Unified algorithms for generalized new Mersenne number transforms

Lujain S. Abdulla, Abdulmutalib A-Wahab Hussein, Mounir Taha Hamood
Department Electrical Engineering, College of Engineering, Tikrit University, Tikrit, Iraq

Article Info

Article history:

Received Apr 13, 2023
Revised Aug 5, 2023
Accepted Aug 15, 2023

Keywords:

Decimation-in-frequency
Fast transform algorithms
New Mersenne number transforms
Number theoretic transforms

ABSTRACT

The generalized new Mersenne number transforms (GNMNTs) have proved to be significant number theoretic transforms (NTTs) used to calculate convolutions and correlations accurately. In this paper, by applying the principles of the decimation-in-frequency (DIF) approach with appropriate relations in finite field modulo Mersenne primes, two new fast algorithms for computing odd NMNT (ONMNT) and odd-squared NMNT (O^{2}NMNT) are introduced. Moreover, by formulating a unified index mapping scheme for data sequence, a close relationship between the structures of the developed algorithms has been established. As a result, it has been shown that only a single universal butterfly structure is adequate to execute both algorithms. Consequently, a unified implementation platform can be used to compute the ONMNT as well as the O^{2}NMNT. The validity of the development has been checked via an example for fast calculations of different types of convolutions, using both the GNMNTs and the proposed algorithms.

Corresponding Author:

Mounir Taha Hamood
Department Electrical Engineering, College of Engineering, Tikrit University
P.O. BOX 42, Tikrit, Iraq
Email: m.t.hamood@tu.edu.iq

1. INTRODUCTION

The generalization of discrete transforms, such as generalized discrete Fourier transform (GDFT) [1] generalized discrete Wang or Hartley transforms (DWT/GDHT) [2]–[4], discrete sine transform (DST) [5] and discrete cosine transform (DCT) [6], have been proved to be an essential tool in digital signal processing and informatics fields, for instance, filter banks applications, signal representations [7] and fast computation of different kinds of convolutions [8]. While the new Mersenne number transform (NMNT) in its present form has wide applications in signal processing [9]–[11], image processing [12] and encryption [13]–[15], therefore, it is desirable to generalize the development of NMNT to other transforms such as the GNMNTs.

A complete set of the generalized NMNTs has been proposed by introducing two new transforms, which are the odd NMNT (ONMNT) and the odd-squared NMNT (O^{2}NMNT), for incorporation into a generalized type of the NMNT transforms, which are defined based on the modulo of the Mersenne primes within a finite field that can be utilized for efficient computations of exact convolution and correlation for signal processing and other applications [16], [17]. Using a transform length that is based on a power-of- two, they are inherently appropriate for implementation into efficient algorithms. Consequently, this allows for calculations using these transforms to be performed both efficiently and free from errors. Unlike their previous counterparts, such as Fermat number transform (FNT) [18], [19] and the Mersenne number transform (MNT) [20], these transforms are highly flexible as they are able to include many different transform lengths and make best use of the range defined by the modulus.

The name odd NMNT comes from the fact that if $X_o(k)$ is the N-point ONMNT for the sequence $x(n)$, then the output samples of $X_o(k)$ are the same as the odd-indexed samples of the length $2N$ NMNT of a sequence $\dot{x}(n)$, where $\dot{x}(n)$ is $x(n)$ with N zero padding. A similar case can be observed for the odd-squared NMNT; if $X_{oo}(k)$ is an N-point O^{2}NMNT for the sequence $x(n)$, then the output samples of $X_{oo}(k)$ are the same as the odd-indexed samples of the length $2N$ ONMNT of a sequence $\ddot{x}(n)$, where $\ddot{x}(n)$ is $x(n)$ with N zero padding.

The new transforms are therefore integer transforms defined by the Mersenne primes, with simple arithmetic operations equivalent to one's complement, and their calculations do not give rise to any rounding and/or truncation errors. These transforms have large transform length power-of-two, and hence their calculations can be developed using fast algorithms. They also possess the skew cyclic convolution property and therefore can be applicable for fast calculation of error free convolution and correlation functions.

The outline of this paper is as follows. Section 2 reviews the definition and properties of the ONMNT and O^{2}NMNT transforms and the calculation of their parameters. A complete derivation of proposed DIF ONMNT and O^{2}NMNT fast algorithms are developed in sections 3. Section 4 shows a method for efficient calculation of different types of convolutions based on GNMNTs.

2. THE GENERALIZED NMNT

Explaining In general, the transform kernel of the NMNT can be expanded to permit shifts in either time, frequency index or both indices [21], [22]. The developing invertible transforms referred to as generalized new Mersenne number transforms (GNMNTs) [16] defined for (n_o, k_o=1,0) as:

$$X(k) = \sum_{n=0}^{N-1} x(n)\beta\left(\frac{(2n+n_o)(2n+k_o)}{4}\right) mod\, M\, p\, k = 0,1,\dots,N-1 \quad (1)$$

The inverse GNMNTs are the scaled transpose of (1) defined as:

$$x(n) = \sum_{k=0}^{N-1} X(k)\beta\left(\frac{(2n+n_o)(2n+k_o)}{4}\right) mod\, M\, pn = 0,1,\dots,N-1 \quad (2)$$

where modMp represents modulo Mp and β is the transform kernel defined as:

$$\begin{aligned} &\beta(nk) = \beta_1(nk) + \beta_2(nk) \\ &\beta_1(nk) = Re((\alpha_1 + j\alpha_2)^d)^{nk}\, mod\, M\, p \\ &\beta_2(nk) = Im((\alpha_1 + j\alpha_2)^d)^{nk}\, mod\, M\, p \end{aligned} \quad (3)$$

Also α_1=±(2^q)modMp and α_2=±(-3^q)modMp for $q=2^{p-2}$, Im(.), Re(.) denote imaginary and real parts of the enclosed term respectively. The NMNT has the transform length for is $N=2^m$, $0 < m \le p$; $d=(2^{p+1}/N)$ is an integer power of two.

