Low Complexity Synchronization for Offset Tolerant DFT-Based BFSK Demodulator

Siavash Safapourhajari, André B. J. Kokkeler
University of Twente
Enschede, Netherlands
s.safapourhajari@utwente.nl, a.b.j.kokkeler@utwente.nl

***Abstract*—A DFT-based demodulator for BFSK is used for applications where the received signal experiences a carrier frequency offset (CFO) much larger than the data rate. It is particularly interesting for emerging ultra-narrowband communications for wireless sensor networks and IoT. The drawback is that the synchronization algorithm proposed for such a demodulator involves calculating the DFT for a window sliding over the whole preamble. This imposes a large computational load which is not desired in low power applications. To overcome CFO the sampling frequency should be larger than the signal bandwidth. Thus, the spectral range of the DFT is larger than the signal bandwidth. This means that only a subset of DFT bins have information about the signal; however, due to unknown CFO, the conventional synchronization algorithm needs all bins of the DFT. In this work, a novel synchronization algorithm is proposed which only needs a subset of DFT bins. Such algorithm can simplify implementation because efficient Single Bin Sliding DFT (SB-SDFT) algorithms can be used. Moreover, to be able to use the SB-SDFT algorithm, it is modified to incorporate zero-padding. The proposed algorithm and its implementation using the modified SB-SDFT reduce the number of complex multiplications, complex additions and memory usage by 28%, 64% and 81%, respectively, while achieving the same BER performance for the demodulator.**

***Index Terms*—BFSK, Frequency Offset, Sliding DFT, Ultra-narrowband, Synchronization, Offset Tolerant Demodulator**

I. Introduction

A DFT-based demodulator for BFSK has been proposed as a solution to large carrier frequency offset (CFO) in low data rate applications such as satellite communications [1]. CFO can be a consequence of either the mismatch between oscillators in the communication nodes or the Doppler shift resulting from their relative movement. In the emerging area of ultra-narrowband communications for wireless sensor nodes and IoT, the CFO might be several times the signal bandwidth [2]. A modified version of the demodulator in [1] was proposed in [3] for narrowband wireless sensor nodes. In these designs, the signal passes through a low pass filter before the demodulator. The filter bandwidth and the sampling frequency can be much larger than the signal bandwidth so that the signal is captured even in presence of large CFO. The samples belonging to each symbol are selected by a rectangular window, padded with zeros and their DFT is calculated. The detection is done based on the magnitudes of the DFT bins for each symbol.

The main drawback of the demodulators in [1] and [3] is the complexity of the window synchronization algorithm. The most computationally intensive part of this algorithm (which is elaborated later) includes calculating the DFT for a window sliding over the whole preamble with one sample hop. In case of a sliding window only one sample is different between the updated set of DFT inputs and the previous set. Exploiting this property, various methods have been proposed for efficient implementation of the DFT with a sliding window [4]–[8]. In [4], the Sliding FFT was introduced based on the Radix-2 FFT structure. It stores and reuses all intermediate values in the FFT algorithm. For an N-point FFT, only $N-1$ butterflies are calculated to update outputs. Significant memory requirement is the cost of this method. Another class of methods, that can save a lot of complexity, focuses on calculating a Single Bin Sliding DFT (SB-SDFT) using an IIR filter structure [5]–[7]. However, there are two problems with using an SB-SDFT method in a conventional demodulator. First, these methods are more efficient than an SFFT only when a subset of DFT bins is required [8] while conventional synchronization uses all DFT bins. Second, the proposed structures for SB-SDFT can not be utilized in presence of zero-padding, which is necessary in a DFT-Based demodulator [1], [3].

Taking an integrated approach, this work contributes to the implementation of an efficient DFT-based demodulator through solving the two mentioned problems. First, a novel synchronization algorithm which only requires a subset of the DFT bins is proposed. It makes it possible to leverage low complexity SB-SDFT algorithms. Second, a modified SB-SDFT algorithm is derived to enable calculating DFTs in presence of zero-padding (which is required by the demodulator).

In the next section, the conventional synchronization algorithm is briefly explained. The proposed algorithm is presented in Section III while Section IV elaborates on its implementation by explicating the modified SB-SDFT and analyzing complexity. Section V includes simulation results and a complexity comparison. Finally, conclusions are drawn in Section VI.

