An Efficient Implementation of NC-OFDM Transceivers for Cognitive Radios

Rakesh Rajbanshi Alexander M. Wyglinski Gary J. Minden

Information and Telecommunication Technology Center
The University of Kansas, Lawrence, KS 66045
Email: {rajbansh, alexw, gminden}@ittc.ku.edu

Abstract

In this paper, we present an efficient implementation of a non-contiguous orthogonal frequency division multiplexing (NC-OFDM) transceiver for cognitive radio systems. NC-OFDM is designed to transmit information in the presence of incumbent users, deactivating subcarriers located in the vicinity of these users to avoid interference. Given that the core component of an NC-OFDM transceiver is the fast Fourier transform (FFT), and that several of the subcarriers are deactivated, it is possible to reduce the execution time by "pruning" the FFT. We propose an algorithm that efficiently and quickly prunes the FFT for NC-OFDM transceivers. Results show that the proposed algorithm substantially outperforms other FFT pruning algorithms when a medium to large number of subcarriers have been deactivated.

1 Introduction

As access to available spectrum is becoming increasingly difficult, several researchers have proposed the concept of spectrum pooling[1], as well as several transceiver designs for transmission across non-contiguous portions of spectrum, to alleviate this problem. For instance, orthogonal frequency division multiplexing (OFDM) is a promising candidate for a flexible spectrum pooling system [1]. In order to support high data rates, the transmission bandwidth of the OFDM transceiver must be large. However, a large contiguous bandwidth may not be available for transmission. To provide high data rates while avoiding interference with incumbent user transmission, a variant of OFDM called non-contiguous OFDM (NC-OFDM) was proposed [2–4], where the implementation achieves high data rates via collective usage of non-contiguous blocks of subcarriers. Simultaneously, NC-OFDM avoids interference with incumbent users by deactivating subcarriers within their vicinity. Thus, NC-OFDM is a viable transmission technology for cognitive radio transceivers [5] operating in dynamic spectrum access (DSA) networks.

In the implementation of an OFDM transceiver, the fast Fourier transform (FFT) algorithm is employed to make modulation and demodulation highly efficient in terms of hardware and computational complexity [6]. However, an NC-OFDM may have several subcarriers that are deactivated, i.e., zero-valued inputs. Thus, the hardware resources of the FFT are not fully being exploited. Therefore, a new approach is needed to efficiently implement the FFT when several subcarriers are deactivated.

It has been shown that for situations in which the relative number of zero-valued inputs is quite large, significant time savings can be obtained by "pruning" the FFT algorithm[2] [7]. Several algorithms have been proposed in literature for enhancing the efficiency of the FFT algorithm based on decimation-in-time (DIT) and decimation-in-frequency (DIF) algorithms [8–15]. However, most of these algorithms are suitable only for systems with specific zero-input pattern distributions. Furthermore, algorithms that prune the FFT for any zero-input pattern do not yield an efficient implementation with respect to computational time [8].

In this paper, we present an FFT pruning algorithm designed for NC-OFDM transceivers. The proposed algorithm can quickly design an efficient FFT implementation for any zero-input pattern. The performance of the proposed algorithm is compared with several other algorithms proposed in the literature with respect to mean execution time. The rest of the paper is organized as follows: Section 2 introduces the NC-OFDM framework. Section 3 provides provides an overview of FFT pruning algorithms. Section 4 presents the proposed algorithm. Simulation results are presented in Section 5, while several concluding remarks are made in Section 6.

2 NC-OFDM Framework

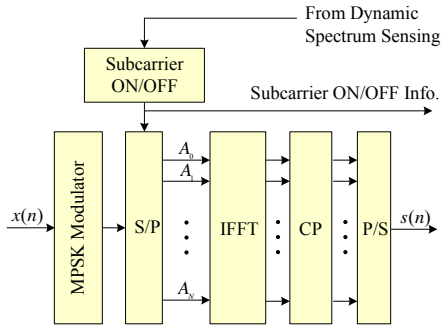
A general schematic of an NC-OFDM transceiver is shown in Fig. 1. Without loss of generality, a high speed data stream, $x(n)$, is modulated using M-ary phase shift keying (MPSK)[3]. Then, the modulated data stream is split into N slower data streams using a serial-to-parallel (S/P)

This work was supported by NSF grants ANI-0230786 and ANI-0335272.

