IEEE Access
Multidisciplinary : Rapid Review : Open Access Journal

Received June 10, 2020, accepted July 28, 2020, date of publication August 3, 2020, date of current version August 20, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3013986

On the Low Complexity Implementation of the DFT-Based BFSK Demodulator for Ultra-Narrowband Communications

SIAVASH SAFAPOURHAJARI, (Member, IEEE), AND ANDRÉ B. J. KOKKELER, (Member, IEEE)
Faculty of Electrical Engineering, Mathematics and Computer Science, University of Twente, 7500 AE Enschede, The Netherlands
Corresponding author: Siavash Safapourhajari (s.safapourhajari@utwente.nl)

This work was supported by the Dutch Research Council (NWO) through the Slow Wireless Project under Project 13769.

ABSTRACT The DFT-based demodulator for BFSK has been introduced for applications where the received signal experiences a carrier frequency offset (CFO) much larger than the symbol rate. The Ultra-Narrowband (UNB) communication techniques have been introduced for implementing the emerging Low Power Wide Area Networks (LPWAN). Since UNB communication is prone to CFO, a DFT-based BFSK demodulator is an interesting option for this type of communication. However, for proper operation in a large frequency offset, the DFT-based demodulator requires a complex window synchronization which is not desirable for low power nodes. The main source of complexity, is calculating the DFT of a window which slides over the preamble. In this work, the complexity is alleviated by considering the window synchronization algorithm and its implementation together. First, a new window synchronization algorithm is proposed which is designed such that an efficient class of implementations of the sliding DFT (SDFT), called Single Bin SDFT (SB-SDFT) in this work, can be used. Moreover, a new stable implementation of SB-SDFT is designed to enable zero-padding which is required by the demodulator. The complexity of the proposed algorithm implemented using the SB-SDFT, scales more efficiently compared to the conventional algorithm when the range of tolerable CFO increases. Using the proposed method, for a CFO tolerance in the order of 14.5 times the symbol rate ($\pm$14.5 kHz for a symbol rate equal to 100 Hz), the number of complex operations is reduced by more than 85% (and memory by 90%) compared to the conventional method.

INDEX TERMS Frequency shift keying (FSK), frequency offset, sliding DFT, ultra-narrowband (UNB), offset tolerant demodulator.

I. INTRODUCTION

The concept of Low Power Wide Area Networks (LPWANs) has been recently introduced as an attractive technology to address a variety of IoT applications [1], [2]. LPWANs focus on low data rate applications which need a long range and favor a battery life as long as ten years or more. One of the proposed communication techniques for implementation of LPWAN is Ultra-Narrowband (UNB) communication [3], [4]. UNB offers very low data rate but its wide coverage, low cost devices, unlicensed band and robustness against interference makes it an attractive technology for LPWAN implementation [3], [5]. One of the challenges in a UNB communication system is Carrier Frequency Offset (CFO). It is particularly challenging in downlink communication as a low power node needs to receive a signal with frequency offset [6]. Considering the ultra-narrow bandwidth of the signal (as narrow as 100 Hz), using a low cost crystal without thermal compensation can lead to a CFO which is several times the symbol rate at the receiver [7].

CFO is a well-known problem in communication systems. It is either a consequence of mismatch between oscillators in the communication nodes or the Doppler shift resulting from their relative movement. Offset tolerant demodulators have been proposed in the last decades as a solution to this problem in low data rate satellite communications where the CFO is larger than the data rate [8], [9]. Such demodulators can also be used in UNB communications [6], [10], [11]. The ability of the demodulator to tolerate CFO relaxes (or even eliminates) the requirement for time and power consuming

The associate editor coordinating the review of this manuscript and approving it for publication was Muhammad Maaz Rehan.

carrier recovery as well as costly precise crystals with power hungry thermal compensation. Thus, a low power communication node for wireless sensor networks and IoT can be designed.

A DFT-based demodulator for BFSK is an example of aforementioned offset tolerant demodulators [8], [10]. To properly select the samples of a symbol for which the DFT is calculated, a window synchronization is required. To enable large offset tolerance for the demodulator, a window synchronization algorithm based on DFT is proposed by Hara *et al.* in [8] (also used in [10]). For tolerating large frequency offset using DFT-based demodulator, it must be implemented together with this window synchronization. Thus, the synchronization is discussed as a phase in the demodulator operation in this work. The main challenge of implementing this demodulator is the complexity of the synchronization algorithm which also increases when the tolerable frequency offset increases. In our previous work [12], a low complexity window synchronization algorithm was proposed and a Single Bin Sliding DFT (SB-SDFT) structure was introduced to efficiently implement the proposed algorithm.

This article elaborates on our previous design, mathematical formulation and related literature while, additionally, contributes to extending and analyzing it. The stability of the SB-SDFT proposed in [12] is considered and a stabilized version (using a damping factor) is proposed. The complexity is analyzed when the new stabilized SB-SDFT is utilized for implementing the proposed algorithm. In addition to the AWGN channel, which was considered in [12], in this work the BER performance in fading channel is also presented. Compared to our previous work, a more efficient implementation of the conventional synchronization algorithm is derived and used as the benchmark for complexity. This makes the complexity comparison between the proposed method and the conventional algorithm more realistic compared to our previous work. To increase the range of the tolerable frequency offset, the sampling frequency increases (a larger number of samples per symbol, N). In our previous work, BER performance and complexity results were presented only for $N = 8$; however, here, the BER performance and complexity are demonstrated for different values of N. This helps us to illustrate how the demodulator scales for larger values of N i.e. a larger tolerable frequency offset. Moreover, the design parameters, including the damping factor, are determined using simulations.

The paper is outlined as follows. The DFT-based demodulator and the conventional window synchronization algorithm (Hara synchronization) are briefly explained in the next section to properly define the problem and show how we can interpret the calculations of the algorithm as a sliding DFT. Section III looks into related work on the efficient implementation of a sliding DFT while jointly motivating the design of the proposed synchronization algorithm and the proposed SB-SDFT. Subsequently, the proposed synchronization algorithm is presented in Section IV. Next, section V elaborates on the proposed SB-SDFT and derives its stabilized version. The complexity analysis follows in Section VI to obtain expressions for the number of operations and memory use. Numerical results and discussions are included in section VII. Finally, section VIII concludes the paper.

II. BACKGROUND AND PROBLEM DEFINITION

In this section, first, the principle of the DFT-based demodulator in [10] is explicated to familiarize the reader with the base of this work. Then, the window synchronization algorithm introduced in [8] (and also used in [10]) is described to clarify the algorithm which is used as a benchmark with respect to performance and complexity. In the last subsection, a primary overview of the complexity of the demodulator is provided. It is shown that the synchronization complexity is dominant and should be alleviated.

A. THE PRINCIPLE OF THE DEMODULATOR

The block diagram of the DFT-based demodulator in [10] is shown in Fig. 1. Signal samples pass through a low pass filter prior to the demodulator. The set of complex samples belonging to a symbol are selected by a rectangular window which needs to be synchronized (aligned with symbols). When these samples are selected by a correctly synchronized window, they are padded with zeros and sent to the DFT calculation block. The zero-padding factor is defined as I which means N samples of the signal are padded by $N(I-1)$ zeros to achieve a zero-padded sequence with length NI as the input of the DFT. Finally, the decision for each BFSK symbol is made based on the magnitudes of the DFT for k_0 and k_1 which are the DFT bins corresponding to the frequencies of symbol zero (f_0) and one (f_1) of BFSK modulation, respectively.

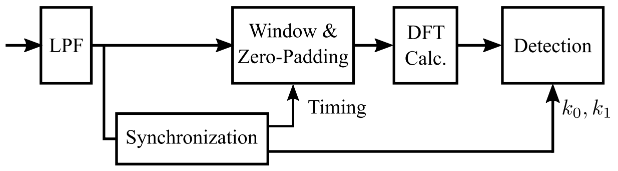

FIGURE 1. Baseband block diagram of the DFT-based demodulator.

Zero-padding is used to improve bin resolution of the DFT as explained in [8]. In [8] and [10], $I = 8$ is chosen because increasing it further adds complexity but does not improve the BER performance. The frequency separation of the BFSK modulation is assumed to be equal to the symbol rate ($f_1 = f_c + 1/2T$ and $f_0 = f_c - 1/2T$ where T is the symbol period and f_c is carrier frequency for passband signal and CFO for the baseband signal). In this work we also consider $I = 8$ and a frequency separation equal to the symbol rate.

To tolerate a large frequency offset, the lowpass filter in Fig. 1 might be much wider than the signal bandwidth. Moreover, complying with the Nyquist criterion (with respect to the filter bandwidth) necessitates a higher sampling frequency. The sampling frequency is assumed to be N times the symbol rate R_{Sym} (N complex samples per symbol).

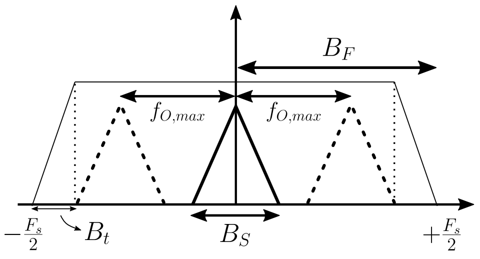

FIGURE 2. The relation between the filter bandwidth, the tolerable frequency offset and the sampling frequency.

To increase the offset tolerance, the sampling frequency and the number of samples per symbol, N, must be increased. The relation between the filter bandwidth, the sampling frequency, the signal bandwidth and the tolerable frequency offset are shown in Fig. 2. B_t is the transition band of the filter and B_F is the filter bandwidth including the transition band. The sampling frequency (for complex samples) is set to $F_s = 2B_F$ to avoid aliasing and noise folding. The bandwidth of the baseband signal is $B_S/2$ (where B_S is the bandwidth of the passband signal) and the maximum tolerable frequency offset is shown $\pm f_{O,max}$.

Two phases can be distinguished in the demodulator operation; synchronization and detection. In the *synchronization phase*, the window must be aligned to the received symbols; otherwise, frequency components of the adjacent symbols introduce inter-symbol interference. Furthermore, the DFT frequency bins corresponding to the frequency of symbol one (k_1) and zero (k_0) of the BFSK modulation are determined in this phase. In the *detection phase*, the DFT is calculated for samples of each symbol. For each symbol the decision is made by comparing the magnitude of the DFT bins k_1 and k_0.

B. THE WINDOW SYNCHRONIZATION ALGORITHM

The window synchronization algorithm described here has been introduced by Hara *et al.* [8] and, in the rest of this work, is referred to as *Hara synchronization* or conventional synchronization algorithm. This algorithm uses a preamble of alternating ones and zeros $(1, 0, 1, 0, \ldots)$. The number of symbols in the preamble is denoted by L. For each symbol, windows with different delay values, between 0 and $N-1$, are considered. Note that, because of oversampling, each symbol consists of N samples. Only one of these windows is fully aligned with a symbol and others include samples of two consecutive symbols. Fig. 3 illustrates the first three symbols of the preamble and the windows in case of four samples per symbol (each square is a sample). The double-headed horizontal arrows denote windows. Each row of arrows corresponds to a certain delay value as shown in the figure. Solid (dotted) arrows show the windows for an Odd (Even) symbol with different delay values. The symbols of the preamble are called Even and Odd depending on their index $m = 1, \ldots L$. Fig. 3 also shows how the magnitude of the DFT changes for different delays. When the DFT for each window is calculated, for each delay value,

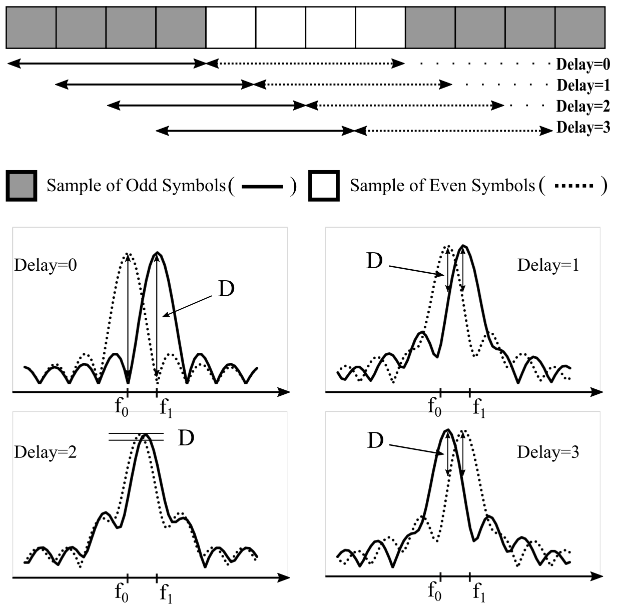

FIGURE 3. Three symbols of a preamble starting with symbol one, the windows of different delay values for the first two symbols and an example of spectrum variation when there are 4 samples per symbol. Solid and dotted lines show windows and spectra of odd and even symbols (which in this figure are considered to be 1 and 0), respectively.

all DFT magnitudes corresponding to Odd symbols (Even symbols) in the preamble are added together to achieve the following.

$$S_i^O(k) = \sum_{n=1}^{L/2} |X_{k,2n-1}^i|^2 \tag{1}$$

$$S_i^E(k) = \sum_{n=1}^{L/2} |X_{k,2n}^i|^2 \tag{2}$$

where $X_{k,2n}^i$ ($X_{k,2n-1}^i$) is the DFT value for the k^{th} bin and symbol $2n$ ($2n-1$) with a window delay i. Considering Fig. 3, (1) and (2) actually calculate the sum of the DFT magnitudes for solid windows within one row of arrows and the sum of those for dotted windows, respectively.