II. Conventional Synchronization

A. The algorithm

The baseband equivalent of a DFT-based demodulator is shown in Fig. 1 [1], [3]. To tolerate large frequency offset, the lowpass filter might be much wider than the signal bandwidth. Moreover, complying with the Nyquist criterion necessitates

978-1-5386-6528-2/19/$31.00 ©2019 IEEE

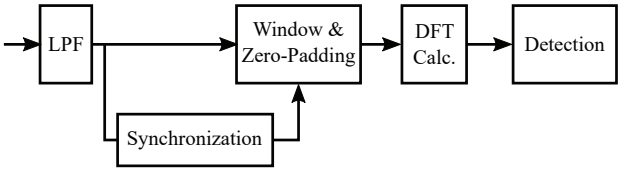

Fig. 1. Baseband equivalent block diagram of DFT-based demodulator

a higher sampling frequency. After the filter, a proper set of samples (all belonging to the same symbol) is selected by a rectangular window, padded with zeros and sent to the DFT calculation block. Zero-padding is necessary to increase the DFT bin resolution and ensure the best performance of the detection block [1]. To make sure that all samples in a set belong to one symbol, the window needs to be aligned to symbols. This is done by a fine synchronization algorithm which is similar in both [1] and [3]. It uses a preamble of alternating ones and zeros. Windows with different delay values (in term of samples) are considered for each symbol. Fig. 2 illustrates the first three symbols of the preamble and the windows in case of four samples per symbol. It also shows how the spectrum varies for different delays. The DFT magnitudes for each symbol and all delay values are calculated. Then, for each delay value, DFT magnitudes corresponding to odd (1) and even (0) symbols are added together. In fact, along each row of windows (corresponding to a certain "Delay") shown in Fig. 2, the sum of the DFT magnitudes for solid windows and the sum of those for dotted windows are calculated. To synchronize the window the algorithm finds the delay value for which R_m in the following equation is maximized [1], [3].

$$R_m = [E_e^m(f_e^m) - E_e^m(f_o^m)] + [E_o^m(f_o^m) - E_o^m(f_e^m)] \quad (1)$$

where E_e^m and E_o^m are the accumulated DFT magnitudes corresponding to delay m for even and odd symbols, respectively. f_e^m and f_o^m are the bins with maximum magnitude in E_e^m and E_o^m, respectively. Finding the maximum of (1) is simply finding the delay value which maximizes the difference shown by D in Fig. 2.

B. Complexity analysis

Considering the DFT calculation for all symbols of the preamble and different delay values, it is, indeed, calculating the DFT for a window sliding over a sequence of samples. As mentioned before, when all bins of a sliding DFT are required, an efficient implementation is achieved using the Sliding FFT (SFFT) [4]. For complexity analysis, it is assumed that the preamble length and the number of samples per symbol are L and N, respectively. The zero-padding factor is denoted by I which means after zero-padding the sequence of samples has a length of NI for each symbol ($N(I-1)$ zeros are added to the samples). Both N and I are powers of two as required by the FFT. In the conventional synchronization algorithm NI-point DFTs are used; therefore, for each update of the SFFT output $NI-1$ butterflies are calculated [4]. Each butterfly is composed of one complex multiplication and two complex additions. Using the SFFT, first, an NI-point FFT is calculated for the first set of N samples which involves

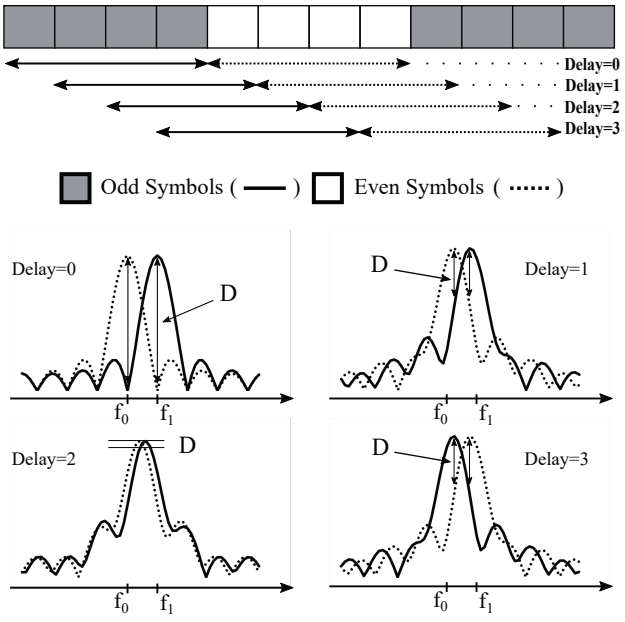

Fig. 2. The first three symbols of a preamble, the windows of different delay values for the first two symbols and an example of spectrum variation when there are 4 samples per symbol. Solid and dotted lines show windows and spectra of odd and even symbols, respectively.