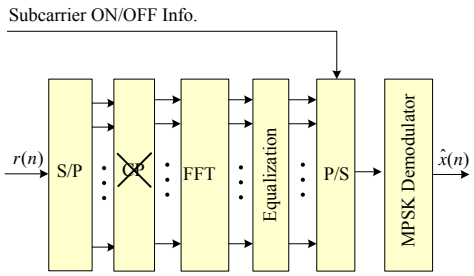
[1] *Spectrum pooling* is a resource sharing strategy with the highest priority for the owner of the license to enhance spectral efficiency. It enables the secondary utilization of already licensed frequency bands as aimed at by several regulatory authorities worldwide [1].

[2] *FFT pruning* refers to the procedure for improving the efficiency of the fast Fourier transform by removing operations on input values which are zeroes, and on output values which are not required [7].

[3] Other forms of digital modulation, including MQAM, can also be employed by the transceiver.



(a) NC-OFDM Transmitter.



(b) NC-OFDM Receiver.

Fig. 1 Schematic of an NC-OFDM transceiver.

converter. Note that the subcarriers in the NC-OFDM transceiver do not need to be all active as in conventional OFDM. Moreover, active subcarriers are located in the unoccupied spectrum bands, which are determined by dynamic spectrum sensing and channel estimation techniques [3, 16, 17]. The inverse fast Fourier transform (FFT) is then applied to these modulated subcarrier signals. Prior to transmission, a guard interval with a length greater than the channel delay spread is added to each NC-OFDM symbol using the cyclic prefix (CP) block in order to mitigate the effects of intersymbol interference (ISI). Following the parallel-to-serial (P/S) conversion, the baseband NC-OFDM signal, $s(n)$, is then passed through the transmitter radio frequency (RF) chain, which amplifies the signal and upconverts it to the desired center frequency.

The receiver performs the reverse operation of the transmitter, mixing the RF signal to baseband for processing, yielding the signal $r(n)$. Then, the signal is converted into parallel streams using S/P converter, the cyclic prefix (CP) is discarded, and the fast Fourier transform (FFT) is applied to transform the time domain data into the frequency domain. After compensating distortion introduced by the channel using per-tone equalization [18], the data in the active subcarriers is multiplexed using a P/S converter, and demodulated into a reconstructed version of the original high-speed input, $\hat{x}(n)$.

From this system overview, we observe that the IFFT and FFT blocks are critical components of the transceiver. In the next section, we will describe how it is possible to implement efficient versions of these blocks.

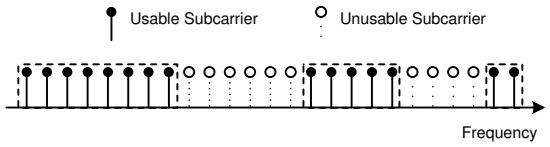


Fig. 2 Subcarrier distribution over wideband spectrum

3 FFT Pruning Technique

In a wide-band communication system, a large portion of frequency channels may be occupied by other transmissions, whether incumbent or other unlicensed users. As a result, these subcarriers are off-limits to our transceiver. Thus, to avoid interfering with these other transmissions, the subcarrier within their vicinity are turned off, or *nulled*, as shown in Fig. 2. For the FFT and IFFT blocks, these null subcarriers are represented as zero-valued inputs. For highly sparse available frequency spectrum, the number of zero-valued inputs in the FFT may be significant relative to the total number of the usable subcarriers. When the relative number of zero-valued inputs is quite large, significant time saving can be obtained by pruning the FFT algorithm.