To synchronize the window, the Hara algorithm finds the delay value for which R_i in the following equation is maximized [8], [10].

$$R_i = [S_i^E(k_i^E) - S_i^E(k_i^O)] + [S_i^O(k_i^O) - S_i^O(k_i^E)], \tag{3}$$

where k_i^E and k_i^O are the bins with maximum magnitude in S_i^E and S_i^O, respectively. Finding the maximum of (3) is simply finding the delay value which maximizes the difference shown by D in Fig. 3 which in the specific case of Fig. 3 is for Delay $= 0$. When the delay value is found, the k_1 and k_0 can be achieved from k_i^O and k_i^E, respectively.

C. AN OVERVIEW OF COMPLEXITY

In the synchronization phase, all bins of an NI-point DFT are calculated over all symbols of the preamble and N different windows for each symbol in the preamble. In the detection phase, only two bins of the DFT are required for each symbol. Calculating only two bins of the NI-point DFT using the definition of the DFT series requires $2(N-1)$ complex multiplications (notice that there are only N non-zero samples in the zero-padded set of samples). The synchronization is done once in a packet; however, the packet length is short in target applications (in the order of 200 symbols) [4], [6], [7] which makes the complexity of the synchronization a significant overhead. Moreover, the operations required for the synchronization phase are executed in a shorter period of time compared to the detection phase which leads to high instantaneous power consumption. Thus, it might be unsuitable for applications where the maximum instantaneous power is limited (such as energy scavenging applications). Thus, the complexity of the synchronization algorithm is dominant and needs to be alleviated.

In our previous work and the current paper, the DFT calculations required for the window synchronization are interpreted as calculating DFT for a window which slides over a sequence of samples (see the arrows in Fig. 3). DFT for a window sliding over a sequence can be implemented efficiently using sliding DFT algorithms. Thus, this interpretation enables us to achieve an efficient implementation. The next section elaborates on implementing the sliding DFT reviewing related literature and clarifies the motivation for the proposed window synchronization algorithm and the proposed SB-SDFT.

III. RELATED WORK ON SLIDING DFT

Calculating the DFT for a window which is sliding over a sequence is called Sliding DFT (SDFT). As a computationally intensive block, many researchers have investigated the efficient implementation of the SDFT. In case of a sliding window, only one sample (or a few samples) is (are) different between the current set of input samples of the DFT and the previous set. Exploiting this property, various methods have been proposed for efficient implementation of the DFT with a sliding window [13]–[29]. By reusing calculations done for each window, the DFT of the next window can be calculated in a more efficient way. For a better understanding, we categorize the methods into two general groups. The first group, Complete SDFT (C-SDFT), includes algorithms that calculate all the DFT bins for each window in a single structure; whereas, the second category, Single-Bin SDFT (SB-SDFT), covers algorithms which derive only a single bin of the DFT. In the next two subsections these categories are explained. Although the final design in this work belongs to the SB-SDFT category, review of the C-SDFT methods is necessary. It helps us to derive an efficient implementation of the Hara synchronization algorithm (the algorithm explained in the previous section) which is used as a benchmark for complexity comparison. In the last subsection, the conclusions that can be drawn from literature are pointed out. This subsection also describes how this insight provokes the design of the proposed algorithm and the proposed SB-SDFT.

A. COMPLETE SDFT (C-SDFT)

One of the initial examples of an SDFT has been introduced in [13]–[15]. Based on the conventional Decimation-in-Time FFT structure and storing calculated intermediate values, this method decreases the complexity of FFT calculation for the new coming sample from $\mathcal{O}(N \log_2 N)$ in the FFT to $\mathcal{O}(N)$ [15]. However, it increases the memory as it needs to store all the intermediate values. Interpreting the FFT structure as a prism, Wang *et al.* [16] proposed a method to calculate the SDFT. Although its complexity is more than that of the SFFT in [15], it can be implemented in parallel for faster calculation [17] which is not necessary in our target application as it involves very low data rates. Rewriting the sliding DFT definition and using time shift properties, Montoya *et al.* have proposed a sliding DFT method with almost 50% reduction in the number of complex multiplications compared to the SFFT [18]. This idea is also extended to a Radix-4 decomposition in [19] which achieves even more savings at the cost of limiting the DFT points to a power of four. Despite the complexity reduction achieved in [18], [19], these cannot be used when zero-padding is involved which is the case in our target application. A guaranteed stable recursive algorithm for calculation of all bins is proposed in [20] which is based on a single bin recursive structure from the same author in [21]. This method is only applicable in a hopping scenario where each window hops $N/4$ samples (N is the DFT size). Another recursive algorithm for C-SDFT is introduced in [22] utilizing the observer theory in the control systems. The algorithm calculates the C-SDFT while solving the stability problem associated with recursive SDFTs with less memory than what is required by the SFFT. However, it cannot work when zero-padding is used in the input sequence. Among the C-SDFT methods, the technique in [15] (the first mentioned method in this section) is the best one for implementing the Hara synchronization algorithm due to its simplicity and possibility of using zero-padding. This method is called SFFT in the sequel.

B. SINGLE BIN SDFT (SB-SDFT)

The second group of SDFT algorithms are those which focus on calculating a single bin of the DFT (SB-SDFT). The core idea of such systems is based on the shifting property of the Fourier transform [23]. To clarify, consider the N-point DFT for $\mathbb{X}_n = \{x(n-N+1), \ldots, x(n)\}$ as follows

$$X_k(n) = \sum_{i=0}^{N-1} x(n-N+i+1)W_N^{-ki}, \tag{4}$$

where $X_k(n)$ is the k^{th} DFT bin of $\mathbb{X}_n$ and $W_N = \exp(j2\pi/N)$. X_k^n can be written in terms of $X_k(n-1)$, which is the DFT for $\mathbb{X}_{n-1} = \{x(n-N), \ldots, x(n-1)\}$, as follows.

$$X_k(n)=X_k(n-1)W_N^k - x(n-N)W_N^k + x(n)W_N^{-(N-1)k} \tag{5}$$

Noticing that $W_N^{Nk} = 1$, (5) can be rewritten as:

$$X_k(n) = W_N^k(X_k(n-1) - x(n-N) + x(n)) \tag{6}$$

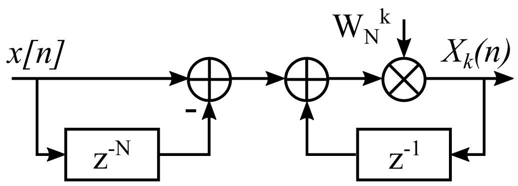

FIGURE 4. The simplest form of the SB-SDFT structure [23].

The block diagram of a filter which realizes (6) is depicted in Fig. 4. It calculates the k^{th} bin of an N-point DFT by implementing the DFT series for each bin as an IIR filter.

With each incoming sample (each filter iteration), the value of the k^{th} bin of an N-point DFT is updated for the last N samples. As a result, when the window hops one sample, only one complex multiplication and two complex additions are required to obtain the k^{th} bin for the new set of samples.

As can be seen, in the second stage there is a feedback loop with a pole on the unity circle of the complex plane. This pole makes the system conditionally stable. In real applications, the limited precision of the twiddle factors (W_N^k) can push the pole out of the unit circle and cause instability. In [23] a damping factor, r, is used to force the pole inside the unit circle while compromising the precision of the DFT calculation and causing errors. Nonetheless, if the damping factor is chosen close to one, the error can be kept small enough for many applications. Due to using the damping factor r, the method in [23] is also called rSDFT. To overcome the stability problem without compromising precision, a modulated SB-SDFT algorithm (modulated-SDFT) has been introduced in [25]. In this method the time-domain modulation duality of the Fourier Transform is used to shift any desired frequency component to the zero frequency. In this way, the pole in the feedback loop is equal to one and the filter will always be stable. Although both stability and precision are addressed in this method, the complexity increases considerably since the input samples need to be modulated with a proper twiddle factor.

C. CONCLUDING THE LITERATURE REVIEW

Considering previous work on the sliding DFT, one may conclude that C-SDFT methods are more efficient if all bins of the DFT are required; however, if a subset of the bins are needed, the complexity of an SB-SDFT is lower [29] (less than half of all bins when SB-SDFT is compared to SFFT). The Hara synchronization algorithm needs all bins of an NI-point DFT. Now, if a new algorithm is designed which only uses a subset of the bins belonging to an NI-point DFT, the complexity can be decreased using an SB-SDFT implementation. This is the incentive for designing the proposed window synchronization algorithm.

On the other hand, as mentioned in section II, zero-padding is necessary for correct detection in the demodulator. None of the mentioned SB-SDFT algorithms can be used together with zero-padding. Hence, for efficient implementation of the proposed algorithm a modified SB-SDFT is needed that incorporates zero-padding. In the next section, a synchronization algorithm is proposed which only needs a subset of the NI-point DFT bins. In section V, a stable SB-SDFT structure is derived for efficient implementation of the proposed algorithm.

IV. THE PROPOSED WINDOW SYNCHRONIZATION ALGORITHM

So far, it has been concluded that using only a subset of the bins of an NI-point DFT for window synchronization reduces complexity compared to the Hara synchronization method. Before introducing the proposed synchronization algorithm, it is needed to check whether it is feasible to only rely on a subset of bins for synchronization (without losing signal information). This is done in the first subsection where a set called *Bins of Interest* is defined and the general concept of the proposed synchronization algorithm is introduced. Afterwards, the algorithm is described in detail.

A. BINS OF INTEREST AND THE PROPOSED SYNCHRONIZATION CONCEPT

In the presence of large frequency offset (multiple times the symbol rate) a filter wider than (multiple times) the signal bandwidth is needed before the demodulator to capture the signal. It means that the sampling frequency should be higher than the actual information bearing bandwidth of the signal. On the other hand, using zero-padding increases the number of bins that must be calculated including those which are out of the signal bandwidth. As a result of this oversampling and zero-padding, only a small part of the spectrum calculated by an NI-point DFT includes signal information. These bins of the spectrum are called *Bins of Interest* hereafter and BoI for each DFT is defined as the set which includes these bins. Since the signal information resides in the BoI, solely the BoI can already provide a correct synchronization without loss of information. The BoI is in the vicinity of the signal center frequency, $f_c = (f_0 + f_1)/2 + CFO$ ($f_c = CFO$ in the baseband signal). This is shown in figure Fig. 5 for a case where the sampling frequency is 8 times the symbol rate and a zero-padding factor of 4 is used (32-point DFT). In Fig. 5,

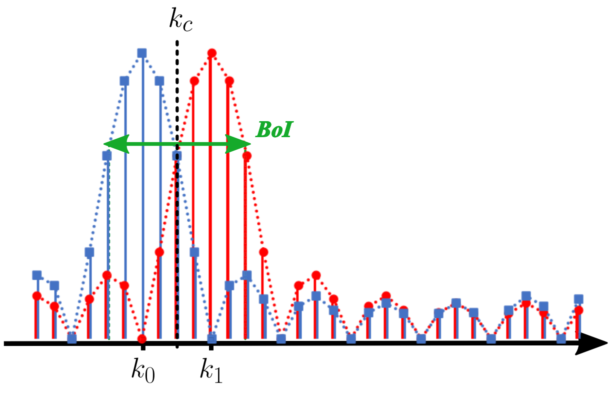

FIGURE 5. The spectrum of the signal and the BoI.

the k_0, k_1 and k_c correspond to the frequencies f_0, f_1 and f_c, respectively.