$(NI/2)\log_2(NI)$ butterflies. Afterwards, $L \times N - 1$ updates are required for sliding over the whole preamble. So the total numbers of complex additions (CA) and multiplications (CM) for the conventional synchronization algorithm implemented by the SFFT are as follows.

$$CA_{SFFT} = NI\log_2(NI) + 2(LN-1)(NI-1) \quad (2)$$

$$CM_{SFFT} = (NI/2)\log_2(NI) + (LN-1)(NI-1) \quad (3)$$

To calculate the memory usage, two prominent storage requirements are considered. First, the memory needed to calculate the DFT for each window (M_{Calc}) and, second, the memory needed to store E_e^m and E_o^m (M_{Acc}). The memory for storing twiddle factors is ignored as they are common in the conventional and the proposed methods. In case of the SFFT, for the outputs of each butterfly (two outputs) in the FFT which is not calculated during an SFFT update, two memories are needed to store their value from the previous window [4]. Besides, as a consequence of zero-padding, at the s^{th} stage ($s = 1, ..., \log_2 I$) of the Radix-2 FFT structure (which is the basis for SFFT) there are only $N2^{(s-1)}$ non-zero butterfly outputs. For the rest of the stages ($s = \log_2 I+1, ..., \log_2(NI)$) all $NI/2$ butterflies have non-zero outputs. These values are complex and need two memories each. Hence, M_{Calc} for the SFFT is:

$$M_{Calc}^{SFFT} = \sum_{s=1}^{\log_2 I} 4(N2^{s-1}-1) + \sum_{s=\log_2 I+1}^{\log_2(NI)} 4(NI/2-1) \quad (4)$$

To compute M_{Acc}^{SFFT}, it should be noticed that for each delay value two sets of memories are used to store the accumulated DFT bins magnitudes for odd and even symbols. For the conventional algorithm, each DFT has NI bins. So, considering N delay values, $M_{Acc}^{SFFT} = 2IN^2$.

III. Proposed Synchronization Algorithm

As said previously, only a limited number of bins which are in the vicinity of the signal center frequency, $f_c = (f_0 + f_1)/2 + CFO$ ($f_c = CFO$ in the baseband signal), are *interesting* for synchronization and detection. This set of bins which the proposed method tries to find is called the Bins of Interest (BoI) hereafter. Notice that the BoI is not known due to frequency offset. Fig. 3 depicts the block diagram of the proposed synchronization. Signal samples $x[n]$ (top left) pass through a sliding rectangular window which selects sets of N samples (where N is the number of samples per symbol). These sample sets are used in the synchronization procedure. As can be seen, this procedure is split into two stages; zoom and window alignment. First, the BoI of an NI-point DFT is detected through step-by-step zooming. In the second stage, a proper window delay is obtained based on the magnitude of the DFT for bins in the BoI. Before elaborating on each stage in Fig. 3, let us mention that T is symbol period and $Ts = T/N$ is the sampling time; $I = 2^{\Gamma}$ denotes the zero-padding factor and N is considered to be a power of two (the same as in [1], [3]). Moreover, the frequency deviation of the BFSK modulation is assumed to be equal to the data rate ($f_1 = f_c + 1/2T$ and $f_0 = f_c - 1/2T$). This is the value that ensures the best BER performance while keeping the minimum bandwidth [1], [3]. To find the BoI of an NI-point DFT which covers the frequencies of the BFSK modulation ($f_1$and f_0), the zoom stage aims at searching for the center bin, k_c, in an NI-point DFT which is closest to the center frequency (f_c) of the signal. This stage includes $\Gamma + 1$ steps while the zero-padding factor at step γ is equal to 2^{γ} ($\gamma = 0, ..., \Gamma$). At step γ, the bin k_c^{γ} of an $N2^{\gamma}$-point DFT which is closest to f_c is detected. Using this bin, the BoI calculator block estimates the next step center bin ($\hat{k}_c^{\gamma+1}$) for an $N2^{\gamma+1}$-point DFT and finds a subset of the bins, $BoI_{\gamma+1}$, around $\hat{k}_c^{\gamma+1}$. In step $\gamma + 1$, the actual center bin $k_c^{\gamma+1}$ is detected by calculating DFT only for bins in the $BoI_{\gamma+1}$. The zoom stage starts with the first step, $\gamma = 0$, which uses an N-point DFT i.e. with zero-padding factor of one (or no zeros). Assume that the BoI at step γ (BoI_{γ}) is obtained from the previous step ($\gamma - 1$). At step γ, the window slides over the received sequence of samples for two symbols (an example of the windows for $N = 4$ and how the spectrum changes can be seen in Fig. 2). For each hop 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 (see Fig. 2). If the magnitudes of all these DFTs for each bin are added, the bin for which the sum is maximum is the closest to the center frequency k_c^{γ} (this is shown later).

