On semifast Fourier transform algorithms

Sergei V. Fedorenko
Information systems security department
St.Petersburg State University of Aerospace Instrumentation
190000, Bolshaya Morskaia, 67, St.Petersburg, Russia

Abstract
In this paper, following [1, 2, 3, 4, 5, 6, 7] we consider the relations between well-known Fourier transform algorithms.

1 Introduction

A semifast algorithm for a computation is an algorithm that significantly reduces the number of multiplications compared with the natural form of the computation, but does not reduce the number of additions. A semifast Fourier transform in $GF(q)$ is a computational procedure for computing the $n$-point Fourier transform in $GF(q)$ that uses about $n \log n$ multiplications in $GF(q)$ and about $n^2$ additions in $GF(q)$ [4].

The $n$-point Fourier transform over $GF(2^m)$ is defined by

$$F = W f,$$

where

$$W = \begin{bmatrix}
\alpha^0 & \alpha^0 & \alpha^0 & \cdots & \alpha^0 \\
\alpha^0 & \alpha^1 & \alpha^2 & \cdots & \alpha^{n-1} \\
(\alpha^0)^2 & (\alpha^1)^2 & (\alpha^2)^2 & \cdots & (\alpha^{n-1})^2 \\
\vdots & \vdots & \vdots & \cdots & \vdots \\
(\alpha^0)^{n-1} & (\alpha^1)^{n-1} & (\alpha^2)^{n-1} & \cdots & (\alpha^{n-1})^{n-1}
\end{bmatrix},$$

where $W = (\alpha^{ij})$, $i, j \in [0, n-1]$, is a Vandermonde matrix, an element $\alpha \in GF(2^m)$ of order $\text{ord}(\alpha) = n | 2^m - 1$ is the Fourier transform kernel.

Consider a few well-known Fourier transform algorithms [1, 2, 3, 4, 5, 6, 7] on examples.

The finite field $GF(2^3)$ is defined by an element $\alpha$, which is a root of the primitive polynomial $x^3 + x + 1$. Cyclotomic cosets modulo 7 over
The finite field \( GF(2) \) are \((0), (1, 2, 4), (3, 6, 5) \). The binary conjugacy classes of \( GF(2^3) \) are \((\alpha^0), (\alpha^1, \alpha^2, \alpha^4), (\alpha^3, \alpha^6, \alpha^8) \). The standard basis of \( GF(2^3) \) is \((\alpha^0, \alpha^1, \alpha^2) \). The normal basis of \( GF(2^3) \) is \((\beta^1, \beta^2, \beta^4) \), where \( \beta = \alpha^3 \). Another normal basis of \( GF(2^3) \) is the cyclic shift of the previous normal basis: \((\gamma^1, \gamma^2, \gamma^4) \), where \( \gamma = \alpha^6 \).

Consider the Fourier transform of length \( n = 7 \) over the field \( GF(2^3) \). Let us take the primitive element \( \alpha \) as the kernel of the Fourier transform. The Fourier transform of a polynomial \( f(x) = \sum_{i=0}^{6} f_i x^i \) consists of components \( F_i = f(\alpha^i), \ i \in [0, 6] \).

Thus,

\[
\begin{pmatrix}
F_0 \\
F_1 \\
F_2 \\
F_3 \\
F_4 \\
F_5 \\
F_6
\end{pmatrix} = \begin{pmatrix}
\alpha^0 & \alpha^0 & \alpha^0 & \alpha^0 & \alpha^0 & \alpha^0 & \alpha^0 \\
\alpha^0 & \alpha^1 & \alpha^2 & \alpha^3 & \alpha^4 & \alpha^5 & \alpha^6 \\
\alpha^0 & \alpha^2 & \alpha^4 & \alpha^6 & \alpha^1 & \alpha^3 & \alpha^5 \\
\alpha^0 & \alpha^3 & \alpha^6 & \alpha^2 & \alpha^5 & \alpha^1 & \alpha^4 \\
\alpha^0 & \alpha^4 & \alpha^1 & \alpha^5 & \alpha^2 & \alpha^6 & \alpha^3 \\
\alpha^0 & \alpha^5 & \alpha^3 & \alpha^1 & \alpha^6 & \alpha^4 & \alpha^2 \\
\alpha^0 & \alpha^6 & \alpha^5 & \alpha^4 & \alpha^3 & \alpha^2 & \alpha^1
\end{pmatrix} \begin{pmatrix}
f_0 \\
f_1 \\
f_2 \\
f_3 \\
f_4 \\
f_5 \\
f_6
\end{pmatrix}.
\]