Since the frequency offset is assumed to be unknown, there is no prior knowledge about the whereabouts of the BoI in the spectrum. The proposed method aims first at finding the BoI to limit the number of the DFT bins required for the synchronization later on. Using a subset of the DFT bins, not only SB-SDFT algorithms can be used to decrease complexity but also fewer bins need to be stored during synchronization which considerably decreases memory requirements.

In the proposed synchronization concept an extra process is added to the synchronization, splitting it into two stages; the zoom stage and the window alignment stage. The concept of the proposed synchronization method is shown in Fig. 6. Input samples pass through a sliding rectangular window providing sequences of length N. First, in the zoom stage (explained in the next subsection) the BoI of an NI-point DFT called BoI_{Final}, is detected. Secondly, BoI_{Final} is sent to the window alignment stage where a proper window delay is obtained only calculating the DFT bins in BoI_{Final} and using the algorithm illustrated in Fig. 3. To compensate for the delay introduced by the zoom stage, an equivalent delay ($z^{-2N(\Gamma+1)}$) has been inserted before the Window Alignment block. It enables the algorithm to reuse the same samples used in the zoom stage for the window alignment. It avoids an increased number of preamble symbols for the proposed synchronization. In the next subsection, the zoom stage is explained in detail.

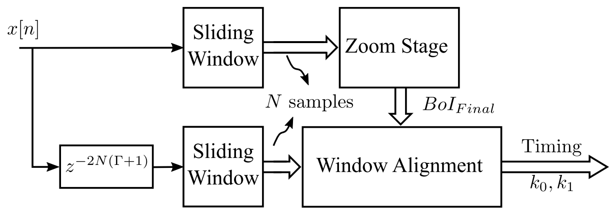

FIGURE 6. The block diagram of the proposed synchronization concept.

B. THE ZOOM STAGE

The target of this stage is to find the BoI of an NI-point DFT (BoI_{Final}) which covers the DFT bins corresponding to the BFSK modulation frequencies k_1 and k_0. For this purpose, a step-by-step zooming approach is utilized which searches for the center bin in an NI-point DFT, $k_{c,Final}$. The center bin is the bin closest to the center frequency (f_c) of the signal. The BoI_{Final} is calculated using the $k_{c,Final}$. Before elaborating on the step-by-step zooming algorithm, parameters are defined. T is the symbol period and N is the number of samples per symbol. $I = 2^{\Gamma}$ is the zero-padding factor. As mentioned earlier, the zero-padding factor for the DFT-based demodulator is selected to be $I = 8$, yet, for the proposed algorithm it can be any power of two, 2^{Γ}. Moreover, the frequency separation of the BFSK modulation is equal to the symbol rate ($f_1 = f_c + 1/2T$ and $f_0 = f_c - 1/2T$) similar to [8], [10]. The zoom stage includes $\Gamma + 1$ steps while the zero-padding

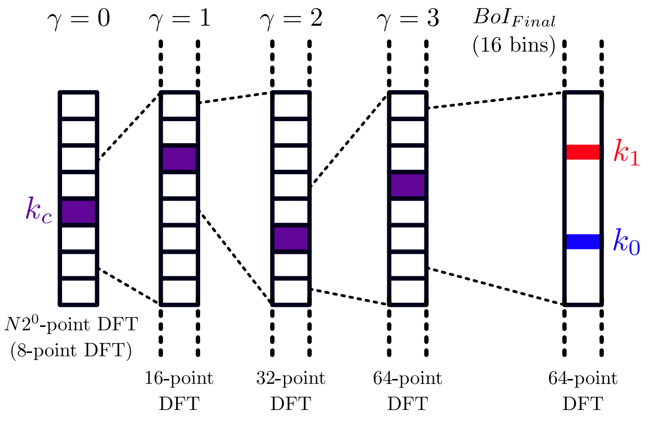

FIGURE 7. A visual description of step-by-step zooming for $N = 8$ and $I = 8$ (or $\Gamma = 3$).

factor at step γ is equal to 2^{γ} ($\gamma = 0, \ldots, \Gamma$). A visual illustration of the step-by-step zooming is shown in Fig. 7. In this figure $N = 8$ and $I = 8$ which is equivalent to $\Gamma = 3$. At each step γ, the bin k_c^{γ} of an $N2^{\gamma}$-point DFT which is closest to f_c is detected based on calculating only the bins within BoI_{γ}. Then, an estimate of the next step center bin, $\hat{k}_c^{\gamma+1}$, is calculated using k_c^{γ}. Following on that, $\hat{k}_c^{\gamma+1}$ is exploited to determine the BoI for step $\gamma + 1$, $BoI_{\gamma+1}$. Subsequently, in step $\gamma + 1$, the actual center bin $k_c^{\gamma+1}$ is detected by calculating only a subset of the bins of an $N2^{\gamma+1}$-point DFT in $BoI_{\gamma+1}$. Continuing this procedure, the center frequency of an NI-point DFT is obtained which is used to find BoI_{Final}.

The zoom stage starts with the first step, $\gamma = 0$, which uses an N-point DFT i.e. with a zero-padding factor of one (or no zeros). The BoI for the first step ($\gamma = 0$) includes all the bins of the N-point DFT. All points of the N-point DFT are required in the very first step because the center frequency can be anywhere in the spectrum that can be covered by the sampling frequency. Thus, we need to look at all bins at the first step and then we can limit ourselves to the BoI_{γ} calculated for further steps. Methods for calculating k_c^{γ}, calculating $\hat{k}_c^{\gamma+1}$ from k_c^{γ} and determining BoI are explicated as follows.

1) CALCULATING k_c^{γ}

At step γ, a window of length N slides for $2N$ hops (in shifts of one sample) over the the preamble (this is equal to the number of samples for two symbols) and the DFT is calculated for each window. For each one-sample shift of the window, only the bins of an $N2^{\gamma}$-point DFT inside BoI_{γ} are calculated. So there will be $2N$ DFTs for which the bin with maximum magnitude is changing from f_1 to f_0. An example of the windows for $N = 4$ and how the spectrum changes can be seen in Fig. 3.

The magnitudes of all these DFTs for each bin are added. Then, the bin for which this sum is maximum is closest to the center frequency k_c^{γ}. Thus, k_c^{γ} is obtained as follows.

$$k_c^{\gamma} = \max_{k \in BOI_{\gamma}} \mathcal{X}_k^{\gamma}, \tag{7}$$

where:

$$\mathcal{X}_k^{\gamma} = \sum_{i=0}^{2N-1} |X_{k,i}^{\gamma}|^2 \tag{8}$$

$X_{k,i}^{\gamma}$ is the DFT for the k^{th} bin and delay $i = 0, \ldots, 2N-1$ in step γ. The first window can start from any part of either a ''one symbol'' or a ''zero symbol'' in the preamble. Proving why the bin achieved using this method is actually the center bin is dealt with in Appendix.

2) CALCULATING $\hat{k}_c^{\gamma+1}$ FROM k_c^{γ}

The zero-padding factor is doubled between two consecutive steps. It means that zeros are added so that the length of the sequence over which DFT is calculated (including zeros) becomes twice of its value in the previous stage. This actually adds a new ''interpolated'' bin between each two bins of an $N2^{\gamma}$-point DFT to achieve an $N2^{\gamma+1}$-point DFT. Therefore, the center bin in step $\gamma+1$ can be estimated as $\hat{k}_c^{\gamma+1} = 2k_c^{\gamma}$. Notice that f_c might not be matched to a bin due to the arbitrary CFO. In this case, two adjacent bins close to the f_c have the largest magnitudes among all (the magnitudes are exactly the same if f_c is exactly in the middle of the two bins and there is no noise). In such cases, a noisy received signal and leakage causes the calculated $\hat{k}_c^{\gamma+1}$ to deviate from the actual center bin in step $\gamma+1$. That is why the center frequency should be detected step-by-step so that the final center frequency bin in the NI-point DFT is selected correctly. $\hat{k}_c^{\gamma+1}$ is used to determine $BoI_{\gamma+1}$ as follows.

3) DETERMINING $BoI_{\gamma+1}$ AND BoI_{Final}

To account for any erroneous detection due to the spectral leakage and noise, I bins (equal to the zero-padding factor) in the vicinity of the estimated center bin are considered as follows.

$$BoI_{\gamma+1} = \{k | k \in [min(a,b), max(a,b)]\}, \tag{9}$$

where:

$$a = (\hat{k}_c^{\gamma+1} - I/2) \mod (N2^{\gamma+1}) \tag{10}$$
$$b = (\hat{k}_c^{\gamma+1} + I/2 - 1) \mod (N2^{\gamma+1}) \tag{11}$$

In (10) and (11), $\mod$ is modulo operator. It is used to map values that are outside the range of bin numbers of an $N2^{\gamma+1}$-point DFT in step $\gamma+1$ to the valid set i.e. $k \in \{0, \ldots, N2^{\gamma+1}-1\}$.

In the final step, when the k_c^{Final} is calculated, the final BoI used for the window alignment stage (BoI_{Final}) is determined based on that. BoI_{Final} should include the frequency bins corresponding to symbols 1 and 0 of the BFSK modulated signal (k_1 and k_0, respectively). Since the frequency separation of the BFSK modulation is equal to the symbol rate (R_{Sym}), when the zero-padding factor is I and the sampling frequency is NR_{Sym}, there are $I-1$ bins between k_1 and k_0. To reduce the effect of noise, more than I bins are considered.

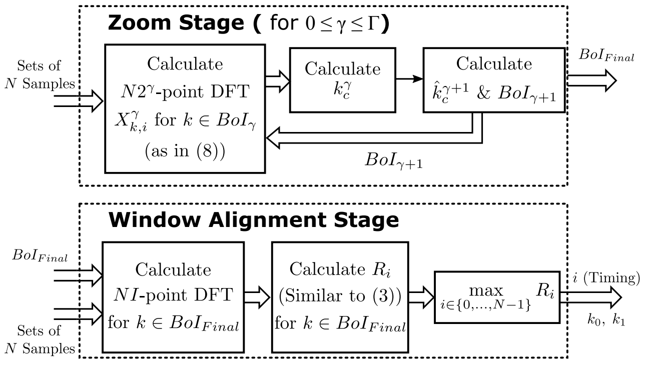
Zoom Stage (for $0 \le \gamma \le \Gamma$)
Sets of N Samples
Calculate $N2^{\gamma}$-point DFT $X_{k,i}^{\gamma}$ for $k \in BoI_{\gamma}$ (as in (8))
Calculate k_c^{γ}
Calculate $\hat{k}_c^{\gamma+1}$ & $BoI_{\gamma+1}$
BoI_{Final}
$BoI_{\gamma+1}$
Window Alignment Stage
BoI_{Final}
Sets of N Samples
Calculate NI-point DFT for $k \in BoI_{Final}$
Calculate R_i (Similar to (3)) for $k \in BoI_{Final}$
$\max_{i\in\{0,\ldots,N-1\}} R_i$
i (Timing)
k_0, k_1

FIGURE 8. The block diagram of the proposed synchronization algorithm; BoI stands for Bins of Interest.

If the number of bins in the BoI_{Final} (its cardinality) is denoted by $|BoI_{Final}|$, the BoI_{Final} is determined as follows.

$$BoI_{Final} = \{k | k \in [min(d,e), max(d,e)]\}, \tag{12}$$

where:

$$d = (k_{c,Final} - |BoI_{Final}|/2) \mod (NI) \tag{13}$$
$$e = (k_{c,Final} + |BoI_{Final}|/2 - 1) \mod (NI) \tag{14}$$

The size of BoI_{Final} is optimized to achieve the best BER performance using simulations which are presented in section VII.