For instance, an 8-point DIF FFT butterfly structure is shown in Fig. 3, where a_i represents the i^{th} input signal to the FFT block. Suppose the incumbent users are located at subcarriers a_1, a_5, a_7 and a_8. Therefore, input data over all these carrier must always be zero. For a conventional FFT algorithm, the total number of multiplications and additions would be $N \log_2 N$. However, with an FFT pruning algorithm, the unnecessary multiplications and addition operations at the stages b_1 and b_5 can be pruned as their values will always be zeroes. Moreover, multiplications and additions at nodes b_3, b_4, b_7, and b_8 can be replaced with simple 'copy' operation, whereas addition operations in nodes c_1, c_3, c_5, and c_7 can be pruned to save the FFT computation time. Therefore, the FFT computation time can be significantly improved with partial and complete pruning.

In wideband communication systems, the channel conditions and incumbent user occupancy[4] (ISO) varies over time. Thus, the FFT pruning algorithm should be able to design an efficient FFT implementation every time the channel condition and ISO changes.

3.1 General FFT Pruning Algorithm

Alves *et al.* proposed an FFT pruning algorithm that operates on any zero-valued input distribution [8]. Suppose we have a radix-2 FFT algorithm with N levels (2^N FFT points). A matrix M_i, with N columns and 2^N rows is generated using Algorithm 1. Each element of the matrix corresponds to a addition/multiplication node of the FFT flow graph. The node needs to be computed if the

[4] *Incumbent spectral occupancy* (ISO) is defined as the fraction of the intended transmission bandwidth occupied by incumbent user transmissions.

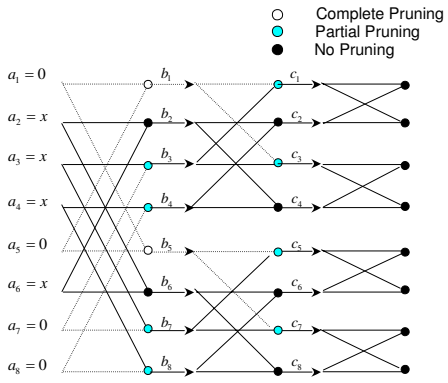


Fig. 3 FFT butterfly structure. A value of '0' denotes a zero-valued subcarrier and 'x' denotes a data bearing subcarrier. The dotted lines represent the computations that can be pruned.

corresponding element in the matrix M_i is non-zero. On the other hand, if the element of the matrix is zero, the corresponding node does not need to be computed. For instance, the matrix M_i for the FFT butterfly structure in Fig. 3 would be:

$$M_i = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}.$$

To obtain the matrix M_i, a subcarrier input vector with 2^N elements, where each element of this vector correspond to each input element. If the input element is nonzero, the corresponding vector element will be unity, and if the input element is zero, the corresponding vector element will be zero. By using this input vector, we can compute the first column of the matrix M_i. In turn, by using the first column of the matrix M_i, second column of matrix M_i can be obtained, and so forth. The algorithm to obtain the matrix M_i is presented in Algorithm 1.

Note that the FFT pruning algorithm proposed by Alves *et al.* uses conditional statements [8]. However, it is known that time to execute the conditional statement often exceeds the savings obtained by the fewer operations [19]. Therefore, we propose a re-indexing algorithm and modified FFT pruning algorithm to avoid using the conditional statements and reduce FFT computation times.

4 Proposed Algorithm

To save an execution time, the proposed algorithm builds upon the previous algorithm by avoiding the use of conditional statements. The proposed FFT pruning algorithm is shown in Algorithm 2. The algorithm is based on Cooley-Tukey divide-and-conquer algorithm that uses in-place computation [20]. For a radix-2 FFT, Cooley-Tukey

Algorithm 1 Generate M_i (proposed by Alves *et al.* [8])

function M_i=generateMi(m,ivector)