2 Goertzel’s algorithm (1958)

Blahut’s modification for finite fields (1983)

The first step of the Goertzel algorithm is a long division \( f(x) \) by every minimal polynomial:

\[
f(x) = (x + 1) q_0(x) + r_{00} \\
f(x) = (x^3 + x + 1) q_1(x) + r_{01}, \quad r_1(x) = r_{21} x^2 + r_{11} x + r_{01},
\]

\[
f(x) = (x^3 + x^2 + 1) q_2(x) + r_{02}, \quad r_2(x) = r_{22} x^2 + r_{12} x + r_{02}
\]

where

\[
\begin{align*}
r_{00} &= f_0 \\
r_{01} &= f_0 + f_3 + f_5 + f_6 \\
r_{11} &= f_1 + f_3 + f_4 + f_5 \\
r_{21} &= f_2 + f_4 + f_5 + f_6 \\
r_{02} &= f_0 + f_3 + f_4 + f_5 \\
r_{12} &= f_1 + f_4 + f_5 + f_6 \\
r_{22} &= f_2 + f_4 + f_5 + f_6.
\end{align*}
\]

The second step of the Goertzel algorithm is an evaluated remainder in every element of the finite field:
\[
F_0 = f(\alpha^0) = r_{00} \\
F_1 = f(\alpha^1) = r_1(\alpha^1) = r_{21}\alpha^2 + r_{11}\alpha^1 + r_{01} \\
F_2 = f(\alpha^2) = r_1(\alpha^2) = r_{21}\alpha^4 + r_{11}\alpha^2 + r_{01} \\
F_3 = f(\alpha^3) = r_2(\alpha^3) = r_{22}\alpha^6 + r_{12}\alpha^3 + r_{02} \\
F_4 = f(\alpha^4) = r_1(\alpha^4) = r_{21}\alpha^6 + r_{11}\alpha^4 + r_{01} \\
F_5 = f(\alpha^5) = r_2(\alpha^5) = r_{22}\alpha^6 + r_{12}\alpha^5 + r_{02}
\]

or
\[
\begin{pmatrix}
F_1 \\
F_2 \\
F_3 \\
F_4 \\
F_5
\end{pmatrix} = \begin{pmatrix}
1 & \alpha^1 & \alpha^2 \\
1 & \alpha^2 & \alpha^4 \\
\alpha^0 & \alpha^2 & \alpha^4 \\
\alpha^0 & \alpha^4 & \alpha^1 \\
\alpha^0 & \alpha^3 & \alpha^6
\end{pmatrix} \begin{pmatrix}
r_{01} \\
r_{11} \\ r_{21}
\end{pmatrix},
\begin{pmatrix}
F_3 \\
F_6 \\
F_5
\end{pmatrix} = \begin{pmatrix}
1 & \alpha^3 & \alpha^6 \\
1 & \alpha^6 & \alpha^5 \\
\alpha^0 & \alpha^5 & \alpha^3
\end{pmatrix} \begin{pmatrix}
r_{02} \\
r_{12} \\ r_{22}
\end{pmatrix}.
\]