Three interesting special forms of the GNMNTs occur when n_o and k_o take the values 0 or 1. Firstly, when n_o=0 and k_o=0, the transform kernel will be $\beta(nk)$ and the GNMNT reduces to the NMNT. Secondly, when n_o=0 and k_o=1 it defines as an NMNT with a half sample delay in a time domain, giving a new transform called the odd NMNT (ONMNT). Its inverse can be obtained simply by introducing a half sample advance in the NMNT domain, i.e., n_o=1 and k_o=0, and it is called inverse ONMNT (IONMNT). Finally, setting both n_o=1 and k_o=1 produces another transform called the odd-squared ONMNT (O^{2}NMNT). Next section will review the definition and properties of the GNMNTs.

2.1. Definition and properties of GNMNTs

The ONMNT $X_o(k)$ of an integer sequence $x(n)$ for transform length ($N_o=2^m$, $0 < m \le p$-1) is defined by setting (n_o=0, k_o=1) in (1):

$$X_o(k) = \sum_{n=0}^{N-1} x(n)\beta\left(\frac{n(2k+1)}{2}\right) mod\, M\, p \qquad k = 0,1,\dots,N-1 \quad (4)$$

Similarly, the odd-squared O^{2}NMNT $X_{oo}(k)$ of an integer sequence $x(n)$ for transform length ($N_{oo}=2^m$, $0 < m \le p$-2) is defined by setting (n_o=1, k_o=1) in (1) as (5):

$$X_{oo}(k) = \sum_{n=0}^{N-1} x(n)\beta\left(\frac{(2n+1)(2k+1)}{4}\right) mod\, M\, p \qquad k = 0,1,\dots,N-1 \quad (5)$$

As shown from the definitions of the ONMNT given by (4) and the O^{2}NMNT given by (5), it is necessary to compute the half and quarter indices of the transform parameters (i.e., $\beta_1(n/2)$ and $\beta_2(n/2)$ for

ONMNT and $\beta_1(n/4)$ and $\beta_2(n/4)$ for O^{2}NMNT), that can be calculated from the definition of β_1 and β_2 given in (3) as:

a. The NMNT has a transform length N defined as (N=2^m, 0 < $m \leq p$), so the value of (d=2^{p+1}/N) in (3) for maximum transform length (N_{max}=2^p) is equal two.
b. Also, the length of transform for the ONMNT N_{o} defined as (N=2^m, 0 < $m \leq p$-1), thus the value of (d=2^{p+1}/N_o) to maximize transform length (N_{max}=2^{p-1}) is equal to four. Hence for a specified p and N, the values of the ONMNT parameters $\beta_1(n/2)$ and $\beta_2(n/2)$ have the identical values of the NMNT parameters $\beta_1(n)$ and $\beta_2(n)$ for length 2N respectively.
c. Similarly, the transform length of the O^{2}NMNT N_{oo} is defined as (N=2^m, 0 < $m \leq p$-2), thus the value of (d=2^{p+1}/N_{oo}) to maximize transform length (N_{max}=2^{p-2}) is equal to eight. Hence, for a specified p and N, the values of O^{2}NMNT parameters $\beta_1(n/4)$ and $\beta_2(n/4)$ have the same values as the NMNT parameters $\beta_1(n)$ and $\beta_2(n)$ respectively at length 4N.

As an example for the calculation of these parameters and without losing of generality, the Mersenne prime p is chosen to be 7 with modulus Mp=2^7-1=127, the values of α_1 and α_2 are α_1=(2^{32})mod 127=16 and α_2=(-3^{32})mod 127=88 respectively. For maximum transform length (d=2), the term (α_1+j α_2)^{2}modMp can be factorized to ((α^2_1- α^2_2)+2j$\alpha_1\alpha_2$)modMp. According to (3), β_1=(α^2_1- α^2_2) modMp and β_2=(2 $\alpha_1\alpha_2$)modMp, therefore, values of β_1 and β_2 are 5 and 22 respectively. For other transform lengths, β_1 and β_2 can be repeatedly calculated for various values of d as illustrated in Table 1. For sake of clarity and assuming all operations performed modulo Mersenne primes, the term modMp will be omitted for convenient in the algorithms development.

Table 1. NMNT parameters for modulus Mp=2^7-1=127

Transform length N	d	β_1	β_2
128	2	5	22
64	4	49	93
32	8	102	97
16	16	106	103
8	32	119	119
4	64	0	1
2	128	126	0

3. DEVELOPMENT OF FAST ALGORITHMS FOR GNMNTs

This section presents the development of ONMNT and O^{2}NMNT fast algorithms using the decimation in frequency approach. For ease derivation it starts with the following NMNT identity that has been proved in [10].

$$\beta(a+b) = \beta_1(a)\beta(b) + \beta_2(a)\beta(-b) \tag{6}$$

Also, some special values for the transform parameters β_1 and β_2 that have been proved in [11] are give in (7), because they are important in the algorithm's derivation.

$$\begin{aligned}\beta_1(a-b) &= [\beta_1(a)\beta_1(b) + \beta_2(a)\beta_2(b)]\\ \beta_2(a-b) &= [\beta_2(a)\beta_1(b) - \beta_1(a)\beta_2(b)]\end{aligned} \tag{7}$$

and for integer (v)

$$\begin{aligned}&\beta_1(v\frac{N}{4}) = (-1)^{v/2} and \beta_2(v\frac{N}{4}) = 0 evenv\\ &\beta_2(v\tfrac{N}{4}) = (-1)^{(v-1)/2} and \beta_1(v\tfrac{N}{4}) = 0 oddv\end{aligned} \tag{8}$$