1: $n = 2^m$
2: $M_i = zeros(n, m)$
3: **for** $l = 1$ to m **do**
4: $shift1 = 2^{(m-l)}$
5: $shift2 = 2 \times shift1$
6: **for** $j = 1$ to $shift1$ **do**
7: **for** $k = shift2 : shift2 : n$ **do**
8: $j1 = k - shift1 - j + 1$
9: $j2 = j1 + shift1$
10: $j1 = k - shift1 - j + 1$
11: $j2 = j1 + shift1$
12: **if** $l == 1$ **then**
13: **if** $ivector(j1) == 1$ **then**
14: $M_i(j1, l) = 1$
15: $M_i(j2, l) = 1$
16: **end if**
17: **if** $ivector(j2) == 1$ **then**
18: $M_i(j2, l) = 1$
19: $M_i(j1, l) = 1$
20: **end if**
21: **else**
22: **if** $M_i(j1, l-1) == 1$ **then**
23: $M_i(j1, l) = 1$
24: $M_i(j2, l) = 1$
25: **end if**
26: **if** $M_i(j2, l-1) == 1$ **then**
27: $M_i(j2, l) = 1$
28: $M_i(j1, l) = 1$
29: **end if**
30: **end if**
31: **end for**
32: **end for**
33: **end for**
34: **return** M_i

algorithm divides the problem size into two interleaved halves with each recursive stage. This manner of computation requires the computations proportional to $N \log_2 N$, whereas the equivalent discrete Fourier transform (DFT) would require the computations proportional to N^2. In this work, the proposed algorithm operates in the similar manner. Additionally, the proposed algorithm prunes the unnecessary multiplication and addition operations at the nodes in the FFT flow graph, in order to reduce the execution time for the FFT computations.

First, the matrix M_i similar to the one in Section 3.1 is calculated, where each element of the matrix corresponds to a node of the FFT flow graph. Suppose we have a radix-2 algorithm with N levels (2^N FFT points). Then, the matrix M_i has N columns and 2^N rows. Second, information in the matrix M_i is processed to the matrix M_{index}, where indices and the total number of nonzero elements in each column of the matrix M are recorded. To obtain the matrix M_{index}, we employ Algorithm 3. The matrix M_{index} has N columns and $2^N + 1$ rows. For the example in Fig. 3,

Algorithm 2 Proposed FFT Pruning Algorithm

$m = 10$
$n = 2^m$
$i = \sqrt{-1}$
for $l = 1$ to m **do**
  $le = 2^{(m+1-l)}$
  $le_2 = le/2$
  $u = 1$
  $w = \cos(\pi/le_2) - \sin(\pi/le_2) \times i$
  for $j = 1$ to $le_2 - M_{\text{index}}(1, l)$ **do**
    for $ii = 1$ to $M_{\text{index}}(j+2, l) - M_{\text{index}}(j+1, l) - 1$ **do**
      $u = u \times w$
    end for
    for $k = M_{\text{index}}(j+2, l)$ to n **do**
      $ip = k + le_2$
      $t = x(k) + x(ip)$
      $x(ip) = (x(k) - x(ip)) \times u$
      $x(k) = t$
      $k = k + le$
    end for
    $u = u \times w$
  end for
end for

the matrix M_{index} would be:

$$M_{\text{index}} = \begin{pmatrix} 6 & 8 & 8 \\ 2 & 1 & 1 \\ 3 & 2 & 2 \\ 4 & 3 & 3 \\ 6 & 4 & 4 \\ 7 & 5 & 5 \\ 8 & 6 & 6 \\ 0 & 7 & 7 \\ 0 & 8 & 8 \end{pmatrix}.$$

The first row of the matrix M_{index} tells the number of nodes and the column of the matrix M_{index} provides the indices of the nodes that needs to be calculated in each FFT stage. Proposed FFT pruning algorithm in Algorithm 2 uses information provided by M_{index} to prune unnecessary computations at the corresponding nodes, hence reducing the execution time for the FFT computation.

5 Simulation Results

For the simulations, $N = 1024$ BPSK-modulated subcarriers were employed. Mean execution time for the FFT operations for the original Cooley-Tukey algorithm, the algorithm by Alves *et al.*, and the proposed algorithm were compared for 10,000 random data inputs, with the range of sparseness factor[5] from $0 - 99\%$.

In Fig. 4, the mean execution times for the three FFT algorithms are presented for the case of 1024-point FFT. We observe significant reduction in the mean execution time for calculating the FFT with the proposed algorithm as compared to the conventional Cooley-Tukey algorithm for the sparseness factor of 60% or higher. On the other hand, for a sparseness factor of less than 60%, the proposed algorithm performs slighly worse. Moreover, we find the time to execute the conditional statements exceeds the savings obtained by the FFT pruning. We observe reduction in the mean execution times of the proposed algorithm for calculating the 1024-point FFT with increase in the sparseness factor. However, the mean execution time of Cooley-Tukey algorithm and Alves *et al.* algorithm remain relatively constant all the time.