Thus,
\[
\begin{pmatrix}
F_0 \\
F_1 \\
F_2 \\
F_3 \\
F_4 \\
F_5
\end{pmatrix} = \begin{pmatrix}
\alpha^0 \\
\alpha^0 & \alpha^1 & \alpha^2 \\
\alpha^0 & \alpha^2 & \alpha^4 \\
\alpha^0 & \alpha^4 & \alpha^1 \\
\alpha^0 & \alpha^3 & \alpha^6 \\
\alpha^0 & \alpha^6 & \alpha^5 \\
\alpha^0 & \alpha^5 & \alpha^3
\end{pmatrix} \begin{pmatrix}
1 & 1 & 1 & 1 & 1 & 1 \\
1 & 0 & 0 & 1 & 0 & 1 \\
0 & 1 & 0 & 1 & 1 & 0 \\
0 & 0 & 1 & 0 & 1 & 1 \\
1 & 0 & 0 & 1 & 1 & 0 \\
0 & 1 & 0 & 0 & 1 & 1 \\
0 & 0 & 1 & 1 & 1 & 0
\end{pmatrix} \begin{pmatrix}
f_0 \\
f_1 \\
f_2 \\
f_3 \\
f_4 \\
f_5 \\
f_6
\end{pmatrix}.
\]

3 Blahut’s algorithm (2008)

We have
\[
\begin{pmatrix}
F_0 \\
F_1 \\
F_2 \\
F_3 \\
F_4 \\
F_5 \\
F_6
\end{pmatrix} = \begin{pmatrix}
\alpha^0 & \alpha^0 & \alpha^0 & \alpha^0 & \alpha^0 & \alpha^0 & \alpha^0 \\
\alpha^0 & \alpha^1 & \alpha^2 & \alpha^3 & \alpha^4 & \alpha^5 & \alpha^6 \\
\alpha^0 & \alpha^2 & \alpha^4 & \alpha^6 & \alpha^1 & \alpha^3 & \alpha^5 \\
\alpha^0 & \alpha^3 & \alpha^6 & \alpha^2 & \alpha^5 & \alpha^1 & \alpha^4 \\
\alpha^0 & \alpha^4 & \alpha^1 & \alpha^5 & \alpha^2 & \alpha^6 & \alpha^3 \\
\alpha^0 & \alpha^5 & \alpha^3 & \alpha^1 & \alpha^6 & \alpha^4 & \alpha^2 \\
\alpha^0 & \alpha^6 & \alpha^5 & \alpha^4 & \alpha^3 & \alpha^2 & \alpha^1
\end{pmatrix} \begin{pmatrix}
(f_0) \\
0 \\
0 \\
0 \\
0 \\
0 \\
0
\end{pmatrix} + \begin{pmatrix}
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1
\end{pmatrix} (f_0)
\]

\[
= \begin{pmatrix}
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1
\end{pmatrix} (f_0) + \begin{pmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1 \\
1 & 1 & 0 \\
0 & 1 & 1 \\
1 & 1 & 1 \\
1 & 0 & 1
\end{pmatrix} \begin{pmatrix}
\alpha^0 & \alpha^0 & \alpha^0 \\
\alpha^1 & \alpha^2 & \alpha^4 \\
\alpha^2 & \alpha^4 & \alpha^1
\end{pmatrix} \begin{pmatrix}
f_1 \\
f_2 \\
f_4
\end{pmatrix} + \begin{pmatrix}
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1
\end{pmatrix} (f_0)
\]
A polynomial \( f(x) = \sum_{i=0}^{6} f_i x^i \), \( l_i \in GF(2^m) \), is called a linearized polynomial. Any polynomial can be decomposed into a sum of linearized polynomials and a free term. A polynomial \( f(x) = \sum_{i=0}^{6} f_i x^i \) can be represented as

\[
\begin{align*}
    f(x) &= L_0(x^0) + L_1(x^1) + L_2(x^3) \\
    L_0(y) &= f_0 \\
    L_1(y) &= f_1 y + f_2 y^2 + f_4 y^4 \\
    L_2(y) &= f_3 y + f_6 y^2 + f_5 y^4.
\end{align*}
\]