3.1. Development of fast ONMNT algorithm

The derivation begins by applying (6) into the definition of ONMNT, therefore (4) can be decomposed to:

$$\begin{aligned}X_o(k) &= \textstyle\sum_{n=0}^{N-1} x(n)\beta\left(nk+\frac{n}{2}\right) = \sum_{n=0}^{N-1} x(n)\beta_1\left(\frac{n}{2}\right)\beta(nk) + \sum_{n=0}^{N-1} x(n)\beta_2\left(\frac{n}{2}\right)\beta(-nk) =\\ &\textstyle\sum_{n=0}^{N-1}\left[x(n)\beta_1\left(\frac{n}{2}\right) + x(N-n)\beta_2\left(\frac{n}{2}\right)\right]\beta(nk)\end{aligned} \tag{9}$$

In the first step, the input sequence *x(n)* is divided into two equal sequences, each having a length equal to *N*/2. Applying (6), we get:

$$X_o(k) = \sum_{n=0}^{N/2-1}\left[x(n)\beta_1\left(\frac{n}{2}\right) + x(N-n)\beta_2\left(\frac{n}{2}\right)\right]\beta(nk) \; - \sum_{n=0}^{N/2-1}\left[x\left(n+\frac{N}{2}\right)\beta_2\left(\frac{n+N/2}{2}\right) - x\left(\frac{N}{2}-n\right)\beta_1\left(\frac{n+N/2}{2}\right)\right]\beta\left((n+\frac{N}{2})k\right) \quad (10)$$

Using (7) and (8) for (*v*= 1), yields:

$$X_o(k) = \sum_{n=0}^{N/2-1}\left[x(n)\beta_1\left(\frac{n}{2}\right) + x(N-n)\beta_2\left(\frac{n}{2}\right)\right]\beta(nk) - \sum_{n=0}^{N/2-1}\left[x\left(n+\frac{N}{2}\right)\beta_2\left(\frac{n}{2}\right) - x\left(\frac{N}{2}-n\right)\beta_1\left(\frac{n}{2}\right)\right]\beta\left((n+\frac{N}{2})k\right) \quad (11)$$

Applying (6) into the second summation of (11), get:

$$\begin{aligned} X_o(k) = &\sum_{n=0}^{N/2-1}\left[x(n)\beta_1\left(\frac{n}{2}\right) + x(N-n)\beta_2\left(\frac{n}{2}\right)\right]\beta\binom{n}{k} \\ &- \sum_{\mathrm{n}=0}^{\mathrm{N}/2-1}\left[\mathrm{x}\left(\mathrm{n}+\frac{\mathrm{N}}{2}\right)\upbeta_2\left(\frac{\mathrm{n}}{2}\right) - \mathrm{x}\left(\frac{\mathrm{N}}{2}-\mathrm{n}\right)\upbeta_1\left(\frac{\mathrm{n}}{2}\right)\right]\upbeta(\mathrm{nk})\upbeta_1\left(\frac{\mathrm{N}}{2}\mathrm{k}\right) \\ &- \sum_{n=0}^{N/2-1}\left[x\left(n+\frac{N}{2}\right)\beta_2\left(\frac{n}{2}\right) - x\left(\frac{N}{2}-n\right)\beta_1\left(\frac{n}{2}\right)\right]\beta(-nk)\beta_2\left(\frac{N}{2}k\right) \end{aligned} \quad (12)$$

The second step of the development is by considering the even- and odd-frequency components separately, as:

a. The even-frequency components *X_o*(2*k*) can be obtained by using (8) for (*v*= 4) get:

$$X_o(2k) = \sum_{n=0}^{N/2-1}\left\{\begin{matrix}\left[x(n) + x(\frac{N}{2}-n)\right]\beta_1(\frac{n}{2}) - \\ \left[x(N-n) - x(n+\frac{N}{2})\right]\beta_2(\frac{n}{2})\end{matrix}\right\}\beta(2nk) \quad (13)$$

b. The odd-frequency components X_o (2*k*+1) can be obtained by applying (7) and (8) for (*v*=2) as (14):

$$\begin{aligned} X_o(2k+1) = &\sum_{n=0}^{N/2-1}\left[x(n)\beta_1\left(\frac{n}{2}\right) + x(N-n)\beta_2\left(\frac{n}{2}\right)\right]\beta(2nk+n) \\ &+ \sum_{n=0}^{N/2-1}\left[x\left(n+\frac{N}{2}\right)\beta_2\left(\frac{n}{2}\right) - x\left(\frac{N}{2}-n\right)\beta_1\left(\frac{n}{2}\right)\right]\beta(2nk+n) \end{aligned} \quad (14)$$

Applying (6) into (12) and rearranging, get:

$$\begin{aligned} X_o(2k+1) = &\sum_{n=0}^{N/2-1}\left[x(n) + x\left(\frac{N}{2}-n\right)\right]\left[\beta_1(n)\beta_1\left(\frac{n}{2}\right) + \beta_2(n)\beta_2\left(\frac{n}{2}\right)\right]\beta(2nk) \\ &+ \sum_{n=0}^{N/2-1}\left[x(N-n) + x\left(n+\frac{N}{2}\right)\right]\left[\beta_2(n)\beta_1\left(\frac{n}{2}\right) + \beta_1(n)\beta_2\left(\frac{n}{2}\right)\right]\beta(2nk) \end{aligned} \quad (15)$$

Therefore, the odd-frequency component X_o (2*k*+1) can be obtained by applying (7) into (15), yields:

$$X_o(2k+1) = \sum_{n=0}^{N/2-1}\left\{\begin{matrix}\left[x(n) - x(\frac{N}{2}-n)\right]\beta_1(\frac{n}{2}) + \\ \left[x(N-n) + x(n+\frac{N}{2})\right]\beta_2(\frac{n}{2})\end{matrix}\right\}\beta(2nk) \quad (16)$$

Combing four points together leads to the in-place butterfly of the DIF ONMNT algorithm shown in Figure 1.