$$\mathcal{X}_k^{\gamma} = \sum_{m=0}^{2N-1} |X_{k,m}^{\gamma}|^2, \;\; and \;\; k_c^{\gamma} = \max_{k \in BOI_{\gamma}} \mathcal{X}_k^{\gamma}, \qquad (5)$$

where $X_{k,m}^{\gamma}$ is the DFT for k^{th} bin and delay $m = 0, ..., 2N - 1$ in step γ. Since the zero-padding factor is doubled between two consecutive steps, the center bin for step $\gamma + 1$ can be estimated as $\hat{k}_c^{\gamma+1} = 2k_c^{\gamma}$. Then, $\hat{k}_c^{\gamma+1}$ is used to determine

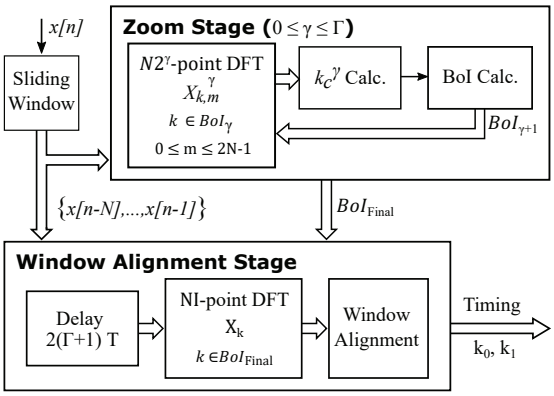

Fig. 3. The block diagram of the proposed synchronization algorithm; BoI stands for Bins of Interest

$BoI_{\gamma+1}$ for searching the center bin during next step, $\gamma + 1$. 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 two bins). In such cases a noisy received signal and leakage can cause wrong estimation of $k_c^{\gamma+1}$. 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. To account for any erroneous detection due to leakage and noise, $BoI_{\gamma+1}$ includes N bins around $\hat{k}_c^{\gamma+1}$. Assume $a = (\hat{k}_c^{\gamma+1} - N/2) mod (N2^{\gamma+1})$ and $b = (\hat{k}_c^{\gamma+1} + N/2 - 1) mod (N2^{\gamma+1})$ where mod is modulo operator. The modulo operator 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. $k \in \{0, ..., N2^{\gamma+1} - 1\}$. The $BoI_{\gamma+1}$ is:

$$BoI_{\gamma+1} = \{k | k \in [min(a, b), max(a, b)]\} \qquad (6)$$

The same step is repeated until the center bin of the NI-point DFT ($k_{c,Final}$) is detected. In this step, the final BoI used for the window alignment stage (BoI_{Final}) is selected. BoI_{Final} should include the frequency bins corresponding to symbol 1 and 0 of BFSK (k_1 and k_0, respectively). Since the frequency deviation of BFSK is equal to the data rate, when the zero-padding factor is I and the sampling frequency is N/T, there are $I - 1$ bins between k_1 and k_0. To reduce the effect of noise, BoI_{Final} is determined as follows.

$$BoI_{Final} = \{k | k \in [min(d, e), max(d, e)]\} \qquad (7)$$

where $d = (k_{c,Final} - I) mod (NI)$ and $e = (k_{c,Final} + I - 1) mod (NI)$. The size of BoI_{Final} is optimized to achieve the best BER performance using simulations; however, the results of these simulations are not included for sake of brevity. The BoI_{Final} is, then, sent to the next stage of synchronization which aligns the window and determines k_0 and k_1 [3]. This stage is exactly the same as the conventional algorithm, with the only difference that the DFT is only calculated for the bins in BoI_{Final}. There is a delay of $2(\Gamma + 1)T$ at the beginning of the alignment stage. This is the time needed to receive all symbols required by zoom stage. By considering this delay the samples can be reused for window alignment when the BoI is determined. This eliminates the need for a longer preamble.