We have

\[
\begin{align*}
    f(\alpha^0) &= L_0(\alpha^0) + L_1(\alpha^0) + L_2(\alpha^0) \\
    f(\alpha^1) &= L_0(\alpha^0) + L_1(\alpha^0) + L_2(\alpha^3) = L_0(1) + L_1(\alpha) + L_2(1) + L_2(\alpha) \\
    f(\alpha^2) &= L_0(\alpha^0) + L_1(\alpha^1) + L_2(\alpha^6) = L_0(1) + L_1(\alpha^2) + L_2(1) + L_2(\alpha^2) \\
    f(\alpha^3) &= L_0(\alpha^0) + L_1(\alpha^3) + L_2(\alpha^2) = L_0(1) + L_1(1) + L_1(\alpha) + L_2(\alpha^2) \\
    f(\alpha^4) &= L_0(\alpha^0) + L_1(\alpha^4) + L_2(\alpha^5) = L_0(1) + L_1(1) + L_1(\alpha^2) + L_2(1) + L_2(\alpha) + L_2(\alpha^2) \\
    f(\alpha^5) &= L_0(\alpha^0) + L_1(\alpha^5) + L_2(1) = L_0(1) + L_1(1) + L_1(\alpha) + L_1(\alpha^2) + L_2(\alpha) \\
    f(\alpha^6) &= L_0(\alpha^0) + L_1(\alpha^6) + L_2(1) = L_0(1) + L_1(1) + L_1(\alpha^2) + L_2(\alpha) + L_2(\alpha^2).
\end{align*}
\]

Using \( F_i = f(\alpha^i) \), these equations can be represented in a matrix form as
of a circulant are matrices, the circulant is referred to as a block circulant. From the preceding row by a left (right) cyclic shift by one position. If entries

\[
\begin{pmatrix}
F_0 \\
F_1 \\
F_2 \\
F_3 \\
F_4 \\
F_5 \\
F_6
\end{pmatrix} = \begin{pmatrix}
1 & 1 & 0 & 0 & 1 & 0 & 0 \\
1 & 0 & 1 & 0 & 1 & 1 & 0 \\
1 & 0 & 0 & 1 & 1 & 0 & 1 \\
1 & 1 & 1 & 0 & 0 & 0 & 1 \\
1 & 0 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 0 & 1 & 0 \\
1 & 1 & 0 & 1 & 0 & 1 & 1
\end{pmatrix} \begin{pmatrix}
\alpha^0 & \alpha^0 & \alpha^0 \\
\alpha^1 & \alpha^2 & \alpha^4 \\
\alpha^2 & \alpha^4 & \alpha^1
\end{pmatrix} \begin{pmatrix}
f_0 \\
f_1 \\
f_2 \\
f_3 \\
f_4 \\
f_5 \\
f_6
\end{pmatrix}.
\]

5 Trifonov–Fedorenko’s algorithm (2003)

A circulant matrix, or a circulant, is a matrix each row of which is obtained from the preceding row by a left (right) cyclic shift by one position. If entries of a circulant are matrices, the circulant is referred to as a block circulant. We call a circulant where the first row is a normal basis, a basis circulant.

Let \((\beta^1, \beta^2, \beta^4) = (\alpha^3, \alpha^6, \alpha^5)\) be a normal basis for \(GF(2^3)\).