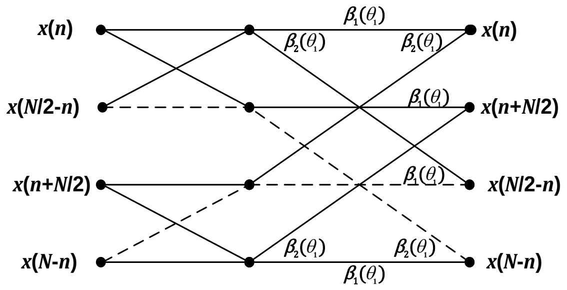

Figure 1. Butterfly of the ONMNT DIF algorithm, dotted and solid lines represent subtractions and additions respectively; θ_1=k/2 is the rotation angle

3.2. Development of fast O^{2}NMNT DIF algorithm

In this algorithm, the derivation begins by dividing the output sequence of $X_{oo}(k)$ in (5) into its even and odd parts, as (17) and (18):

$$X_{oo}(2k) = \sum_{n=0}^{N-1} x(n)\beta\left(\frac{(2n+1)((2k+1)2k)}{4}\right) = \sum_{n=0}^{N-1} x(n)\beta\left(\frac{(2n+1)(2k+1)}{4} + \frac{(2n+1)k}{2}\right) = \sum_{n=0}^{N-1} x(n)\beta(A) \quad (17)$$

$$X_{oo}(2k+1) = \sum_{n=0}^{N-1} x(n)\beta\left(\frac{(2n+1)(2(2k+1)+1)}{4}\right)$$
$$= \sum_{n=0}^{N-1} x(n)\beta\left(\frac{(2n+1)(2k+1)}{4} + \frac{(2n+1)(k+1)}{2}\right) = \sum_{n=0}^{N-1} x(n)\beta(B) \quad (18)$$

Define two auxiliary sequences $Y(k)$ and $Z(k)$ as (19):

$$Y(k) = (X_{oo}(2k+1) + X_{oo}(2k))/2$$
$$Z(k) = (X_{oo}(2k+1) - X_{oo}(2k))/2 \quad (19)$$

Also, from (6) the following relations can be obtained:

$$\beta(A) + \beta(B) = 2\beta_1(\frac{A-B}{2})\beta(\frac{A+B}{2})$$
$$\beta(A) - \beta(B) = 2\beta_2(\frac{A-B}{2})\beta(-\frac{A+B}{2}) \quad (20)$$

Replacing values of A and B in (17) and (18), yields:

$$A+B = \left(\frac{(2n+1)(2k+1)}{4} + \frac{(2n+1)k}{2}\right) + \left(\frac{(2n+1)(2k+1)}{4} + \frac{(2n+1)(k+1)}{2}\right) = (2n+1)(2k+1)$$
$$A-B = \left(\frac{(2n+1)(2k+1)}{4} + \frac{(2n+1)k}{2}\right) - \left(\frac{(2n+1)(2k+1)}{4} + \frac{(2n+1)(k+1)}{2}\right) = \frac{(2n+1)}{2} \quad (21)$$

Substituting (21) into (20) and then in (19), $Y(k)$ and $Z(k)$ can be written as (22) and (23):

$$Y(k) = \sum_{n=0}^{N-1} x(n)\beta_1\left(\frac{2n+1}{4}\right)\beta\left(\frac{(2n+1)(2k+1)}{2}\right) \quad (22)$$

$$Z(k) = \sum_{n=0}^{N-1} x(n)\beta_2\left(\frac{2n+1}{4}\right)\beta\left(-\frac{(2n+1)(2k+1)}{2}\right) \quad (23)$$

The second step of the algorithm's derivation is by dividing sequences $Y(k)$ and $Z(k)$, into two equal parts each having a length equals to N/2 as (14) and (25):

$$Y(k) = \sum_{n=0}^{N/2-1} x(n)\beta_1\left(\frac{2n+1}{4}\right)\beta\left(\frac{(2n+1)(2k+1)}{2}\right)$$
$$+\sum_{n=0}^{N/2-1} x\left(n+\frac{N}{2}\right)\beta_1\left(\frac{2n+1}{4}+\frac{N}{4}\right)\beta\left(\frac{(2n+1)(2k+1)}{2}+\frac{N}{2}\right) \quad (24)$$

$$Z(k) = \sum_{n=0}^{N/2-1} x(n)\beta_2\left(\frac{2n+1}{4}\right)\beta\left(\frac{(2n+1)(2k+1)}{2}\right)$$
$$+\sum_{n=0}^{N/2-1} x\left(n+\frac{N}{2}\right)\beta_2\left(\frac{2n+1}{4}+\frac{N}{4}\right)\beta\left(-\frac{(2n+1)(2k+1)}{2}-\frac{N}{2}\right) \tag{25}$$

Applying (8) for (v=1) into (24) and (25), get:

$$Y(k) = \sum_{n=0}^{N/2-1}\left[x(n)\beta_1\left(\frac{2n+1}{4}\right) + x(n+\frac{N}{2})\beta_2\left(\frac{2n+1}{4}\right)\right]\beta\left(\frac{(2n+1)(2k+1)}{2}\right) \tag{26}$$

$$Z(k) = \sum_{n=0}^{N/2-1}\left[x(N-n-1)\beta_2\left(\frac{2n+1}{4}\right) - x(\frac{N}{2}-n-1)\beta_1\left(\frac{2n+1}{4}\right)\right]\beta\left(\frac{(2n+1)(2k+1)}{2}\right) \tag{27}$$