As mentioned earlier, the algorithm is based on the fact that the bin which maximizes the sum in (5) is the closest one to the center frequency. This is shown in the following. The DFT calculated over N samples with zero-padding factor I, actually consists of the values $X(\Omega)$ of Discrete Time Fourier Transform (DTFT) for N samples, sampled at $\Omega = 2\pi F_s k/NI$ where F_s is the sampling frequency [9]. So to show (5) for the general case of all zero-padding factors, the DTFT is considered. The input signal is $x[n] = e^{j\omega_i n T_s}$ where $\omega_i = 2\pi f_i$ and f_i is either f_0 or f_1 of the BFSK. The DTFT of $x[n]$ over N samples is as follows.

$$X(\Omega) = \sum_{n=0}^{N-1} x[n] e^{-j\Omega n} \tag{8}$$

Let us consider a pair of windows in the summation of (5) with delay d of the odd symbol (with f_1) and the same delay for the next even symbol (f_0) (i.e. $m = d$ and $m = d + N$ in (5)). In Fig. 2 these are two consecutive windows of the same delay value. The DTFT for each window is calculated as follows.

$$X_d(\Omega) = \sum_{n=0}^{N-d-1} e^{j(\omega_1 T_s(n+d)-\Omega n)} + \sum_{n=N-d}^{N-1} e^{j(\omega_0 T_s(n+d)-\Omega n)} \tag{9}$$

$$X_{d+N}(\Omega) = \sum_{n=0}^{N-d-1} e^{j(\omega_0 T_s(n+d+N)-\Omega n)} + \sum_{n=N-d}^{N-1} e^{j(\omega_1 T_s(n+d+N)-\Omega n)} \tag{10}$$

where ω_1 and ω_0 are $2\pi f_1$ and $2\pi f_0$, respectively. Now, let us assume that $\Omega = \omega_c T_s + \alpha$ and calculate $|X_d(\alpha)|^2$ and $|X_{d+N}(\alpha)|^2$ based on (9) and (10) as follows.

$$|X_d(\alpha)|^2 = \sum_{n,p=0}^{N-d-1} e^{j(\frac{\pi}{N}-\alpha)(n-p)} + \sum_{n,p=0}^{d-1} e^{-j(\frac{\pi}{N}+\alpha)(n-p)} + 2\sum_{n=0}^{N-d-1}\sum_{p=N-d}^{N-1} \cos(\frac{\pi(2d+n+p)}{N} - \alpha(n-p)) \tag{11}$$

$$|X_{d+N}(\alpha)|^2 = \sum_{n,p=0}^{N-d-1} e^{-j(\frac{\pi}{N}+\alpha)(n-p)} + \sum_{n,p=0}^{d-1} e^{j(\frac{\pi}{N}-\alpha)(n-p)} + 2\sum_{n=0}^{N-d-1}\sum_{p=N-d}^{N-1} \cos(-\frac{\pi}{N}(2d+n+p) - \alpha(n-p)) \tag{12}$$

To derive (11) and (12), $\omega_1 T_s$ and $\omega_0 T_s$ are replaced with $\omega_c T_s \pm \frac{\pi}{N}$ while $Ts/T = 1/N$. In the second term of (11) and the first term of (12) the summation indexes are interchanged so that the negative sign in power can be removed. Considering this and using $\cos(x) = \cos(-x)$ for the third term of (12), it is easy to show that (11) and (12) are in the form of $F_d(\alpha)$ and $F_d(-\alpha)$. Thus, for $G_d(\alpha) = |X_d(\alpha)|^2 + |X_{N+d}(\alpha)|^2$ the first and the second derivatives are as follows.