Using

\[
\begin{pmatrix}
\alpha^0 & \alpha^0 & \alpha^0 \\
\alpha^1 & \alpha^2 & \alpha^4 \\
\alpha^2 & \alpha^4 & \alpha^1
\end{pmatrix} = \begin{pmatrix}
1 & 1 & 1 \\
0 & 1 & 1 \\
1 & 0 & 1
\end{pmatrix} \begin{pmatrix}
\beta^1 & \beta^2 & \beta^4 \\
\beta^2 & \beta^4 & \beta^1 \\
\beta^4 & \beta^1 & \beta^2
\end{pmatrix},
\]

we get

\[
\begin{pmatrix}
F_0 \\
F_1 \\
F_2 \\
F_3 \\
F_4 \\
F_5 \\
F_6
\end{pmatrix} = \begin{pmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 0 & 1 & 1 & 1 & 0 & 0 \\
1 & 1 & 0 & 1 & 1 & 0 & 1 \\
1 & 1 & 0 & 0 & 1 & 1 & 1 \\
1 & 1 & 1 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 1 & 0 & 1 & 1 \\
1 & 0 & 1 & 0 & 1 & 1 & 0
\end{pmatrix} \begin{pmatrix}
\beta^1 & \beta^2 & \beta^4 \\
\beta^2 & \beta^4 & \beta^1 \\
\beta^4 & \beta^1 & \beta^2
\end{pmatrix} \begin{pmatrix}
f_0 \\
f_1 \\
f_2 \\
f_3 \\
f_4 \\
f_5 \\
f_6
\end{pmatrix}.
\]

6 Fedorenko’s algorithm (1) (2006)

Combining

\[
\begin{pmatrix}
\alpha^3 & \alpha^6 & \alpha^5 \\
\alpha^6 & \alpha^5 & \alpha^3 \\
\alpha^5 & \alpha^3 & \alpha^6
\end{pmatrix} = \begin{pmatrix}
\beta^1 & \beta^2 & \beta^4 \\
\beta^2 & \beta^4 & \beta^1 \\
\beta^4 & \beta^1 & \beta^2
\end{pmatrix}, \quad \begin{pmatrix}
\alpha^1 & \alpha^2 & \alpha^4 \\
\alpha^2 & \alpha^4 & \alpha^1 \\
\alpha^4 & \alpha^1 & \alpha^2
\end{pmatrix} = \begin{pmatrix}
0 & 1 & 1 \\
1 & 0 & 1 \\
1 & 1 & 0
\end{pmatrix} \begin{pmatrix}
\beta^1 & \beta^2 & \beta^4 \\
\beta^2 & \beta^4 & \beta^1 \\
\beta^4 & \beta^1 & \beta^2
\end{pmatrix}.
\]
\[
\begin{bmatrix}
\alpha^2 & \alpha^4 & \alpha^1 \\
\alpha^4 & \alpha^1 & \alpha^2 \\
\alpha^1 & \alpha^2 & \alpha^4
\end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} \beta^1 & \beta^2 & \beta^4 \\ \beta^2 & \beta^4 & \beta^1 \\ \beta^4 & \beta^1 & \beta^2 \end{bmatrix},
\]
we obtain the equivalent Fourier transform
\[
\begin{bmatrix}
F_0 \\
F_1 \\
F_2 \\
F_4 \\
F_3 \\
F_6 \\
F_5
\end{bmatrix} = \begin{bmatrix}
\alpha^0 & \alpha^1 & \alpha^2 & \alpha^3 & \alpha^4 & \alpha^5 & \alpha^6 \\
\alpha^0 & \alpha^2 & \alpha^4 & \alpha^6 & \alpha^1 & \alpha^3 & \alpha^5 \\
\alpha^0 & \alpha^4 & \alpha^1 & \alpha^3 & \alpha^2 & \alpha^5 & \alpha^6 \\
\alpha^0 & \alpha^6 & \alpha^5 & \alpha^3 & \alpha^4 & \alpha^1 & \alpha^2 \\
\alpha^0 & \alpha^3 & \alpha^6 & \alpha^2 & \alpha^4 & \alpha^1 & \alpha^5 \\
\alpha^0 & \alpha^5 & \alpha^3 & \alpha^1 & \alpha^2 & \alpha^4 & \alpha^6 \\
\alpha^0 & \alpha^1 & \alpha^2 & \alpha^3 & \alpha^4 & \alpha^5 & \alpha^6
\end{bmatrix} \begin{bmatrix}
f_0 \\
f_1 \\
f_2 \\
f_4 \\
f_3 \\
f_6 \\
f_5
\end{bmatrix} =
\begin{bmatrix}
1 & 0 & 1 & 1 & 1 & 1 & 1 \\
1 & 0 & 1 & 1 & 0 & 0 & 0 \\
1 & 1 & 1 & 0 & 0 & 1 & 1 \\
1 & 1 & 0 & 0 & 1 & 0 & 1 \\
1 & 0 & 1 & 0 & 1 & 1 & 0 \\
1 & 0 & 0 & 1 & 0 & 1 & 1
\end{bmatrix} \begin{bmatrix}
\beta^1 & \beta^2 & \beta^4 \\
\beta^2 & \beta^4 & \beta^1 \\
\beta^4 & \beta^1 & \beta^2 \\
\beta^1 & \beta^2 & \beta^4 \\
\beta^2 & \beta^1 & \beta^2 \\
\beta^1 & \beta^1 & \beta^1
\end{bmatrix} \begin{bmatrix}
f_0 \\
f_1 \\
f_2 \\
f_3 \\
f_6 \\
f_5
\end{bmatrix}.
\]