From (19), $X_{oo}(2k)=Y(k)-Z(k)$ and is given by:

$$X_{oo}(2k) = \sum_{n=0}^{N/2-1}\left\{\begin{matrix}\left[x(n)+x(\frac{N}{2}-n-1)\right]\beta_1\left(\frac{2n+1}{4}\right)+\\ \left[x(n+\frac{N}{2})-x(N-n-1)\right]\beta_2\left(\frac{2n+1}{4}\right)\end{matrix}\right\}\beta\left(\frac{(2n+1)(2k+1)}{2}\right) \tag{28}$$

and the X_{oo} (2k+1)=$Y(k)$+$Z(k)$] and is given by:

$$X_{oo}(2k+1) = \sum_{n=0}^{N/2-1}\left\{\begin{matrix}\left[x(n)-x(\frac{N}{2}-n-1)\right]\beta_1\left(\frac{2n+1}{4}\right)+\\ \left[x(n+\frac{N}{2})+x(N-n-1)\right]\beta_2\left(\frac{2n+1}{4}\right)\end{matrix}\right\}\beta\left(\frac{(2n+1)(2k+1)}{2}\right) \tag{29}$$

Combining four points together gives the in-place butterfly shown in Figure 2.

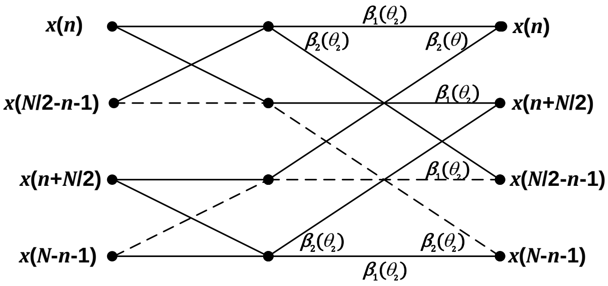

Figure 2. Butterfly of the O^{2}NMNT DIF algorithm, dotted and solid lines represent subtractions and additions respectively; θ_2=(2k+1)/4 is the rotation angle

3.3. Comparison between algorithms

In this section, the relationship between the structures of the developed algorithms is discussed by showing the similarities between butterflies of the developed algorithms. It is clearly noticed that the ONMNT butterfly shown in Figure 1 is almost identical to the O^{2}NMNT butterfly shown in Figure 2. Both butterflies have the same size; twiddle factors and structures. Moreover, both butterflies have the property of in-place computation and all rotation angles (θ_1 for ONMNT and θ_2 for O^{2}NMNT) of the multipliers needed to be computed or read from a lookup table in a specified stage of the ONMNT algorithm are also needed in the analogous stage of the O^{2}NMNT algorithm. The main difference between two butterflies is in the indexing scheme, as the butterfly shown in Figure 2 is shifted compared to the corresponding butterfly shown in Figure 1. This change in indexing is due to the existence of the frequency index (k_o) of the O^{2}NMNT given by (5), which does not exist in the ONMNT given by (4). It can be concluded that there are different indexing schemes in two algorithms. This is to be anticipated, since the ONMNT has only shift in time index (n_o) while the O^{2}NMNT has two shifts for each of the time and frequency indices. Making the indexing process for both algorithm identical, all data that are stored for normal indexing (i.e., $x(n)$ and $x(n+N/2)$) for the ONMNT algorithm are stored at the corresponding locations assigned for the O^{2}NMNT algorithm, and the remaining data that are stored for retrograde indexing [23] (i.e., $x(N-n)$ and $x(N/2-n)$) in the former are stored at the locations designated for the new retrograde indexing (i.e., $x(N-n-1)$ and $x(N/2-n-1)$) of the latter. Therefore, with this indexing adaptation, both butterflies shown in Figures 1 and 2 are identical. As a result, there is a close relationship between the ONMNT and O^{2}NMNT algorithms that should help in the development of a single software or hardware module for the implementation of these algorithms to compute the ONMNT as well as the O^{2}NMNT.

4. SKEW-CYCLIC CONVOLUTION OF GNMNTs

Both ONMNT and O^{2}NMNT have the skew-cyclic convolution (SCC) property and therefore can be utilized in a number of applications, such as for fast computation of linear convolution [8], [24]. The SCC of two signals *x(n)* and *h(n)* of length *N* denoted by $y_{SCC}(n)$ is defined as (30):

$$y_{SCC}(n) = \sum_{l=0}^{n} x(l)h(n-l) - \sum_{l=n+1}^{N-1} x(l)h(N+n-l) \tag{30}$$

Let $X_o(k)$ and $H_o(k)$ be the ONMNT of *x(n)* and *h(n)* and let $Y_{oo}(k)$, $X_{oo}(k)$ and $H_{oo}(k)$ be the O^{2}NMNT of $y_{SCC}(n)$, *x(n)* and *h(n)* respectively. The SCC property of GNMNTs is defined into two forms as (31) and (32) (see [6] for proof)

$$Y_{oo}(k) = X_{oo}(k)\Pi_1 H_o(k) = X_{oo}(k)H_o^{ev}(k) - X_{oo}(N-k-1)H_o^{od}(k) \tag{31}$$

$$Y_{oo}(k) = X_{oo}(k)\Pi_1 H_o(k) = X_{oo}(k)H_o^{ev}(k) - X_{oo}(N-k-1)H_o^{od}(k) \tag{32}$$

where $H_o^{ev}(k)$, $H_o^{od}(k)$ in (31) and $H_{oo}^{ev}(k)$, $H_{oo}^{od}(k)$ in (32) represent even and odd parts for $H_o(k)$ and $H_{oo}(k)$ respectively. Figures 3 and 4 illustrate the SCC calculations of the GNMNTs, where the operator $\boldsymbol{\Pi_1}$ and $\boldsymbol{\Pi_2}$ are as defined by (31) and (32) respectively.