A detailed block diagram of the proposed synchronization algorithm is also shown in Fig. 8. Both the zoom stage and the window alignment stage receive a set of N samples from a sliding window as shown in Fig. 6. The sets are the result of a rectangular window which slides over the preamble by one-sample shifts. The zoom stage is composed of three blocks. The first block calculates the DFT, the second calculates k_c^{γ} and the third calculates $\hat{k}_c^{\gamma+1}$ and $BoI_{\gamma+1}$. The BoI_{Final} is the output of the zoom stage which is sent to the window alignment stage. The window alignment stage is similar to the algorithm explained in section II. Getting samples of windows with different delays over the preamble, the DFT calculation block calculates the DFT magnitudes for bins in BoI_{Final}. These values are passed to the next block which calculates R_i (where $0 \le i \le N-1$ are the window delays) similar to (3) but only for the bins in BoI_{Final}. Finally, the third block finds the i value for which R_i is maximum and finds k_0 and k_1 in the BoI_{final}. This will be the proper window delay (the timing information) and, together with k_0 and k_1, it is used during the detection phase. For the DFT calculation blocks in the zoom and the window alignment stages, an SB-SDFT calculator can be used which is introduced in the next section.

V. THE PROPOSED SB-SDFT WITH ZERO-PADDING

In the previous section, a synchronization algorithm was presented. The second issue mentioned in the conclusion of the literature review for efficient implementation of the SDFT is an SB-SDFT scheme in the presence of zero-padding. Aforementioned implementations for an SB-SDFT (see section III) do not take the zero-padding into account. The main derivation of these algorithms is based on the periodicity of the twiddle factors. When each set of the samples is padded with zeros before the DFT calculation, this periodicity is violated due to zeros appended to the samples sequence. This leads to incorrect calculation by SB-SDFT schemes. For the proposed synchronization algorithm, a new SB-SDFT algorithm is needed to incorporate the zero-padding. A procedure similar to the conventional SB-SDFT derivation in [23], [25] and [20] is followed.

The k^{th} bin of an M-point DFT over a set of N samples, $\mathbb{X}_n = \{x(n-N+1), \ldots, x(n)\}$, which are padded with $M-N$ zeros, is as follows.

$$X_k(n) = \sum_{i=0}^{N-1} x(n-N+i+1)W_M^{-ki}, \tag{15}$$

where $W_M = e^{j\frac{2\pi}{M}}$ and the last $M-N$ terms of sum are ignored as they are zero. The k^{th} DFT bin in (15) can be obtained using the k^{th} DFT bin for sample set $\mathbb{X}_{n-1} = \{x(n-N), \ldots, x(n-1)\}$ as follows.

$$X_k(n) = X_k(n-1)W_M^k - x(n-N)W_M^k + x(n)W_M^{-(N-1)k} \tag{16}$$

where $X_k(n-1)$ is the k^{th} DFT bin for $\mathbb{X}_{n-1}$. When there is no zero-padding, $N = M$ and $W_M^{-(N-1)k}$ in the last term of (16) can be simplified to W_M^k leading to the known SB-SDFT equation (see (6)). If zero-padding is used, this simplification is not valid anymore. This is the violation of the periodicity mentioned above and necessitates a modified version of the SB-SDFT. In case of zero-padding, $N \neq M$ and (16) can be written as follows.

$$X_k(n) = W_M^k(X_k(n-1) - x(n-N) + x(n)W_M^{-Nk}) \tag{17}$$

Equation (17) can be seen as a filter taking samples of x and generating the k^{th} DFT bin while sliding the window by one sample. The transfer function of the filter is:

$$H(z) = \frac{W_M^{-kN} - z^{-N}}{W_M^{-k} - z^{-1}} \tag{18}$$

The block diagram of such a filter is also depicted in Fig. 9. The SB-SDFT with zero-padding has an extra multiplication in the first loop compared to the SB-SDFT in Fig. 4. Similar to the conventional SB-SDFT in [23], it has a pole on the unity circle. The stability of this modified version of SB-SDFT can be guaranteed using a damping factor r (similar to the idea proposed in [23]). Another method is to extend this derivation to achieve a modified version of the modulated-SDFT introduced in [25]. The former compromises precision while the latter increases complexity almost twice. Using a damping factor causes an error in the calculated DFT values. As shown in [30], this error is in the order of 1% when r is close enough to one. In our proposed method the exact value of the DFT is not the target but the relative value of different bins is important. As a result, to avoid the complexity of the modulated-SDFT, the modified SB-SDFT method is stabilized by adding a damping factor. To change the pole from W_M^k to rW_M^k, the z in (18) is replaced with z/r. This leads to the following transfer function.

$$H(z) = \frac{W_M^{-kN} - r^N z^{-N}}{W_M^{-k} - rz^{-1}} \tag{19}$$

The block diagram of the stable filter is shown in Fig. 10. For each loop, one multiplication between a complex number and a real number is added which is composed of two real multiplications. The extra multiplications in the second loop can be integrated into the twiddle factor. To do so the nominator and the denominator of the transfer function are multiplied with r^{-1} to achieve:

$$H(z) = \frac{r^{-1}W_M^{-kN} - r^{N-1}z^{-N}}{r^{-1}W_M^{-k} - z^{-1}} \tag{20}$$

The final block diagram is shown in Fig. 11. Interestingly, the one extra multiplication compared to (18) which is in the first loop is independent of the bin number and can be shared between filters which are calculating different DFT bins. The effect of the damping factor on the demodulator performance and the proper value for it are investigated using simulations presented in section VII.

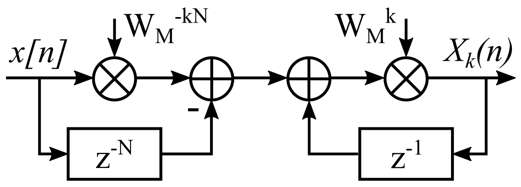

FIGURE 9. The modified SB-SDFT filter.

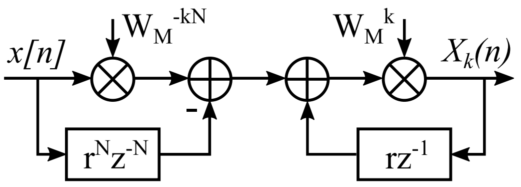

FIGURE 10. The modified SB-SDFT filter stabilized using r.

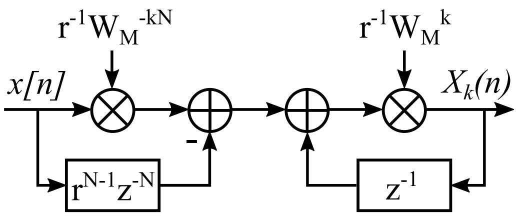

FIGURE 11. Modified and stabilized SB-SDFT filter with reduced multiplication.

VI. COMPLEXITY ANALYSIS

In the first subsection the number of complex multiplications/additions (CM/CA) required for DFT calculation are obtained. Next, the memory usage is calculated including the memory needed to compute the Sliding-DFT and the memory required to store S_i^O/S_i^E in (1)/(2) and the twiddle factors. In the final design, the zero-padding factor I is 8; however, in the following, parameter I is used for clarity. The proposed SB-SDFT is shown again for easier reference in Fig. 12. The dotted circles in Fig. 12 demonstrate different operations and/or memory required for the calculation of the SB-SDFT. Notice that the calculations shown in Fig. 12, are executed for each bin k of the DFT; however, when several bins are calculated the result of these operations/memory might be reused for multiple bins i.e. that part of the block can be shared between a few bins. This is further explained in the following analysis wherever applicable.

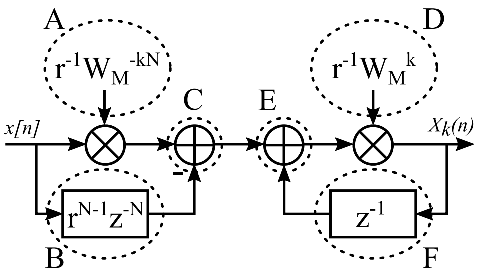

FIGURE 12. Modified and stabilized SB-SDFT filter with reduced multiplication.

A. COMPLEX OPERATIONS

The complexity of the proposed method is separately calculated for the zoom and the window alignment stage while explaining the operations shown by the dotted circles of Fig. 12. Each time that a new value for the DFT bin is calculated is called an iteration. Each iteration updates the DFT value based on the input sample and the last $N-1$ samples of the input sequence (in total N samples). In the following, multiplication and addition refer to complex operations unless stated otherwise.

1) ZOOM STAGE

A: The twiddle factor in A for the last step $\gamma = \Gamma$ is $r^{-1}W_{NI}^{-kN}$ for bin k which is written as $r^{-1}W_I^{-k}$. Since $W_I = \exp(j2\pi/I)$ is always a point on the unit circle within the complex plane, its powers have only I different values regardless of the number of calculated bins. Since $r^{-1}W_I^{-(k+I/2)} = -r^{-1}W_I^{-k}$, only half of the twiddle factor multiplications, $0.5I$, need to be calculated and the rest are simply sign conversions (multiplications with $+/-$ j are still considered as a complete complex multiplication). The last step of the zoom stage has the largest zero-padding factor and consequently, the largest number of different twiddle factors. So the number of multiplications within operation A is the largest in the last step. To simplify the complexity expressions we consider $0.5I$ multiplications for the other steps of the zoom stage as well. So A leads to $(0.5I)(\Gamma + 1)$ multiplications per iteration for all bins and all steps together.

B: The operation in B is a real-by-complex multiplication and can be approximated by 0.5 complex multiplication. It can be shared between all bins at each step. For all steps together, it leads to $0.5(\Gamma + 1)$ multiplications for each iteration and all bins together.

C: Based on above discussion, the output of A may have at most I different values in each step for all bins and B can be shared with all bins. As a result, C, leads to I additions (at most) per iteration in each step for all bins together. So C leads to $I(\Gamma + 1)$ additions per iteration for all bins and all steps together.

D/E: These parts together include one multiplication and one addition per bin and per iteration. In the zoom stage, at the first step, N bins are calculated and for the other Γ steps I bins are calculated ($N + I\Gamma$ in total). Thus, for the zoom stage, per iteration and for all bins together D and E lead to $N + I\Gamma$ multiplications and additions, respectively.

Iterations: Remember that for each step of the zoom stage the window shifts $2N$ samples (i.e. $2N$ iterations). Starting from the initial state of zero, N iterations are required to generate the DFT value for the first window of N samples. Then, each iteration gives the SB-SDFT for the next window. As a result, $3N$ iterations for each bin at each step of the zoom stage. Now, considering the above discussion and the number of iterations, the total number of multiplications and additions for the zoom stage (CM_{Zoom} and CA_{Zoom}, respectively) are obtained as follows.

$$CM_{Zoom} = 3N(0.5(I + 1)(\Gamma + 1) + (N + I\Gamma)) \quad (21)$$
$$CA_{Zoom} = 3N(I(\Gamma + 1) + (N + I\Gamma)) \quad (22)$$

2) WINDOW ALIGNMENT STAGE

A: In the window alignment stage, the zero-padding factor is I. Following the same reasoning as the zoom stage, A needs $0.5I$ multiplications per iteration for all bins together.

B: In total, this part needs two real multiplications or 0.5 complex multiplication per iteration for all bins together.

C: Similar to what was mentioned for the zoom stage, C leads to I complex additions per iteration for all bins together.

D/E: In the window alignment stage, $|BoI_{Final}|$ bins are calculated. Thus, per iteration and for all bins together D and E lead to $|BoI_{Final}|$ multiplications and additions, respectively.

Iterations: The window alignment stage needs NL iterations where L is the length of the preamble (N delays for each symbol of the preamble). Following the same reasoning for the zoom stage, N extra iterations are required when starting from the initial state of zero so the alignment stage needs $N(L + 1)$ iterations for each bin. The number of multiplications and additions for the window alignment stage (CM_{Align} and CA_{Align}, respectively) are:

$$CM_{Align} = N(L + 1)(0.5(I + 1) + |BoI_{Final}|) \quad (23)$$
$$CA_{Align} = N(L + 1)(I + |BoI_{Final}|) \quad (24)$$

B. MEMORY

Since the zoom stage and the window alignment stage are not executed in parallel, the registers (memory elements) used in one can be used in the other as well. Here, we only focus on the window alignment stage as it needs more memory and therefore, dictates the total required memory for calculating SDFT. For the Sliding DFT each filter needs a memory of size N for samples and a delay of one for the previous DFT value (See B and F in Fig. 11). The memory within B (z^{-N} with length N) can be shared between all bins and the second is unique for each bin. Additionally, the delay block in Fig. 6 requires storing $2N(\Gamma + 1)$ samples. All these values have two parts (real and imaginary), so the required memory for calculating SDFT (M_{Calc}) is as follows.

$$M_{Calc} = 2(N + |BoI_{Final}| + 2N(\Gamma + 1)) \tag{25}$$

where $|BoI_{Final}|$ is the cardinality of the set of BoI used in the window alignment stage. The proposed synchronization algorithm needs to store the results of the accumulation of the DFT magnitudes for $|BoI_{Final}|$ bins and N delay values. This memory is denoted by M_{Acc} and is derived as follows.

$$M_{Acc} = 2|BoI_{Final}|N \tag{26}$$

The factor two comes from the two sets of magnitudes that must be accumulated for odd and even symbols.