The equivalent Fourier transform \( F_e = W_e f_e \) has the following structure:

\[
F_e = W_e f_e = A_e D_e f_e = \begin{bmatrix}
A_{11} & \ldots & A_{1l} \\
\ldots & \ldots & \ldots \\
A_{l1} & \ldots & A_{ll}
\end{bmatrix} \begin{bmatrix}
C_1 & 0 & \ldots & 0 \\
0 & C_2 & \ldots & 0 \\
0 & 0 & \ldots & C_l
\end{bmatrix} f_e, \quad (1)
\]

where \( A_e \) is a binary matrix,
\( A_e \) consists of binary circulants \( A_{ij} \),
\( D_e \) is a block diagonal matrix,
\( D_e \) consists of basis circulant matrices \( C_i \),
\( l \) is the number of cyclotomic cosets modulo \( n \) over \( GF(2) \).

7 Fedorenko’s algorithm (2) (2006)

Let \( (\gamma^1, \gamma^2, \gamma^4) = (\alpha^6, \alpha^5, \alpha^3) \) be a normal basis for \( GF(2^3) \).
Combining
\[
\begin{bmatrix}
\alpha^6 & \alpha^5 & \alpha^3 \\
\alpha^5 & \alpha^3 & \alpha^6 \\
\alpha^3 & \alpha^6 & \alpha^5
\end{bmatrix} = \begin{bmatrix}
\gamma^1 & \gamma^2 & \gamma^4 \\
\gamma^2 & \gamma^4 & \gamma^1 \\
\gamma^4 & \gamma^1 & \gamma^2
\end{bmatrix}, \quad \begin{bmatrix}
\alpha^1 & \alpha^2 & \alpha^4 \\
\alpha^2 & \alpha^4 & \alpha^1 \\
\alpha^4 & \alpha^1 & \alpha^2
\end{bmatrix} = \begin{bmatrix}
1 & 1 & 0 \\
0 & 1 & 1 \\
1 & 0 & 1
\end{bmatrix} \begin{bmatrix}
\gamma^1 & \gamma^2 & \gamma^4 \\
\gamma^2 & \gamma^4 & \gamma^1 \\
\gamma^4 & \gamma^1 & \gamma^2
\end{bmatrix},
\]