$$\frac{d}{d\alpha}G(\alpha) = \frac{d}{d\alpha}F(\alpha) - \frac{d}{d\alpha}F(-\alpha) \tag{13}$$

$$\frac{d^2}{d\alpha^2}G(\alpha) = \frac{d^2}{d\alpha^2}F(\alpha) + \frac{d^2}{d\alpha^2}F(-\alpha) \tag{14}$$

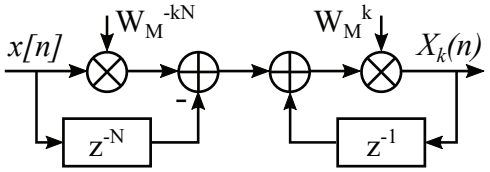

Fig. 4. The modified SB-SDFT filter

For $\alpha = 0$ the right side of (13) is zero for any d (statement I). Moreover, both $F_d(\alpha)$ and $F_d(-\alpha)$ have a maximum in $-\pi/N \leq \alpha \leq \pi/N$ (see spectra in Fig. 2) which means both have a negative second derivative at $\alpha = 0$. As a result, $\frac{d^2}{d\alpha^2}G(\alpha)$ is negative at $\alpha = 0$ for any d (statement II). From (I) and (II) it is concluded that the $G_d(\alpha)$ has a maximum at $\alpha = 0$. Considering the DTFT instead of DFT, the sum in (5) is the sum of $G_d(\alpha)$ for $d = 0, ..., N-1$. So, this sum has a maximum at $\alpha = 0$ which corresponds to $\Omega = \omega_c Ts$ or center frequency.

IV. Implementation

A. Modified SB-SDFT

Aforementioned implementations for SB-SDFT do not take the zero-padding into account. For the proposed synchronization algorithm, a new SB-SDFT algorithm is needed to include the zero-padding effect. A procedure similar to the conventional SB-SDFT derivation [5]–[7] is followed. The k^{th} bin of an M-point DFT over a set of N samples, $\mathbb{X}_n = \{x(n-N+1), ..., x(n)\}$, 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 are ignored as they are equal to zero. The k^{th} DFT bin in (15) can be obtained by using the k^{th} DFT bin for sample set $\mathbb{X}_{n-1} = \{x(n-N), ..., x(n-1)\}$ 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 (15) can be simplified to W_M^k leading to the known SB-SDFT equation [5]; however, 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 block diagram of such a filter is depicted in Fig. 4. The SB-SDFT with zero-padding has an extra multiplication in the samples path compared to SB-SDFT in [5].

B. Complexity analysis

The zoom stage has $\Gamma + 1$ steps ($\Gamma = \log_2 I$). For each step, $3N$ iterations of the filter shown in Fig. 4 are required to generate $2N$ DFT values of each bin. Notice that N iterations are required from the initial state of zero to generate the DFT

value for the first window. Each iteration needs two complex additions and two complex multiplications. For each step, N bins are required. Therefore, the number of complex additions and multiplications for the zoom stage are as follows.

$$CA_{Zoom} = CM_{Zoom} = 6(\log_2 I + 1)N^2 \tag{18}$$

In the window alignment stage, a window of length N slides over a preamble of length L and DFT values are calculated for $2I$ bins. So, for each bin $(L+1)N$ iterations of the SB-SDFT filter are required. Thus, the number of complex additions and multiplications for the window alignment stage are as follows.

$$CA_{Align} = CM_{Align} = 4IN(L+1) \tag{19}$$

Similar to Section II, M_{Calc}^{SDFT} and M_{Acc}^{SDFT} are derived for the proposed algorithm to evaluate memory usage. For the SB-SDFT each filter needs a memory of N for samples and a delay of one for the previous DFT value (See Fig. 4). The first stage of the memory can be shared between all $2I$ bins and the second is unique for each bin. Moreover, the delay block in Fig. 3 requires storing $2N(\Gamma+1)$ samples.

$$M_{Calc}^{SDFT} = 2(N + 2I + 2N(\Gamma+1)) \tag{20}$$

The proposed method only needs to store accumulation of the DFT magnitudes for $2I$ bins which leads to $M_{Acc}^{SDFT} = 2IN$. This considerable decrease in M_{Acc} is another advantage of using a subset of DFT bins (BoI).