For an NI-point DFT, $W_{NI} = \exp(\frac{j2\pi}{NI})$ which means that W_{NI}^{-k} takes NI different values. Since $W_{NI}^{-(k+NI/2)} = -W_{NI}^{-k}$, only half of these values should be stored from which two values are equal to 1 and j for $k = 0$ and $k = NI/4$, respectively. As a result, $NI/2 - 2$ complex twiddle factors should be stored so the required memory is $NI - 4$.

C. COMPARISON

As mentioned in section III, the SFFT method in [15] provides an efficient implementation of the Hara synchronization algorithm. The complexity of the SFFT for each iteration can be found in [15]. Due to zero-padding, in a few first stages of the SFFT some operations have zero inputs and are not executed which leads to a pruned structure with reduced complexity. The number of iterations for the SFFT in the Hara synchronization algorithm can be derived based on a reasoning similar to what was used above for the window alignment stage. Taking these into account, the complexity of the Hara synchronization can be calculated. For the sake of brevity, the detailed calculations are not included in this article and only the final results are presented for comparison.

The number of operations and memory required for the Hara synchronization and the proposed synchronization algorithms are presented in TABLE 1. For the total number of operations only the term with the largest power of N is included in the table. Although the number of CM/CA increases with N^2 for both methods, in practical cases, the factor of N^2 in the Hara method is much larger than that of the proposed method. Considering $L = 16$ and $I = 8$ similar to both [8] and [10], the factor of N^2 in the number of complex multiplications required for the Hara synchronization (implemented using SFFT) is about 45 times the proposed synchronization. Furthermore, the total memory of the Hara method (with SFFT) increases with N^2 while the total memory of the proposed method increases with N. These differences between the complexity stem from the fact that in the proposed synchronization, in contrast with the Hara synchronization, the number of the calculated DFT bins for the window alignment does not change with N and is constant (BoI_{Final}). The numerical results and comparison of complexity for different values of N are presented in the next section.

VII. SIMULATION RESULTS AND DISCUSSION

A. DESIGN PARAMETERS

According to our discussions in the previous sections, two parameters need yet to be determined. Those are the number of bins in the BoI_{Final} and the damping factor r for the SB-SDFT. The most important performance metric for our system is the overall BER of the demodulator. Therefore, BER values obtained by simulations are utilized to determine the appropriate design parameters. As stated in IV, in all simulations of this section $I = 8$ and $L = 16$ similar to [8], [10]. Fig. 13 shows how the BER of the demodulator changes relative to $|BoI_{Final}|$ for different values of E_b/N_0. The damping factor is considered to be $r = 1$ in these simulations.

As can be seen in Fig. 13, the bit error rate value hardly changes when the number of bins in the BoI_{Final} is increased more than 14. For a safety margin the number of bins in the BoI_{Final} is considered to be 16.

The value of the damping factor determines the error at the output of the SB-SDFT filter. As mentioned earlier, the damping factor introduces errors to the calculated values for the DFT which may degrade the performance of the demodulator. This can be solved by choosing r very close to one.

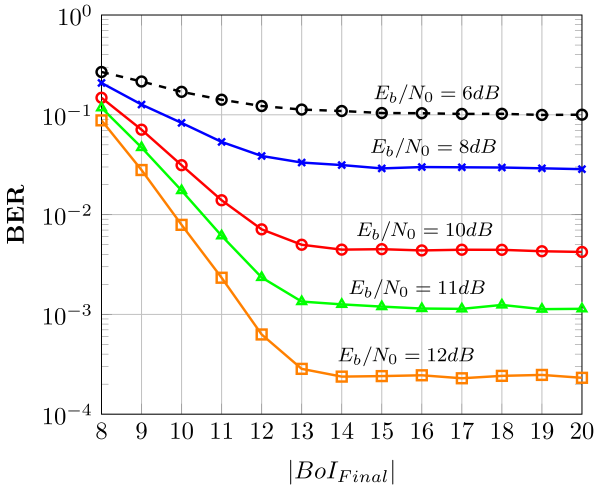

FIGURE 13. The BER values for the demodulator, using the proposed synchronization and the proposed SB-SDFT, for different sizes of the BoI_{Final} when $N = I = 8$, $L = 16$ and $r = 1$.

TABLE 1. Complexity comparison. *L* is the length of the preamble, *I* is the zero-padding factor, *N* is the number of samples per symbol and *BoI_{Final}* is the number of bins used in the window alignment stage of the proposed synchronization. For simulation results presented in section VII, *L* = 16, *I* = 8, Γ = 3 and *BoI_{Final}* = 16.

		Hara synchronization (with SFFT)	Proposed synchronization (with SB-SDFT)	
			Zoom	Align
DFT Calculation	CM	$I(N-1)N(L+1)$	$3N(0.5(I+1)(\Gamma+1)+N+I\Gamma)$	$N(L+1)(0.5(I+1)+\lvert BoI_{Final}\rvert)$
	CA	$2I(N-1)N(L+1)$	$3N(I(\Gamma+1)+N+I\Gamma)$	$N(L+1)(I+\lvert BoI_{Final}\rvert)$
Total Operations	**CM**	$\boldsymbol{I(L+1)N^2+\ldots}$	$\boldsymbol{3N^2+\ldots}$	
	CA	$\boldsymbol{2I(L+1)N^2+\ldots}$	$\boldsymbol{3N^2+\ldots}$	
Memory	Calc	$2NI\log_2 N-(4I-2)(N-1)$	$(4\Gamma+6)N+2\lvert BoI_{Final}\rvert$	
	Acc	$2IN^2$	$2\lvert BoI_{Final}\rvert N$	
	Twiddle	$NI-4$	$NI-4$	
Total Memory		$\boldsymbol{2IN(N+\log_2 N)+\ldots}$	$\boldsymbol{(I+4\Gamma+2\lvert BoI_{Final}\rvert+6)N+\ldots}$	

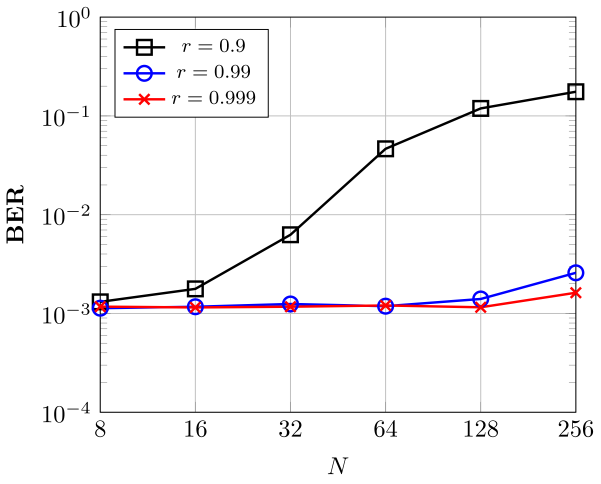

FIGURE 14. The BER of the demodulator with the proposed synchronization which is implemented using the proposed stable SB-SDFT. Eb/N_0 = 11 *dB*, *I* = 8 and *L* = 16.

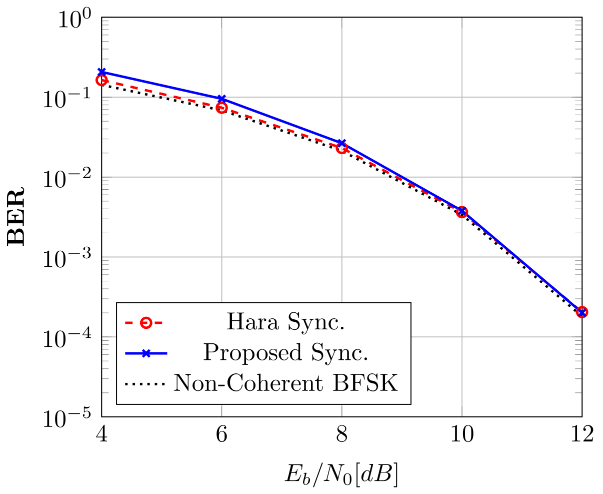

FIGURE 15. The BER curves for the demodulator in [10] with the proposed synchronization and Hara synchronization.

Additionally, the number of samples which pass through the SB-SDFT filter plays an important role in the value of r. Due to the recursive behavior of the filter, the error accumulation increases when an SB-SDFT is applied to a long sequence of samples and, to avoid that, r values closer to one might be required. Thus, for a constant preamble length, the appropriate value for r also depends on the number of samples per symbol. As explained in section II, N is proportional to the bandwidth of the filter before the demodulator so it determines the tolerable frequency offset. Fig. 14 illustrates the BER at $E_b/N_0 = 11\ dB$ (corresponding to $BER \approx 0.1\%$) for different values of N and r when $L = 16$ and $I = 8$. According to Fig. 14, $r = 0.999$ guarantees a consistent performance up to a sampling frequency more than 100 times the symbol rate.

B. BER PERFORMANCE

To evaluate the BER performance and compare it with the conventional algorithm (Hara synchronization), the proposed algorithm is applied to a DFT-based demodulator with parameters similar to [10]. The sampling frequency is $8R_{Sym}$ where R_{Sym} is the symbol rate while $I = 8$. The total system is simulated for an AWGN channel. It is assumed that the samples are the output of a brick-wall lowpass filter so the noise samples are uncorrelated. The BER performance of a DFT-based demodulator using the proposed synchronization and the Hara synchronization ([10]) algorithms is depicted in Fig. 15. As can be seen, the performance of the proposed method is only slightly worse at high BER values; however, for practical BER values in the order of 10^{-3} or smaller, it performs the same as the Hara method. The small increase in the BER at low SNR is due to a slight increase of error caused by using a small set of bins.

Fig. 16 illustrates the BER for a range of frequency offsets values up to twice the symbol rate and different E_b/N_0. It can be seen that the demodulator with the proposed synchronization can tolerate frequency offset as expected.

Fig. 17 shows the BER performance of the proposed method for different values of N and E_b/N_0. For these simulations $I = 8$, $L = 16$ and $r = 0.999$. It is seen that the increase of N does not affect the BER performance of the demodulator using the proposed synchronization algorithm and the proposed SB-SDFT. Notice that increasing N means that the bandwidth of the filter shown in Fig. 2 increases. This means that the noise bandwidth increases in our simulation.

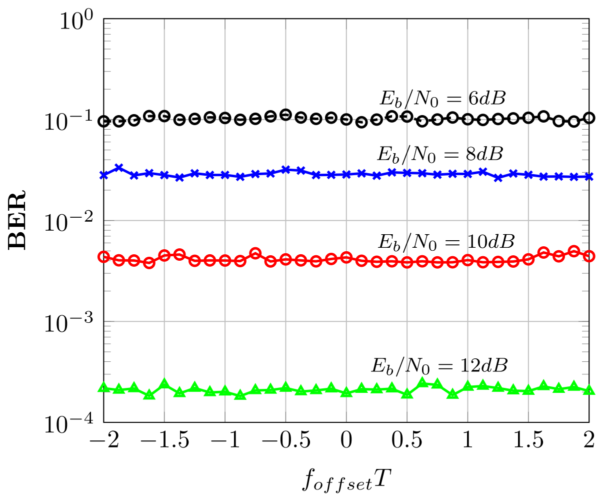

FIGURE 16. The BER of the demodulator with the proposed synchronization for different frequency offset normalized to the symbol rate ($f_{offset}T$).