[5] *Sparseness factor* simply denotes the fraction of the zeroes in a given data set.

Algorithm 3 Proposed M_{index} Calculator

function M_{index}=generateindex(M_i)
$[n, m]$=size(M) % n rows, m columns
y=zeros(n, m)
for $j = 1$ to m **do**
  te=find$(M(:, j))$
  $y(1$:length$(te), j) = te$
end for
$y = [sum(M, 1); zeros(1, m); y]$
$j = [1 : m]$
$y(1, :) = (n - y(1, :))./2^j$
return $M_{\text{index}} = y$

In Fig. 5, the mean number of multiplications employed by 1024-point FFT are presented for three FFT algorithms. We observe that the reduction in the multiplication and addition operations due to FFT pruning. The reduction in addition and multiplication operations with the proposed algorithm is same as that achieved by Alves *et al.* algorithm. On the other hand, the proposed algorithm avoids using conditional statements. Therefore, the proposed algorithm achieves reduction in FFT computation time.

6 Conclusion

In this paper, we present an FFT pruning algorithm for use in NC-OFDM transceivers. The proposed algorithm can accept any zero-valued input distribution and prune the FFT to yield an implementation that results in a faster execution time. Given that the cognitive radio units employing NC-OFDM would need to quickly adapt to the changing operating environment, and that the hardware resources of small form factor cognitive radios are limited, such an algorithm would be very beneficial.

References

[1] T. Weiss and F. Jondral, "Spectrum pooling: an innovative strategy for the enhancement of spectrum efficiency," *IEEE Commun. Mag.*, vol. 42, pp. S8– 14, March 2004.

[2] J. D. Poston and W. D. Horne, "Discontiguous OFDM considerations for dynamic spectrum access in idle TV channels," in *Proc. IEEE Int. Symp. New Frontiers Dynamic Spectr. Access Networks*, vol. 1, (Baltimore, MD, USA), pp. 607–610, Nov. 2005.

[3] H. Tang, "Some physical layer issues of wide-band cognitive radio systems," in *Proc. IEEE Int. Symp. New Frontiers Dynamic Spectr. Access Networks*, vol. 1, (Baltimore, MD, USA), pp. 151–159, Nov. 2005.

[4] M. P. Wylie-Green, "Dynamic spectrum sensing by multiband OFDM radio for interference mitigation," in *Proc. IEEE Int.*

Symp. New Frontiers Dynamic Spectr. Access Networks, vol. 1, (Baltimore, MD, USA), pp. 619–625, Nov. 2005.

[5] J. Mitola, III, "Cognitive radio for flexible mobile multimedia communications," in *Proc. IEEE Int. Wksp. Mobile Multimedia Commun.*, vol. 1, (San Diego, CA, USA), pp. 3–10, Nov. 1999.

[6] S. B. Weinstein and P. M. Ebert, "Data transmission by frequency division multiplexing using the discretefourier transform," *IEEE Trans. Commun. Technol.*, vol. 19, pp. 628 – 634, Oct 1971.

[7] J. D. Markel, "FFT Pruning," *IEEE Trans. Audio Electroacoust.*, vol. 19, pp. 305 – 311, Dec. 1971.

[8] R. G. Alves, P. L. Osorio, and M. N. S. Swamy, "General FFT Pruning Algorithm," in *Proc. 43rd IEEE Midwest Symp. Circuits and Systems*, vol. 3, pp. 1192 – 1195, Aug. 2000.

[9] S. Holm, "FFT pruning applied to time domain interpolation and peak localization," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, vol. 35, pp. 1776 – 1778, Dec. 1987.

[10] Z. Hu and H. Wan, "A Novel Generic Fast Fourier Transform Pruning technique and Complexity Analysis," *IEEE Trans. Signal Processing*, vol. 53, pp. 274 – 282, Jan. 2005.