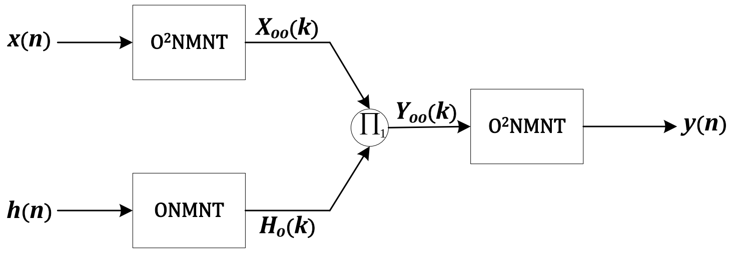

Figure 3. Fast skew-cyclic convolution using the GNMNTs based on (30)

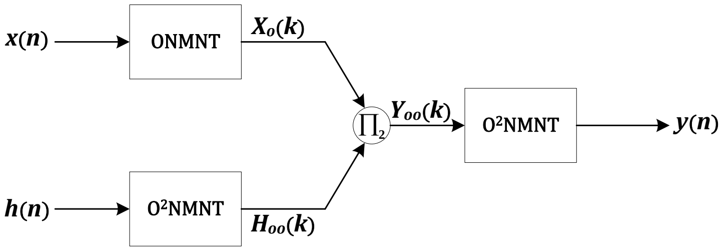

Figure 4. Fast skew-cyclic convolution using the GNMNTs based on (31)

4.1. Calculation of convolutions using GNMNTs

One of the most important applications of the GNMNTs is for fast computating of different kinds of convolutions to be used in digital filtering and other applications [25]. The linear convolution can be computed efficiently using both CC and SCC. For instance, let $y_{scc}(n)$ and $y_{cc}(n)$ be the output of the skew-cyclic and cyclic convolutions respectively, and let the output of the linear convolution is given as $y(m) = [y_1(n)\ y_2(n)]$ for $m = 0,1,\ldots,2N\text{-}1$, such that:

$$y(m) = \begin{cases} y_1(n) = \sum_{l=0}^{n} x(l)h(n-l) & 0 \le n \le N-1 \\ y_2(n) = \sum_{l=n+1}^{N-1} x(l)h(n+N-l) & N \le n \le 2N-1 \end{cases} \quad (33)$$

From the CC and SCC definitions, it follows that $y_1(n)$=[$y_{cc}(n)$+$y_{scc}(n)$]/2 and $y_2(n)$=[$y_{cc}(n)$- $y_{scc}(n)$]/2. Consequently, the CC can be calculated by mapping it to SCC (or conversely) based on the relationships between ONMNT and NMNT, which can then be computed by the fast convolution algorithm described by (31) and (32). An important aspect for this approach is that the conventional convolution can be computed via combining both SCC and CC circular convolutions each of length *N*, as a substitute of the traditional method based on stuffing zeros that requires length of 2*N* transforms.

Furthermore, there are savings in arithmetic complexity compared with the standard method. Let *A*(*N*) and *M*(*N*) be the number of additions and multiplications respectively required to compute the linear convolution. If the assumption is made that *h(n)* is known a priori, and consequently *H(k)* can be pre calculated and saved, then the complexity calculations of carried out as follows

a. For GNMNT, we need to calculate two O^{2}NMNTs of length *N* to obtain the SCC and CC, plus 2*N* multiplications and *N* additions to compute $\prod$'s , as given in (31) and (32). Also, 2*N* additions are required to calculate *y(n)* from $y_1(n)$ and $y_2(n)$, as given in (33). Therefore, for this method *A(N)*= 4$A_o(N)$+4*N* and *M(N)*=4$M_o(N)$+4*N*. where $A_o(N)$ and $M_o(N)$ are the number of additions and multiplications for the O^{2}NMNT respectively.
b. For the conventional method [12], we need to calculate one NMNT of length 2*N* to get the CC, plus 4*N* multiplications and 2*N* additions for [both forward and inverse transforms. Therefore for this method *A(N)*= 2$A_N(2N)$+2*N* and *M(N)*=2$M_N(2N)$+2*N*. Where $A_N(N)$ and $M_N(N)$ are the number additions and of multiplications for the NMNT respectively.

Assuming that the transforms are implemented based on single butterfly implementations that require $Nlog_2N$ multiplications and (3*N*/2)log_2N additions [5], we get arithmetic complexity based on GNMNTs *M*(*N*)= 4$Nlog_2N$+4*N* and *A*(*N*)=6$Nlog_2N$+4*N* while the arithmetic complexity based on NMNT is equal to *A*(*N*)= 6$Nlog_2N$+6*N* and *A*(*N*)=6$Nlog_2N$+8*N*. Therefore, it clear that the calculations of the linear convolution based on GNMNTs method require 2*N* fewer multiplications and 4*N* fewer additions than the conventional method.