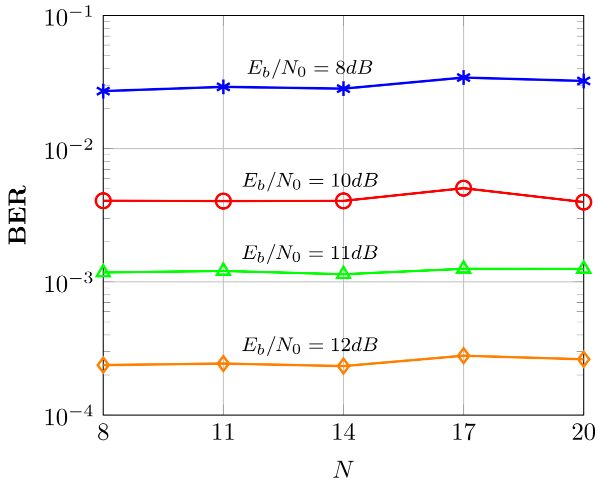

FIGURE 17. The BER of the demodulator with the proposed synchronization for different values of N and E_b/N_0 while $I = 8$, $L = 16$ and $r = 0.999$.

As can be seen in the results, this extra noise bandwidth does not degrades the BER performance. This can be justified by looking at the DFT as a bank of narrowband filters. What determines the detection performance is the bandwidth of these narrow filers and not the bandwidth of the wide filter before the demodulator [8].

Finally, Fig. 18 illustrates the performance of the demodulator with the proposed synchronization in Rayleigh and Rician fading ($K = 4$) channels. It can be seen that a demodulator using the proposed synchronization performs the same as a demodulator with Hara synchronization algorithm.

C. COMPLEXITY

The number of complex multiplications (CM), complex additions (CA) and memory (M) for the Hara method (with SFFT) and the proposed method using the proposed SB-SDFT are listed in TABLE 2. The values in TABLE 2 are calculated for $I = N = 8$ and $L = 16$ which are the same parameters

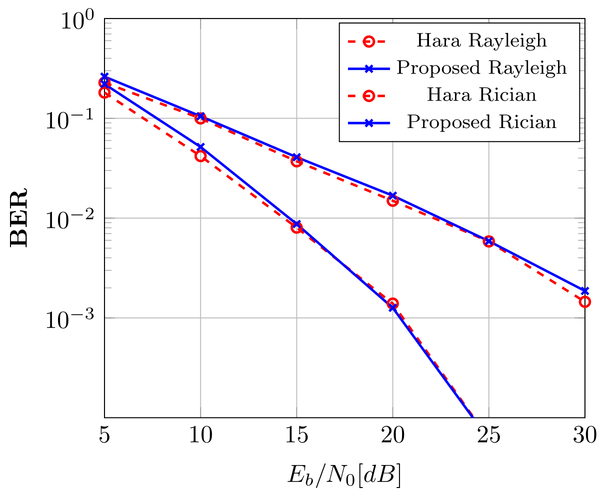

FIGURE 18. The BER curves for the DFT-based demodulator with the proposed synchronization and Hara synchronization in a Rayleigh and a Rician (K = 4) channel. In this simulations $N = 8$, $I = 8$, $L = 16$ and $r = 0.999$.

TABLE 2. Complexity comparison between Hara synchronization and the proposed synchronization when they are implemented using SFFT and the proposed SB-SDFT, respectively. $N = I = 8$, $L = 16$ and $BoI_{Final} = 16$.

	CM	CA	M
Hara Sync.	7616	15232	1258
Proposed Sync.	3988	4800	492
Improvement	**48%**	**68%**	**61%**

used for achieving the BER curves. Compared to the efficient implementation of the Hara algorithm, the proposed method reduces the number of complex additions, complex multiplications and memory by 68%, 48% and 61%, respectively.

Fig. 19 shows the number of complex operations and memory required for both methods and different values of N; while, Fig. 20 shows the improvement achieved using the proposed algorithm and SB-SDFT. The improvement is defined as the percentage of the reduced operations relative to the number of operations in the Hara method implemented using SFFT (e.g. $100\% \times (CM_{Hara} - CM_{Proposed})/CM_{Hara}$). Notice that the values of N are selected to be a power of two as required by the SFFT implementation of the Hara method; however, this is not necessary for the efficient implementation of the proposed algorithm. Both axes in Fig. 19 are on logarithmic scale.

Fig. 19 and Fig. 20 show that the difference between the complexity of the proposed method (with SB-SDFT) and the Hara algorithm (with SFFT) increases when N increases. One of the reasons is that only the number of DFT bins calculated in the first step of the zoom stage depends on N. For the next steps of the zoom stage and the alignment stage, the number of the calculated bins is constant for all N (I and $2I$, respectively). Although the number of samples involved in the DFT calculation is still related to N, the complexity grows slower as the number of bins required to be calculated

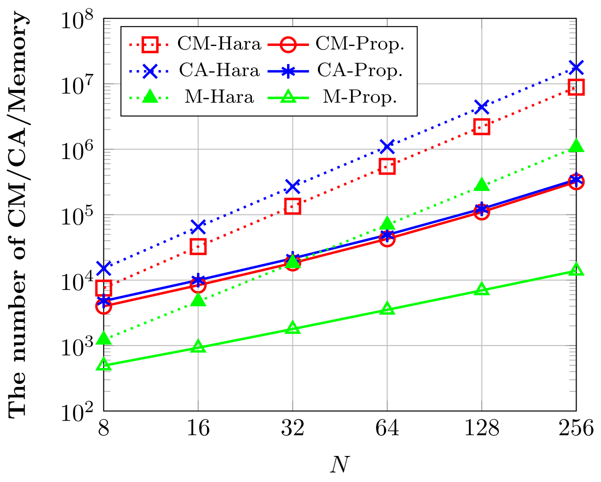

FIGURE 19. The complexity of Hara algorithm implemented using SFFT (shown by Hara) and the proposed algorithm implemented using the proposed SB-SDFT (shown by Prop.) in terms of complex additions (CA), complex multiplications (CM) and memory (M).

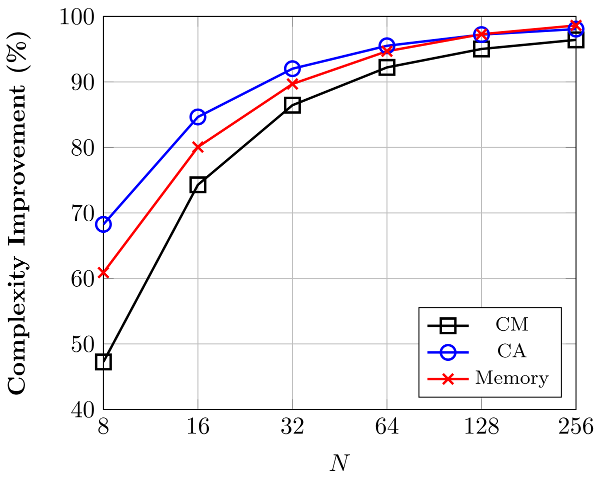

FIGURE 20. The improvement in complexity achieved by the proposed method implemented using the proposed SB-SDFT.

remains constant. That is why the difference between the complexity of the proposed synchronization and the Hara synchronization increases when N increases. For a sampling frequency which is 32 times of the symbol rate, almost 85% saving can be achieved in arithmetic operations (and around 90% in the memory). The null-to-null bandwidth of a BFSK modulated signal (B_S in Fig. 2) for a frequency separation of $f_{sep} = R_{Sym}$ is $3R_{Sym}$ [31]. If a very steep filter is considered ($B_t = 0$ in Fig. 2) such a sampling frequency means that a frequency offset around $\pm 14.5 R_{Sym}$ can be tolerated assuming that the main lobe should remain inside the filter (see Fig. 2). As shown in Fig. 15 and Fig. 18, the DFT-based demodulator which uses the proposed synchronization implemented using SB-SDFT has a BER performance similar to the DFT-based demodulator with the Hara synchronization; thus, the proposed synchronization and its efficient implementation reduce the complexity without any sacrifice in the performance.

VIII. CONCLUSION

The offset tolerant DFT-based demodulator is an interesting solution for low data rate applications including emerging ultra-narrowband communications for LPWANs. However, the existing synchronization algorithm for this demodulator is complex as it involves calculating the DFT of a sliding window. In this work, a new synchronization algorithm is proposed to decrease complexity. Using a step-by-step zooming technique, the proposed algorithm only requires a subset of the DFT bins. Consequently, efficient Single Bin Sliding DFT (SB-SDFT) implementations can be used. Besides, a stable SB-SDFT was introduced to incorporate zero-padding. The proposed algorithm and its implementation using the proposed SB-SDFT for an oversampling factor of e.g. $N = 8$, obtains 48%, 68%, 61% saving in the number of complex multiplications, complex additions and memory usage compared to the conventional window synchronization algorithm (Hara synchronization) while achieving the same BER performance. The relative reduction in the complexity further increases when larger N is required. The higher the sampling frequency (the larger value of N), the larger frequency offset can be tolerated. For a frequency offset tolerance about ± 14.5 times the symbol rate which requires an oversampling factor $N = 32$, the number of complex operations is reduced by more than 85% (and memory by 90%) compared to the conventional method (Hara synchronization implemented using SFFT).

To implement a low power receiver, the proposed algorithm should be combined with efficient digital design which requires research on the circuit level techniques. The effect of limited wordlength on the BER performance should also be studied. Additionally, it was shown that stabilizing the proposed SB-SDFT using a damping factor decreases the precision of the calculated DFT values. Further research is needed on designing a stable SB-SDFT with zero-padding for applications where high precision is required.

APPENDIX
MATHEMATICAL PROOF FOR k_c CALCULATION ALGORITHM

According to section IV, to achieve k_c^γ at each step, the magnitudes of DFTs for a sliding window with $2N$ one-sample shifts are added. Then, k_c^γ is the bin for which this sum has a maximum value (see (7) and (8)). In this appendix, it is mathematically shown that this sum over DFT magnitudes has a maximum at the center frequency. This is correct as long as all the $2N$ windows are in the preamble. The first window can start in any part of either a one symbol or a zero symbol in the preamble.

As the zero-padding factor changes during the zoom stage, (7) and (8) must be shown for the general case of all zero-padding factors in different steps of the zooming algorithm. For this purpose the concept of the Discrete Time Fourier Transform (DTFT) is used here. DTFT is a version of the Fourier Transform which maps discrete time samples to a continuous frequency domain. The DTFT of a discrete

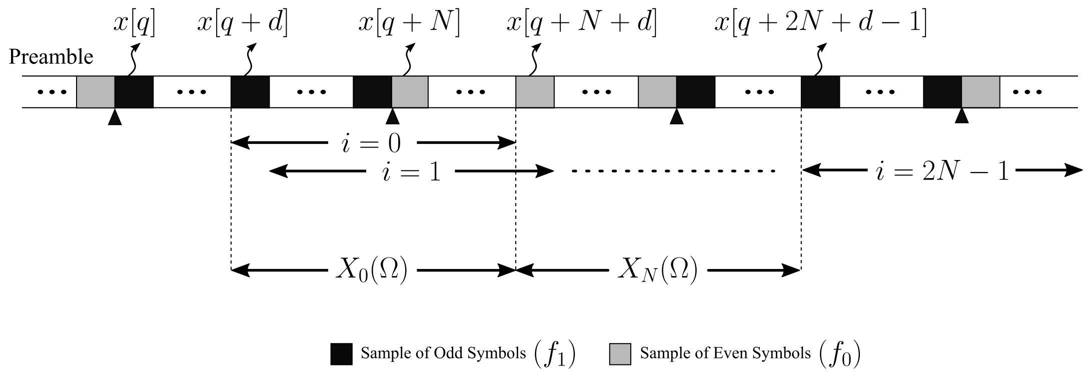

FIGURE 21. A part of the preamble and the windows used for calculating X_k^γ in the zoom stage. Three complete symbols (1, 0, 1) with a part of the symbol before the first one and a part of the symbol after the last one are shown. The boundaries of the symbols are shown by triangle markers. Black and grey squares show samples of Odd and Even symbols, respectively. Double-headed arrows are windows for which the DFT or the DTFT are calculated. $i = 0$ to $i = 2N-1$ are the windows used for the sum in (8). The two windows shown by $X_0(\Omega)$ and $X_N(\Omega)$ are windows used for the mathematical proof of the algorithm.

function in the time domain, shown by $x[n]$, is denoted by $X(\Omega)$ where $\Omega = 2\pi f/F_s$ $(-\pi < \Omega < \pi)$ is the frequency normalized to the sampling frequency, F_s. The k^{th} bin of the DFT calculated for N samples with zero-padding factor I, actually is the value $X(\Omega)$ of DTFT for effectively N samples, sampled at $f = (k/NI)F_s$ $(\Omega = 2\pi k/NI)$ [32]. Using the concept of the DTFT, (7) and (8) can be written as follows.

$$\Omega_c = \max_{\Omega\in[-\pi.+\pi]} \mathcal{X}(\Omega) \tag{27}$$

where:

$$\mathcal{X}(\Omega) = \sum_{i=0}^{2N-1} |X_i(\Omega)|^2 \tag{28}$$

$X_i(\Omega)$ is the DTFT for the i^{th} window and $\Omega_c = 2\pi f_c/F_s = \omega_c T_s$. Because (27) and (28) are generalized versions of (7) and (8), which are true for all zero-padding factors, the superscript γ from (8) is removed in (28). If it is proved that $\mathcal{X}(\Omega)$ has a maximum at Ω_c, then, the sum in (8) has a maximum at the bin closest to f_c (called k_c) for each zero-padding factor. This can be concluded considering that the DFT values for each k and window are actually samples of the DTFT.

