT_Library.pdf.
[57] O. Artiles, F. Saeed, GPU-SFFT: a GPU based parallel algorithm for computing the Sparse Fast Fourier Transform (SFFT) of k-sparse signals, in: 2019 IEEE International Conference on Big Data (Big Data), 2019, pp. 3303–3311.
[58] H. Hassanieh, P. Indyk, D. Katabi, E. Price, Sparse fast Fourier transform code documentation (sfft 1.0 and 2.0), 2012.
[59] Q. Wu, Y. Xiang, Y. Chen, M. Tang, S. Fu, D. Liu, Sparse-fast-Fourier-transform-based quick synchronization for optical direct detection orthogonal frequency division multiplexing systems, Opt. Lett. 43 (9) (2018) 2014–2017.
[60] J. Lu, Z. Tan, A.P.T. Lau, S. Fu, M. Tang, C. Lu, Modulation format identification assisted by sparse-fast-Fourier-transform for hitless flexible coherent transceivers, Opt. Express 27 (5) (2019) 7072–7086.
[61] C.-Y. Lee, K.-Y. Hsu, C.-K. Lee, SFFT based phase demodulation for faster interference fringes analysis, in: Photonic Instrumentation Engineering III, vol. 9754, 2016, p. 975417.
[62] Hao Qun, Hu Yao, Ye Mingzhe, Digital moire interference phase real-time measurement method based on sparse Fourier transform, CN 110500968 A, Nov. 26, 2019 [Online]. Available: https://lens.org/189-394-417-926-589.

Elias Rajaby received his B.Sc from Electrical Engineering Department, Yazd University, Yazd, Iran, in 2014, and the M.Sc degree in Digital Electronic Engineering from Amirkabir University of Technology, Tehran, Iran, in 2016. He is currently pursuing the P.hD degree in Isfahan University of Technology, Isfahan, Iran. His research interests include digital system implementation, image processing, computer vision, and evolutionary algorithms.

Sayed Masoud Sayedi received the B.Sc. and M.Sc. degrees in electrical engineering from the Isfahan University of Technology (IUT), Isfahan, Iran, in 1986 and 1988, respectively, and the Ph.D. degree in electronics from Concordia University, Montreal, QC, Canada, in 1996. From 1988 to 1992, and then since 1997, he has been with IUT, where he is currently a Full Professor at the Department of Electrical and Computer Engineering. His current research interests include VLSI fabrication processes, image sensors, low-power VLSI circuits, and data converters.