The following example illustrate the utilization of the GNMNTs for the calculations of cyclic, skew-cyclic and linear convolutions. For the sake of validation and without losing of generality, it is necessary to calculate these convolutions for the following 8-point two integer data *x*(*n*) and *h*(*n*) generated randomly:

x(*n*)=[11 4 12 19 29 3 19]

h(*n*)=[22 19 13 5 11 9 7 2]

Choosing *Mp*=2^7-1=127, will be enough to calculate these convolutions without overflow. From Table 1, with the transform length *N*=8 and *d*=32, values of β_1 and β_2 are 119 and 119 respectively. For ONMNT, the parameters can be calculated for *d*=16, therefore values of β_1(1/2) and β_2(1/2) are 106 and 103 respectively. Using these parameters, the ONMNT transformed sequences $X_o(k)$ and $H_o(k)$ for the *x*(*n*) and *h*(*n*) respectively, are given as:

$X_o(k)$ =[35 7 89 7 42 49 121 119]

$H_o(k)$=[36 64 120 94 122 115 108 25]

Similarly, For O^{2}NMNT, the parameters can be calculated for *d*=8, therefore values of β_1(1/4) and β_2(1/4) are 102 and 97 respectively. Using these parameters, the O^{2}NMNT transformed sequences $X_{oo}(k)$ and $H_{oo}(k)$ for the *x*(*n*) and *h*(*n*) respectively, are given as:

$X_{oo}(k)$ =[0 18 93 10 69 99 74 20]

$H_{oo}(k)$ =[1 91 80 99 40123 85 53]

Firstly, the SCC of *x*(*n*) and *h*(*n*) can be computed using the O^{2}NMNT convolution property, The output of skew convolution will be:

$y_{scc}(n)$=[4 42 104 94 12 19 6 73]

Secondly, the CC of $x(n)$ and $h(n)$ can be computed using the NMNT convolution property [10]. The output of cyclic convolution will be:

$y_{cc}(n)$=[99 44 100 15 40 83 82 73]

Finally, the linear convolution of these sequences is computed from $y_{scc}(n)$ and $y_{cc}(n)$ as given in section 4.1; the desired convolution result $y(n)$ is:

$y(n)$=[115 43 102 118 26 51 44 73 111 1 125 24 14 32 38]

Therefore, we can clearly observe that the advantage of using the GNMNTs for computing the error-free convolutions is that the linear convolution of length 2N-1 can be calculated by N-point circularly convolution sequences using transforms lengths N-point, without any reduction in the dynamic range.

5. CONCLUSIONS

In this project, two new algorithms based on decimation-in-frequency approach and proper finite field Cooley-Tukey relations for computing fast ONMNT and O^{2}NMNT transforms have been developed. Additionally, a unified approach has been proposed to illustrate that there have existence of a closed relation between the developed GNMNTs algorithms. It is obvious from this relationship that both algorithms have a very close structure and the identical number of stages are required. Consequently, the ONMNT butterfly can also be used for computing the O^{2}NMNT by applying an appropriate indexing scheme. However, the multipliers have to be calculated or loaded from a lookup table in a specified stage of the ONMNT algorithms are saved in different locations for the corresponding stage of the O^{2}NMNT algorithm. As a result, it has shown that just a single integrated butterfly structure is enough to implement each of the developed algorithms. Therefore, a combined software or hardware module can be used to calculate the ONMNT as well as the O^{2}NMNT. Finally, since the NMNT is a special case of the GNMNTs, therefore the similar module can be used to cover all applications that involve NMNT or GNMNTs.

REFERENCES

[1] V. Britanak and K. R. Rao, “The fast generalized discrete Fourier transforms: A unified approach to the discrete sinusoidal transforms computation,” *Signal Processing*, vol. 79, no. 2, pp. 135–150, Dec. 1999, doi: 10.1016/S0165-1684(99)00088-2.

[2] Neng-Chung Hu, Hong-I Chang, and O. K. Ersoy, “Generalized discrete Hartley transforms,” *IEEE Transactions on Signal Processing*, vol. 40, no. 12, pp. 2931–2940, 1992, doi: 10.1109/78.175737.

[3] Z. Wang and B. R. Hunt, “The discrete W transform,” *Applied Mathematics and Computation*, vol. 16, no. 1, pp. 19–48, Jan. 1985, doi: 10.1016/0096-3003(85)90008-6.

[4] Z. Wang, G. A. Jullien, and W. C. Miller, “The generalized discrete W transform and its application to interpolation,” *Signal Processing*, vol. 36, no. 1, pp. 99–109, Mar. 1994, doi: 10.1016/0165-1684(94)90181-3.

[5] K. R. Rao and P. Yip, *Discrete Cosine Transform: Algorithms, Advantages, Applications*. San Diego: Academic Press Professional, Inc., 1990.

[6] Z. Hua, B. Zhou, and Y. Zhou, “Sine-Transform-Based Chaotic System With FPGA Implementation,” *IEEE Transactions on Industrial Electronics*, vol. 65, no. 3, pp. 2557–2566, Mar. 2018, doi: 10.1109/TIE.2017.2736515.

[7] O. K. Ersoy, *Fourier-Related Transforms, Fast Algorithms and Applications*. New Jersey, United States: Prentice-Hall, Inc., 1997.

[8] S. A. Martucci, “Symmetric convolution and the discrete sine and cosine transforms,” *IEEE Transactions on Signal Processing*, vol. 42, no. 5, pp. 1038–1051, May 1994, doi: 10.1109/78.295213.

[9] S. Boussakta and A. G. J. Holt, “New number theoretic transform,” *Electronics Letters*, vol. 28, no. 18, pp. 1683–1684, 1992, doi: 10.1049/el:19921070.

[10] S. Boussakta, “New transform using the Mersenne numbers,” *IEE Proceedings - Vision, Image, and Signal Processing*, vol. 142, no. 6, pp. 381–388, 1995, doi: 10.1049/ip-vis:19952323.

[11] M. T. Hamood and S. Boussakta, “Efficient algorithms for computing the new Mersenne number transform,” *Digital Signal Processing*, vol. 25, pp. 280–288, Feb. 2014, doi: 10.1016/j.dsp.2013.10.018.

[12] S. Boussakta and A. G. J. Holt, “Number Theoretic Transforms and their Applications in Image Processing,” *Advances in Imaging and Electron Physics*, vol. 111, no. C, pp. 1–90, 1999, doi: 10.1016/S1076-5670(08)70216-7.