A. CALCULATING THE COMPONENTS OF $\mathcal{X}(\Omega)$

The DTFT of a signal $x[n]$ over N samples is as follows.

$$X(\Omega) = \sum_{n=0}^{N-1} x[n]e^{-j\Omega n}, \quad \Omega = 2\pi f/F_s \tag{29}$$

Fig. 21 depicts a part of the preamble which is used for mathematical derivation. Black squares are the samples of an Odd symbol (f_1) and gray ones belong to an Even symbol (f_0). Each double-headed arrow stands for a window (set of samples) of length N which is used for calculating the DTFT. In general, during the zoom stage, the window is not synchronized yet. Thus, the set of sliding windows in (28) does not necessarily start from the beginning of a symbol. To account for this, it is assumed that the first window from the set of $2N$ windows starts from a window with delay d from the beginning of a symbol. This is shown with $i = 0$ in Fig. 21. The first sample of that symbol is shown by $x[q]$. The following procedure is executed considering windows $i = 0$ to $i = 2N-1$ and calculating the sum of the magnitudes of the DTFTs for these windows. The window $i = 0$ starts in an Odd symbol; nevertheless, the same mathematical formulation can be achieved if the black and gray samples are interchanged in Fig. 21 i.e. if $i = 0$ starts in an Even symbol. This generalization is possible because of the specific frequency separation for BFSK which is equal to the symbol rate.

Now, let us consider a pair of windows in the summation of (28) with delay d of the odd symbol (f_1) and the same delay for the next even symbol (f_0) (i.e. $i = 0$ and $i = N$ in (28)). In Fig. 21 these are shown by $X_0(\Omega)$ and $X_N(\Omega)$. If $G_0(\Omega)$ is defined as follows:

$$G_0(\Omega) = |X_0(\Omega)|^2 + |X_N(\Omega)|^2 \tag{30}$$

Then, $\mathcal{X}(\Omega)$ can be derived in terms of N similar pairs and different G_m as:

$$\mathcal{X}(\Omega) = \sum_{m=0}^{N-1} G_m(\Omega) \tag{31}$$

where $G_m(\Omega) = |X_m(\Omega)|^2 + |X_{m+N}(\Omega)|^2$. First, $G_0(\Omega)$ is analyzed. Considering BFSK modulation, the samples for the $i = 0$ and $i = N$ window are as follows.

$$\mathbf{x}_0[n] = \begin{cases} e^{j(\omega_1(n+d+q)T_s)} & 0 \le n < N-d \\ e^{j(\omega_0(n+d+q-N)T_s+\theta_1)} & N-d \le n < N \end{cases} \tag{32}$$

$$\mathbf{x}_N[n] = \begin{cases} e^{j(\omega_0(n+d+q)T_s+\theta_1)} & 0 \le n < N-d \\ e^{j(\omega_1(n+d+q-N)T_s+\theta_0+\theta_1)} & N-d \le n < N \end{cases} \tag{33}$$

where ω_1 and ω_0 are $2\pi f_1$ and $2\pi f_0$, respectively. In (32) and (33), $\theta_1 = \omega_1 NT_s$ and $\theta_0 = \omega_0 NT_s$ are actually accumulated phase at the end of an Odd symbol and Even symbol, respectively. Without loss of generality, it is assumed that the initial phase before $x[q]$ (in Fig. 21) is zero. Using $\mathbf{x}_0[n]$,

$\mathbf{x}_N[n]$ and the definition of the DTFT in (29), $X_0(\Omega)$ and $X_N(\Omega)$ are calculated.

$$X_0(\Omega) = \sum_{n=0}^{N-d-1} e^{j(\omega_1 T_s(n+d+q)-\Omega n)} + \sum_{n=N-d}^{N-1} e^{j(\omega_0 T_s(n+d+q)-\Omega n)} \tag{34}$$

$$X_N(\Omega) = \sum_{n=0}^{N-d-1} e^{j(\omega_0 T_s(n+d+q)+\omega_1 NT_s-\Omega n)} + \sum_{n=N-d}^{N-1} e^{j(\omega_1 T_s(n+d+q)+\omega_1 NT_s-\Omega n)} \tag{35}$$

Notice that in (32) and (33), $\theta_1 - \omega_0 NT_s$ and $\theta_0 - \omega_1 NT_s$ are equal to $+(\omega_1 - \omega_0)NT_s$ and $-(\omega_1 - \omega_0)NT_s$, respectively. Remember from section II that the frequency separation of BFSK is equal to the symbol rate $R_{Sym} = 1/T$ and $T_s/T = 1/N$. So $\pm(\omega_1 - \omega_0)NT_s$ are removed from the phase of the signals since they equal $\pm 2\pi$. Now that the primary expressions for $X_0(\Omega)$ and $X_N(\Omega)$ are obtained, a new variable is introduced to simplify the rest of the proof.

B. DEFINITION OF α

The main target is to prove that $\mathcal{X}(\Omega)$ has a maximum at $\Omega_c = \omega_c T_s$. Since the frequency separation of BFSK is equal to the symbol rate ($1/T = 1/NT_s$), $\omega_1 T_s$ and $\omega_0 T_s$ can be written in terms of center frequency ω_c as follows.

$$\omega_0 T_s = \omega_c T_s - \pi/N, \quad \omega_1 T_s = \omega_c T_s + \pi/N, \tag{36}$$

Now that $\omega_0 T_s$ and $\omega_1 T_s$ are written in the form of deviations from $\omega_c T_s$, let us define α as the deviation of Ω from Ω_c.

$$\alpha = \Omega - \omega_c T_s \tag{37}$$

In the following, the variable Ω is changed to $\omega_c T_s + \alpha$ and (30), (31), (34) and (35) are formulated as functions of α. These new functions are denoted by a tilde over their name (for instance $\mathcal{X}(\Omega)$ changes to $\tilde{\mathcal{X}}(\alpha)$). When all functions are formulated in terms of α, instead of proving that there is a maximum at $\Omega = \Omega_c$ for $\mathcal{X}(\Omega)$, it must be proved that $\tilde{\mathcal{X}}(\alpha)$ has a maximum at $\alpha = 0$. This simplifies the calculations, particularly, it enables us to eliminate the terms related to $\omega_c T_s$ which is seen in the expressions for $\omega_1 T_s$ and $\omega_0 T_s$ (36) as well.

By change of variable to α, (31) can be written as:

$$\tilde{\mathcal{X}}(\alpha) = \sum_{m=0}^{N-1} \tilde{G}_m(\alpha) \tag{38}$$

For now, we focus on $\tilde{G}_0(\alpha) = |\tilde{X}_0(\alpha)|^2 + |\tilde{X}_N(\alpha)|^2$. Substituting $\omega_1 T_s$ and $\omega_0 T_s$ from (36) to (34) and (35) while replacing Ω with $\omega_c T_s + \alpha$ we have:

$$\tilde{X}_0(\alpha) = e^{j(\omega_c T_s+\frac{\pi}{N})(d+q)} \sum_{n=0}^{N-d-1} e^{j(\frac{\pi}{N}-\alpha)n} + e^{j(\omega_c T_s-\frac{\pi}{N})(d+q)} \sum_{n=N-d}^{N-1} e^{-j(\frac{\pi}{N}+\alpha)n} \tag{39}$$

$$\tilde{X}_N(\alpha) = -e^{j[(\omega_c T_s-\frac{\pi}{N})(d+q)+\omega_c T]} \sum_{n=0}^{N-d-1} e^{j-(\frac{\pi}{N}+\alpha)n} - e^{j[(\omega_c T_s+\frac{\pi}{N})(d+q)+\omega_c T]} \sum_{n=N-d}^{N-1} e^{j(\frac{\pi}{N}-\alpha)n} \tag{40}$$

where $T = NT_s$ and the terms which are independent of n are taken out of summation. As a result of the change of variable, π appears in the phase of the exponential functions in (35) which is converted to a negative sign before the summations in (40). For a complex function $f(x)$, $|f(x)|^2 = f(x)f(x)^*$ where $f(x)^*$ is the complex conjugate of $f(x)$. Thus, the magnitudes of $\tilde{X}_0(\alpha)$ and $\tilde{X}_N(\alpha)$ are as follows.

$$|\tilde{X}_0(\alpha)|^2 = \sum_{n,p=0}^{N-d-1} e^{j(\frac{\pi}{N}-\alpha)(n-p)} + \sum_{n,p=0}^{d-1} e^{-j(\frac{\pi}{N}+\alpha)(n-p)} + 2\sum_{n=0}^{N-d-1}\sum_{p=N-d}^{N-1} \cos(\Theta(n,p)-\alpha(n-p)) \tag{41}$$

$$|\tilde{X}_N(\alpha)|^2 = \sum_{n,p=0}^{N-d-1} e^{-j(\frac{\pi}{N}+\alpha)(n-p)} + \sum_{n,p=0}^{d-1} e^{j(\frac{\pi}{N}-\alpha)(n-p)} + 2\sum_{n=0}^{N-d-1}\sum_{p=N-d}^{N-1} \cos(-\Theta(n,p)-\alpha(n-p)) \tag{42}$$

where $\Theta(n,p) = \frac{\pi}{N}(2(d+q)+n+p)$ and the following equality is used:

$$\sum_{m,k=0}^{M-1} a_m b_k = \sum_{m,k=0}^{M-1} a_k b_m = \left(\sum_{m=0}^{M-1} a_m\right)\left(\sum_{m=0}^{M-1} b_m\right) \tag{43}$$

The first summation of (42) and the second summation of (41) can be rewritten as follows.

$$A(\alpha) = \sum_{n,p=0}^{N-d-1} e^{-j(\frac{\pi}{N}+\alpha)(n-p)} = \sum_{n,p=0}^{N-d-1} e^{j(\frac{\pi}{N}+\alpha)(n-p)} \tag{44}$$

$$B(\alpha) = \sum_{n,p=0}^{d-1} e^{-j(\frac{\pi}{N}+\alpha)(n-p)} = \sum_{n,p=0}^{d-1} e^{j(\frac{\pi}{N}+\alpha)(n-p)} \tag{45}$$

To obtain (44) and (45), first the negative sign before j is dissolved to $(n-p)$ to make $(p-n)$; then, using a simple change of indexes in summation (see (43)), $(p-n)$ is converted to $(n-p)$. Also, using $\cos(x) = \cos(-x)$, the third term of $|\tilde{X}_N(\alpha)|^2$ can be written as follows.

$$C(\alpha) = 2\sum_{n=0}^{N-d-1}\sum_{p=N-d}^{N-1} \cos(\Theta(n,p)+\alpha(n-p)) \tag{46}$$

Taking (44)-(46) into account, $|\tilde{X}_0(\alpha)|^2$ and $|\tilde{X}_N(\alpha)|^2$ can be written based on $A(\alpha)$, $B(\alpha)$ and $C(\alpha)$.

$$|\tilde{X}_0(\alpha)|^2 = A(-\alpha) + B(\alpha) + C(-\alpha) \tag{47}$$

$$|\tilde{X}_N(\alpha)|^2 = A(\alpha) + B(-\alpha) + C(\alpha) \tag{48}$$

This simplified notation of $|X_0(\alpha)|^2$ and $|X_N(\alpha)|^2$ shapes the basis of the final stage of our proof.