V. Results and Discussion

The synchronization algorithm is applied to a DFT-based demodulator with parameters similar to [3]. The sampling frequency is $8R_B$ where R_B is data rate while $N = I = 8$. The total system is simulated in an AWGN channel. It is assumed that the samples are the output of an ideal lowpass filter so the noise samples are uncorrelated and independent. The BER performance of a DFT based demodulator using the proposed and conventional synchronization [3] algorithms are depicted in Fig. 5. As can be seen, the performance of the proposed method is only slightly worse at high BER values; however, for practical BER values in order of 10^{-3} or smaller it performs the same as the conventional method. The small increase in BER at low SNR is due to a slight increase of error caused by using a small set of bins.

Table I lists the required complex additions, complex multiplications and memory when $L = 16$ and $N = I = 8$ (parameters in [3]) for the proposed and conventional algorithms. Compared to the efficient implementation of the conventional method using the SFFT, the proposed method achieves 64%, 28% and 81% saving in the number of complex additions, complex multiplications and memory, respectively.

VI. Conclusion

The offset tolerant DFT-based demodulator is an interesting solution for low data rate applications including emerging ultra-narrowband communications for wireless sensor nodes. 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 modified version of SB-SDFT was introduced to incorporate zero padding which is part of the demodulator. The proposed algorithm and its implementation using the modified Sliding DFT, obtains 64%, 28%, 81% saving in the number of complex additions, complex multiplications and memory usage while achieving the same BER performance.

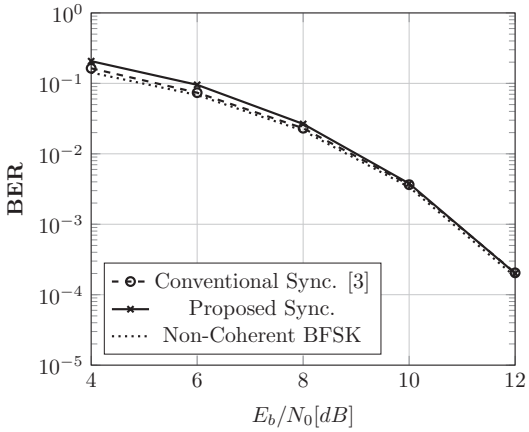

Fig. 5. The BER curves for the demodulator in [3] with the proposed and the conventional synchronization

TABLE I
Complexity comparison

Synchronization	CA ($\times 10^3$)	CM ($\times 10^3$)	Memory ($\times 10^3$)
Conventional [3]	**16.386**	**8.193**	**1.608**
Proposed	**5.888**	**5.888**	**0.304**

References

[1] 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 Transactions on Vehicular Technology*, vol. 46, no. 3, pp. 625–633, Aug 1997.

[2] D. Lachartre, F. Dehmas, C. Bernier, C. Fourtet, L. Ouvry, F. Lepin, E. Mercier, S. Hamard, L. Zirphile, S. Thuries, and F. Chaix, "7.5 a TCXO-less 100Hz-minimum-bandwidth transceiver for ultra-narrow-band sub-GHz IoT cellular networks," in *2017 IEEE International Solid-State Circuits Conference (ISSCC)*, Feb 2017, pp. 134–135.

[3] S. Safapourhajari and A. B. J. Kokkeler, "Frequency offset tolerant demodulation for low data rate and narrowband wireless sensor node," in *2017 11th International Conference on Signal Processing and Communication Systems (ICSPCS)*, Dec 2017, pp. 1–8.

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

[5] E. Jacobsen and R. Lyons, "The sliding DFT," *IEEE Signal Processing Magazine*, vol. 20, no. 2, pp. 74–80, March 2003.

[6] K. Duda, "Accurate, guaranteed stable, sliding discrete Fourier transform," *IEEE Signal Processing Magazine*, vol. 27, no. 6, pp. 124–127, Nov 2010.

[7] C. Park, "Fast, accurate, and guaranteed stable sliding discrete Fourier transform," *IEEE Signal Processing Magazine*, vol. 32, no. 4, pp. 145–156, July 2015.

[8] ——, "Guaranteed-stable sliding DFT algorithm with minimal computational requirements," *IEEE Transactions on Signal Processing*, vol. 65, no. 20, pp. 5281–5288, Oct 2017.

[9] J. Proakis and D. Manolakis, *Digital Signal Processing*. Pearson Prentice Hall, 2007.