[13] Y. Al-Aali and S. Boussakta, “Lightweight Hash Function Based on the New Mersenne Number Transform Family,” in *2022 7th International Conference on Frontiers of Signal Processing (ICFSP)*, Sep. 2022, pp. 179–183, doi: 10.1109/ICFSP55781.2022.9924893.

[14] M. F. Al-Gailani and S. Boussakta, “Evaluation of one-dimensional NMNT for security applications,” in *2010 7th International Symposium on Communication Systems, Networks & Digital Signal Processing (CSNDSP 2010)*, Jul. 2010, pp. 715–720, doi: 10.1109/CSNDSP16145.2010.5580331.

[15] D. Kehil and Y. Ferdi, “Signal encryption using new Mersenne number transform,” in *2010 7th International Symposium on Communication Systems, Networks & Digital Signal Processing (CSNDSP 2010)*, Jul. 2010, pp. 736–740, doi: 10.1109/CSNDSP16145.2010.5580327.

[16] S. Boussakta, M. T. Hamood, and N. Rutter, “Generalized New Mersenne Number Transforms,” *IEEE Transactions on Signal Processing*, vol. 60, no. 5, pp. 2640–2647, May 2012, doi: 10.1109/TSP.2012.2186131.

[17] N. Rutter, “Implementation and Analysis of the Generalised New Mersenne Number Transforms for Encryption,” Newcastle

University, Newcastle upon Tyne, UK, 2015.

[18] H. H. ALAEDDINE, O. BAZZI, A. H. ALAEDDINE, Y. MOHANNA, and G. BUREL, "Fast Convolution Using Generalized Sliding Fermat Number Transform with Application to Digital Filtering," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E95.A, no. 6, pp. 1007–1017, 2012, doi: 10.1587/transfun.E95.A.1007.

[19] S.-C. Pei, C.-C. Wen, and J.-J. Ding, "Closed-Form Orthogonal Number Theoretic Transform Eigenvectors and the Fast Fractional NTT," *IEEE Transactions on Signal Processing*, vol. 59, no. 5, pp. 2124–2135, May 2011, doi: 10.1109/TSP.2011.2113176.

[20] H. Krishna, *Digital signal processing algorithms: Number theory, convolution, fast Fourier transforms, and applications*. Florida: Routledge, 2000, doi: 10.1016/s0898-1221(98)91145-2.

[21] G. Bi and Y. Chen, "Fast generalized DFT and DHT algorithms," *Signal Processing*, vol. 65, no. 3, pp. 383–390, Mar. 1998, doi: 10.1016/S0165-1684(97)00234-X.

[22] G. Bongiovanni, P. Corsini, and G. Frosini, "One-dimensional and two-dimensional generalised discrete fourier transforms," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 24, no. 1, pp. 97–99, Feb. 1976, doi: 10.1109/TASSP.1976.1162764.

[23] O. Nibouche, S. Boussakta, and M. Darnell, "Pipeline Architectures for Radix-2 New Mersenne Number Transform," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 56, no. 8, pp. 1668–1680, Aug. 2009, doi: 10.1109/TCSI.2008.2008266.

[24] M. J. Narasimha, "Linear Convolution Using Skew-Cyclic Convolutions," *IEEE Signal Processing Letters*, vol. 14, no. 3, pp. 173–176, Mar. 2007, doi: 10.1109/LSP.2006.884034.

[25] Ersoy, "Semisystolic Array Implementation of Circular, Skew Circular, and Linear Convolutions," *IEEE Transactions on Computers*, vol. C–34, no. 2, pp. 190–196, Feb. 1985, doi: 10.1109/TC.1985.1676558.

BIOGRAPHIES OF AUTHORS

Lujain S. Abdulla received B.Sc. degree in Electrical Engineering from Tikrit University, Iraq 2004. M.Sc. degree in Communication Engineering from University of Mosul, Iraq 2010. In 2004 he had worked at Tikrit University, Iraq as an engineer in Electrical Engineering Laboratories. In 2010 he had worked as a lecturer in Department of Engineering Electrical, Tikrit University Faculty of Iraq and currently he still. He has many research papers interests' communication engineering, mobile communication, computer networks and encryption. He can be contacted at email: lujainsabah@tu.edu.iq.

Abdulmutalib A-Wahab Hussein received B.Sc. degree in Electrical Engineering from Tikrit University, Iraq 1997. M.Sc. Degree in Communication Engineering from University of Mosul, Iraq 2009. In 2005 he had worked at Tikrit University, Iraq as an Engineer in Electrical Engineering Laboratories. In 2009 he had worked as a lecturer in Department of Engineering Electrical, Tikrit University Faculty of Iraq and currently he still. His interests are in communication engineering, mobile communication and computer networks. He can be contacted at email: abdulmuttalib.a.hussein@tu.edu.iq.

Mounir Taha Hamood received the B.Sc. degree in Electrical Engineering from University of Technology, Baghdad, Iraq, in 1990 and the M.Sc. degree in Electronic and Communications Engineering from Al-Nahrain University, Baghdad, Iraq, in 1995. He graduated from Newcastle University, Newcastle upon Tyne, UK in 2012 with the Ph.D. degree in Communications and Signal Processing. His doctoral research was in the development of efficient algorithms for fast computation of discrete transforms. From 2012 to 2016, he was a lecturer in Signal Processing for Communication at Tikrit University. He is currently working as the head of Electrical department and a professor of signal processing at the Department of Electrical Engineering, College of Engineering, Tikrit University, Tikrit, Iraq. His research interest includes discrete transforms, fast algorithms for digital signal processing in one and multidimensional applications, and communication systems. He can be contacted at email: m.t.hamood@tu.edu.iq.