[11] L. P. Jaroslavski, "Comments on "FFT algorithm for both input and output pruning"," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, vol. 29, pp. 448 – 449, June 1981.

[12] J. Schoukens, R. Pintelon, and H. Van Hamme, "The interpolated fast Fourier transform: a comparative study," *IEEE Trans. Instrum. Meas.*, vol. 41, pp. 226 – 232, Apr. 1992.

[13] D. P. Skinner, "Pruning the Decimation in time FFT algorithm," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, vol. 24, pp. 193 – 194, Apr. 1976.

[14] T. V. Sreenivas and P. V. S. Rao, "High-resolution narrow band spectra by FFT pruning," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, vol. 28, pp. 254 – 257, Apr. 1980.

[15] T. V. Sreenivas and P. Rao, "FFT algorithm for both input and output Pruning," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, vol. 27, pp. 291 – 292, June 1979.

[16] R. Etkin, A. Parekh, and D. Tse, "Spectrum sharing for unlicensed bands," in *Proc. IEEE Int. Symp. New Frontiers Dynamic Spectr. Access Networks*, vol. 1, (Baltimore, MD, USA), pp. 251–258, Nov. 2005.

[17] F. Weidling, D. Datla, V. Petty, P. Krishnan, and G. J. Minden, "A framework for RF spectrum measurements and analysis," in *Proc. IEEE Int. Symp. New Frontiers Dynamic Spectr. Access Networks*, vol. 1, (Baltimore, MD, USA), pp. 573–576, Nov. 2005.

[18] H. Sari, G. Karam, and I. Jeanclaude, "Transmission techniques for digital terrestrial TV broadcasting," *IEEE Commun. Mag.*, vol. 33, pp. 100–109, Feb. 1995.

[19] H. V. Sorensen and C. S. Burrus, "Efficient computation of the DFT with only a subset of input or output points," *IEEE Trans. Signal Processing*, vol. 41, pp. 1184 – 1200, Mar. 1993.

[20] J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex Fourier series," *Math. Computation*, vol. 19, pp. 297–301, Apr. 1965.

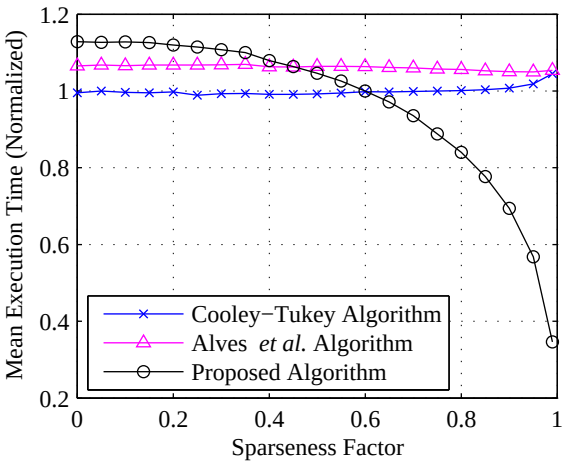


Fig. 4 Mean execution times for 1024-point FFT employing the three FFT algorithms

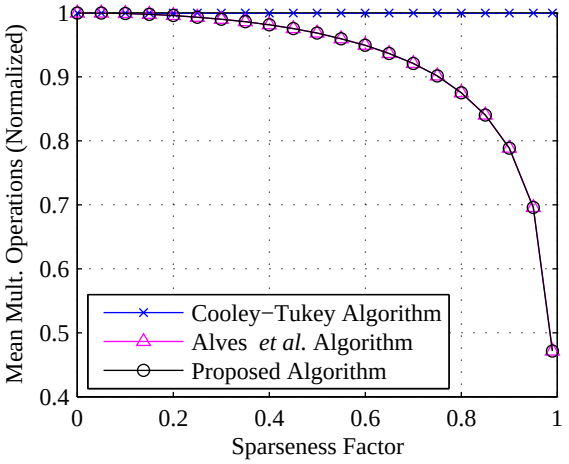


Fig. 5 Mean number of addition and multiplication operations for different 1024-point FFT employing the three FFT algorithms