we obtain

\[
\begin{pmatrix}
F_0 \\
F_1 \\
F_2 \\
F_3 \\
F_4 \\
F_5 \\
F_6
\end{pmatrix} = \begin{pmatrix}
\alpha^0 & \alpha^1 & \alpha^4 & \alpha^4 & \alpha^2 & \alpha^3 & \alpha^3 & \alpha^5 & \alpha^2 & \alpha^4 & \alpha^6 & \alpha^5 & \alpha^3 & \alpha^6 & \alpha^2 & \alpha^4 & \alpha^6 & \alpha^5 & \alpha^3 & \alpha^6
\end{pmatrix} \begin{pmatrix}
f_0 \\
f_1 \\
f_2 \\
f_3 \\
f_4 \\
f_5 \\
f_6
\end{pmatrix}
\]

\[
= \begin{pmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 0 & 1 & 0 & 0 \\
1 & 0 & 1 & 1 & 0 & 1 & 0 \\
1 & 1 & 0 & 1 & 1 & 0 & 1 \\
1 & 1 & 0 & 1 & 1 & 0 & 1 \\
1 & 0 & 1 & 0 & 0 & 1 & 1 \\
1 & 0 & 0 & 1 & 1 & 0 & 1
\end{pmatrix} \begin{pmatrix}
\gamma^1 & \gamma^2 & \gamma^4 \\
\gamma^2 & \gamma^4 & \gamma^1 \\
\gamma^4 & \gamma^1 & \gamma^2 \\
\gamma^1 & \gamma^2 & \gamma^4 \\
\gamma^2 & \gamma^4 & \gamma^1 \\
\gamma^4 & \gamma^1 & \gamma^2
\end{pmatrix} \begin{pmatrix}
1 \\
f_1 \\
f_2 \\
f_3 \\
f_4 \\
f_5 \\
f_6
\end{pmatrix}
\]

Note that the binary matrix \( A_e \) for this algorithm in (1) consists of binary circulants and block circulants.

## 8 Complexity

The Fourier transform algorithms \([5, 6, 7]\) of length \( n = 2^m - 1 \) over \( GF(2^m) \) take two stages:

1. The first stage is calculation of \( l \) \( m \)-point cyclic convolutions;
2. The second stage is multiplying the binary matrix \( A_e \) by the vector \( D_e f_e \).

The complexity of the first stage is about \( n \log n \) multiplications and additions over elements of \( GF(2^m) \). The complexity of the second stage is \( N_{add} < 2n^2 / \log n \) additions over elements of \( GF(2^m) \) \([8]\).
References

[1] Gerald Goertzel. An Algorithm for the Evaluation of Finite Trigonometric Series. *The American Mathematical Monthly*, vol. 65, no. 1, pp. 34–35, January 1958.

[2] R. E. Blahut, *Theory and Practice of Error Control Codes*. Reading, MA: Addison-Wesley, 1983.

[3] R. E. Blahut, *Fast Algorithms for Digital Signal Processing*. Reading, MA: Addison-Wesley, 1985.

[4] R. E. Blahut. *Algebraic Codes on Lines, Planes, and Curves: An Engineering Approach*. Cambridge, U.K.: Cambridge University Press, 2008.

[5] S. Fedorenko, P. Trifonov. On computing the fast Fourier transform over finite fields. Proceedings of Eighth International Workshop on Algebraic and Combinatorial Coding Theory at Tsarskoe Selo, Russia, pp. 108–111, September 2002.

[6] P. V. Trifonov, S. V. Fedorenko. A method for fast computation of the Fourier transform over a finite field. *Problems of Information Transmission*, vol. 39, no. 3, pp. 231–238, 2003. Translation of *Problemy Peredachi Informatsii*.

[7] S. V. Fedorenko. A method for computation of the discrete Fourier transform over a finite field. *Problems of Information Transmission*, vol. 42, no. 2, pp. 139–151, 2006. Translation of *Problemy Peredachi Informatsii*.

[8] A. Aho, J. Hopcroft, and J. Ullman, *The Design and Analysis of Computer Algorithms*. Reading, MA: Addison-Wesley, 1976.