C. PROVING $\alpha = 0$ IS MAXIMUM

For a function $f(x)$ to have a maximum at x_0, its first and second derivatives at x_0 must be zero and negative, respectively. Hence, in the following we prove:

- Statement I: $\frac{d}{d\alpha}\tilde{\mathcal{X}}(\alpha)|_{\alpha=0} = 0$
- Statement II: $\frac{d^2}{d\alpha^2}\tilde{\mathcal{X}}(\alpha)|_{\alpha=0} < 0$

From (47) and (48), $|\tilde{X}_0(\alpha)|^2$ and $|\tilde{X}_N(\alpha)|^2$ can be written as $F(\alpha)$ and $F(-\alpha)$, respectively. Besides, $\tilde{G}_0(\alpha)$ can be written as $F(\alpha)+F(-\alpha)$. It is known from the chain rule of derivative that $\frac{d}{dx}f(g(x)) = g'(x)f'(g(x))$. Thus, $\frac{d}{d\alpha}F(-\alpha) = -F'(-\alpha)$. So the first derivative of $\tilde{G}_0(\alpha)$ is as follows.

$$\frac{d}{d\alpha}\tilde{G}_0(\alpha) = F'(\alpha) - F'(-\alpha) \tag{49}$$

For $\alpha = 0$ the right side of (49) is zero. Notice that all derivations so far are based on a general case of d, and, as mentioned earlier, the same procedure can be followed if the $i = 0$ window starts from an Even symbol. The main reason that allows this is the frequency separation which is equal to the symbol rate. This means the result of (49) can be generalized to any $\tilde{G}_m(\alpha)$. Considering the definition of $\tilde{\mathcal{X}}(\alpha)$ from (38), statement I is proved as follows:

$$\frac{d}{d\alpha}\tilde{\mathcal{X}}(\alpha)|_{\alpha=0} = \sum_{m=0}^{N-1}\frac{d}{d\alpha}\tilde{G}_m(\alpha)|_{\alpha=0} = 0 \tag{50}$$

Statement I shows that $\alpha = 0$ is an extremum (maximum or minimum) for $\tilde{\mathcal{X}}(\alpha)$. To further prove that $\alpha = 0$ is a maximum (and not a minimum), it must be shown that its second derivative is negative. Taking another derivative from both sides of (49) according to the chain rule, the $\frac{d^2}{d\alpha^2}\tilde{G}_0(\alpha)$ is derived as follows.

$$\frac{d^2}{d\alpha^2}\tilde{G}_0(\alpha) = F''(\alpha) + F''(-\alpha) \tag{51}$$

Moreover, for any d, $|\tilde{X}_0(\alpha)|^2 = F(\alpha)$ and $|\tilde{X}_N(\alpha)|^2 = F(-\alpha)$ have a maximum in $-\pi/N \leq \alpha \leq \pi/N$ (see the spectra between f_0 and f_1 in Fig. 3) which means both have a negative second derivative in this interval including $\alpha = 0$. As a result, $\frac{d^2}{d\alpha^2}\tilde{G}_m(\alpha)$ is negative at $\alpha = 0$ so statement II holds as follows.

$$\frac{d^2}{d\alpha^2}\tilde{\mathcal{X}}(\alpha)|_{\alpha=0} = \sum_{m=0}^{N-1}\frac{d^2}{d\alpha^2}\tilde{G}_m(\alpha)|_{\alpha=0} < 0 \tag{52}$$

From statements I and II, it is concluded that the sum in (28) has a maximum at $\alpha = 0$ or $\Omega = \omega_c T_s$.

REFERENCES

[1] Q. M. Qadir, T. A. Rashid, N. K. Al-Salihi, B. Ismael, A. A. Kist, and Z. Zhang, "Low power wide area networks: A survey of enabling technologies, applications and interoperability needs," *IEEE Access*, vol. 6, pp. 77454–77473, 2018.

[2] M. Chen, Y. Miao, Y. Hao, and K. Hwang, "Narrow band Internet of Things," *IEEE Access*, vol. 5, pp. 20557–20577, 2017.

[3] K. Mekki, E. Bajic, F. Chaxel, and F. Meyer, "A comparative study of LPWAN technologies for large-scale IoT deployment," *ICT Exp.*, vol. 5, no. 1, pp. 1–7, Mar. 2019.

[4] *Sigfox*. Accessed: May 1, 2020. [Online]. Available: https://www.sigfox.com/en

[5] N. Naik, "LPWAN technologies for IoT systems: Choice between ultra narrow band and spread spectrum," in *Proc. IEEE Int. Syst. Eng. Symp. (ISSE)*, Oct. 2018, pp. 1–8.

[6] D. Lachartre, F. Dehmas, C. Bernier, C. Fourtet, L. Ouvry, F. Lepin, E. Mercier, S. Hamard, L. Zirphile, S. Thuries, and F. Chaix, "A TCXO-less 100 Hz-minimum-bandwidth transceiver for ultra-narrow-band sub-GHz IoT cellular networks," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2017, pp. 134–135.

[7] C. Goursaud and Y. Mo, "Random unslotted time-frequency ALOHA: Theory and application to IoT UNB networks," in *Proc. 23rd Int. Conf. Telecommun. (ICT)*, May 2016, pp. 1–5.

[8] S. Hara, A. Wannasarnmaytha, Y. Tsuchida, and N. Morinaga, "A novel FSK demodulation method using short-time DFT analysis for LEO satellite communication systems," *IEEE Trans. Veh. Technol.*, vol. 46, no. 3, pp. 625–633, Aug. 1997.

[9] M. R. Yuce and W. Liu, "A low-power multirate differential PSK receiver for space applications," *IEEE Trans. Veh. Technol.*, vol. 54, no. 6, pp. 2074–2084, Nov. 2005.

[10] S. Safapourhajari and A. B. J. Kokkeler, "Frequency offset tolerant demodulation for low data rate and narrowband wireless sensor node," in *Proc. 11th Int. Conf. Signal Process. Commun. Syst. (ICSPCS)*, Dec. 2017, pp. 1–8.

[11] S. Safapourhajari and A. B. J. Kokkeler, "Demodulation of double differential PSK in presence of large frequency offset and wide filter," in *Proc. IEEE 87th Veh. Technol. Conf. (VTC Spring)*, Jun. 2018, pp. 1–5.

[12] S. Safapourhajari and A. B. J. Kokkeler, "Low complexity synchronization for offset tolerant DFT-based BFSK demodulator," in *Proc. IEEE 20th Int. Workshop Signal Process. Adv. Wireless Commun. (SPAWC)*, Jul. 2019, pp. 1–5.

[13] M. Covell and J. Richardson, "A new, efficient structure for the short-time Fourier transform, with an application in code-division sonar imaging," in *Proc. ICASSP-Int. Conf. Acoust., Speech, Signal Process.*, vol. 3, Apr. 1991, pp. 2041–2044.

[14] B. Farhang-Boroujeny and Y. C. Lim, "A comment on the computational complexity of sliding FFT," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 39, no. 12, pp. 875–876, Dec. 1992.

[15] B. Farhang-Boroujeny and S. Gazor, "Generalized sliding FFT and its application to implementation of block LMS adaptive filters," *IEEE Trans. Signal Process.*, vol. 42, no. 3, pp. 532–538, Mar. 1994.

[16] J. Wang, B. Woods, and W. F. Eddy, "MEG, RFFTs, and the hunt for high frequency oscillations," in *Proc. 2nd Int. Congr. Image Signal Process.*, Oct. 2009, pp. 1–5.

[17] J.-M. Wang and W. F. Eddy, "Parallel algorithm for SWFFT using 3D data structure," *Circuits, Syst., Signal Process.*, vol. 31, no. 2, pp. 711–726, Apr. 2012.

[18] D. A. Montoya-Andrade, J. A. Rosendo-Macías, and A. Gómez-Expósito, "Efficient computation of the short-time DFT based on a modified radix-2 decimation-in-frequency algorithm," *Signal Process.*, vol. 92, no. 10, pp. 2525–2531, Oct. 2012.

[19] D.-E.-A. Montoya, J. A. R. Macías, and A. Gómez-Expósito, "Short-time DFT computation by a modified Radix-4 decimation-in-frequency algorithm," *Signal Process.*, vol. 94, pp. 81–89, Jan. 2014.

[20] C.-S. Park, "Fast, accurate, and guaranteed stable sliding discrete Fourier transform," *IEEE Signal Process. Mag.*, vol. 32, no. 4, pp. 145–156, Jul. 2015.

[21] C.-S. Park and S.-J. Ko, "The hopping discrete Fourier transform," *IEEE Signal Process. Mag.*, vol. 31, no. 2, pp. 135–139, Mar. 2014.

[22] Z. Kollar, F. Plesznik, and S. Trumpf, "Observer-based recursive sliding discrete Fourier transform [tips & tricks]," *IEEE Signal Process. Mag.*, vol. 35, no. 6, pp. 100–106, Nov. 2018.

[23] E. Jacobsen and R. Lyons, "The sliding DFT," *IEEE Signal Process. Mag.*, vol. 20, no. 2, pp. 74–80, Mar. 2003.

[24] S. C. Douglas and J. K. Soh, "A numerically-stable sliding-window estimator and its application to adaptive filters," in *Proc. Conf. Rec. 31st Asilomar Conf. Signals, Syst. Comput.*, vol. 1, Nov. 1997, pp. 111–115.
[25] K. Duda, "Accurate, guaranteed stable, sliding discrete Fourier transform [DSP tips & tricks]," *IEEE Signal Process. Mag.*, vol. 27, no. 6, pp. 124–127, Nov. 2010.
[26] D. A. Gudovskiy and L. Chu, "An accurate and stable sliding DFT computed by a modified CIC filter [tips & tricks]," *IEEE Signal Process. Mag.*, vol. 34, no. 1, pp. 89–93, Jan. 2017.
[27] Z. Rafii, "Sliding discrete Fourier transform with kernel windowing [lecture notes]," *IEEE Signal Process. Mag.*, vol. 35, no. 6, pp. 88–92, Nov. 2018.
[28] A. G. Exposito and J. A. R. Macfas, "Fast non-recursive computation of individual running harmonics," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 47, no. 8, pp. 779–782, Aug. 2000.
[29] C.-S. Park, "Guaranteed-stable sliding DFT algorithm with minimal computational requirements," *IEEE Trans. Signal Process.*, vol. 65, no. 20, pp. 5281–5288, Oct. 2017.
[30] A. van der Byl and M. R. Inggs, "Constraining error—A sliding discrete Fourier transform investigation," *Digit. Signal Process.*, vol. 51, pp. 54–61, Apr. 2016.
[31] F. Xiong, *Digital Modulation Techniques* (Artech House Telecommunications Library). Norwood, MA, USA: Artech House, 2000.
[32] J. Proakis and D. Manolakis, *Digital Signal Processing*. London, U.K.: Pearson Prentice-Hall, 2007.

SIAVASH SAFAPOURHAJARI (SAFAPOUR) (Member, IEEE) was born in Bandaranzali, Iran, in 1987. He received the B.Sc. and M.Sc. degrees in electrical engineering from the University of Tehran, Tehran, Iran, and the Ph.D. degree from the Faculty of Electrical Engineering, Computer Science and Mathematics (EEMCS), University of Twente, The Netherlands, in 2015, where he focused on ultra-narrowband communications for low power Internet of Things (IoT).

His research interests include signal processing for communications, efficient implementation of signal processing algorithms, the low power IoT, and antenna arrays. He was a recipient of the German Academic Exchange Service (DAAD) Fellowship for research project in the University of Rostock during his M.Sc. study.

ANDRÉ B. J. KOKKELER (Member, IEEE) received the M.Sc. and the Ph.D. degrees in computer science from the University of Twente, The Netherlands, in 1988 and 2005, respectively.

He has worked more than six years at Ericsson as a System Engineer and eight years at The Netherlands Institute for Radio Astronomy (ASTRON) as a Scientific Project Manager. In 2003, he joined the University of Twente, where he is currently the Chair of the Radio Systems (RS) Research Group. He has a background in telecommunications, mixed signal design, and signal processing. His main research interests include the area of the design of low-power architectures for telecommunications and computationally intensive applications. He is involved in research projects, sponsored by the Dutch and European governments and industry.