On Computing the Fast Fourier Transform over Finite Fields

Sergei Fedorenko[1] **and Peter Trifonov**

St.Petersburg State Polytechnical University,
Distributed Computing and Networking Department,
Politekhnitcheskaya st., 21, Room 9–104,
St.Petersburg, 194021, Russia.
sfedorenko@ieee.org petert@dcn.nord.nw.ru

In this paper we consider the problem of computing the Fast Fourier Transform of a polynomial over finite fields. The polynomial is decomposed into a sum of linearized polynomials allowing one to use fast evaluation algorithms. An example of the FFT algorithm with the complexity lower than the best one known to the authors is provided.

1 Introduction

Currently there exist a lot of algorithms for computing the Fast Fourier Transform (FFT) over the field of complex numbers. Many of these algorithms can be used in the case of finite fields, but in practice the problem of construction of FFT for a finite field remains hard and poorly formalized [3].

In this paper we suggest an universal approach for the construction of FFT algorithms over the fields of characteristics 2. The algorithm is based on the decomposition of an arbitrary polynomial into a sum of linearized polynomials allowing thus usage of the effective evaluation algorithms [2].

2 Basic definitions

Definition 1. The polynomial over $GF(2^m)$ is called linearized if

$$L(x) = \sum_i l_i x^{2^i}, \quad l_i \in GF(2^m).$$

It can be easily proved that for linearized polynomials $L(a+b) = L(a) + L(b)$ holds. This property leads to the following theorem presented here in a slightly modified form:

[1]This work was supported by the Alexander von Humboldt Foundation.

Theorem 1 ([1]). *Let $x \in GF(2^m)$ and let $\beta_0, \beta_1, \ldots, \beta_{m-1}$ be a basis of the field.*

$$\text{If} \quad x = \sum_{i=0}^{m-1} x_i \beta_i, \quad x_i \in GF(2), \qquad \text{then} \quad L(x) = \sum_{i=0}^{m-1} x_i L(\beta_i).$$

Let us consider cyclotomic cosets modulo $n = 2^m - 1$ over $GF(2)$:
$\{0\}, \{k_1, k_1 2, k_1 2^2, \ldots, k_1 2^{m_1 - 1}\}, \ldots, \{k_l, k_l 2, k_l 2^2, \ldots, k_l 2^{m_l - 1}\}$, where $k_i \equiv k_i 2^{m_i} \bmod n$.

Then any polynomial $f(x) = \sum_{i=0}^{n-1} f_i x^i$, $f_i \in GF(2^m)$ can be decomposed as

$$f(x) = \sum_{i=0}^{l} L_i(x^{k_i}),\ L_i(y) = \sum_{j=0}^{m_i - 1} f_{k_i 2^j \bmod n} y^{2^j}. \tag{1}$$

In fact (1) represents a way of grouping numbers $0 \le s < n$ into cyclotomic cosets: $s \equiv k_i 2^j \bmod n$. Obviously, this decomposition is always possible. Note, that term f_0 can be represented as $L_0(x^0)$, where $L_0(y) = f_0 y$.

3 Fast Fourier Transform

Let us consider the problem of computing the FFT of a polynomial $f(x)$, i.e. computing values $f(\alpha^j) = \sum_{i=0}^{n-1} f_i \alpha^{ij}$, where α is a primitive element of $GF(2^m)$. According to (1), $f(\alpha^j)$ can be represented as $f(\alpha^j) = \sum_{i=0}^{l} L_i(\alpha^{j\,k_i})$. It is known [1] that α^{k_i} is a root of a minimal polynomial of degree m_i and thus belongs to a subfield $GF(2^{m_i})$, $m_i \mid m$. Thus all the values $(\alpha^{k_i})^j$ lie in $GF(2^{m_i})$ and so they can be decomposed in some basis $(\beta_{i,0}, \ldots, \beta_{i,m_i - 1})$ of the subfield: $\alpha^{j\,k_i} = \sum_{s=0}^{m_i - 1} a_{ijs} \beta_{i,s}, a_{ijs} \in GF(2)$. Then, according to the theorem 1,

$$F_j = f(\alpha^j) = \sum_{i=0}^{l} \sum_{s=0}^{m_i - 1} a_{ijs} L_i(\beta_{i,s}) = \sum_{i=0}^{l} \sum_{s=0}^{m_i - 1} a_{ijs} \left(\sum_{p=0}^{m_i - 1} \beta_{i,s}^{2^p} f_{k_i 2^p} \right). \tag{2}$$

This equation can be represented in matrix form as $F = ALf$, where $F = \|F_j\|$, $f = \|f_j\|$, A is a matrix with elements $a_{ijs} \in GF(2)$, L is a block diagonal matrix with elements $\beta_{i,s}^{2^p}$.

It is possible to choose the same basis for all the linearized polynomials of the same degree m_i in (1) and obtain very small amount of different blocks in matrix L. This can simplify the problem of construction of a fast algorithm for multiplication of a matrix L by a vector f over $GF(2^m)$.

The described transforms are similar to the ones presented in [4]. The main differences are:

1. Matrix L has regular structure which can be used for a further optimization.
2. There is a single multiplication of a binary matrix by a vector. This can be used for a better optimization.

Example 1. A polynomial $f(x) = \sum_{i=0}^{6} f_i x^i, \quad f_i \in GF(2^3)$ can be represented as

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

Let us choose as basis elements of $GF(2^3)$ the standard basis and represent the components of Fourier transform as

$$\begin{aligned} f(\alpha^0) &= L_0(\alpha^0) + L_1(\alpha^0) + L_2(\alpha^0) \\ f(\alpha^1) &= L_0(\alpha^0) + L_1(\alpha) + L_2(\alpha^3) = L_0(1) + L_1(\alpha) + L_2(1) + L_2(\alpha) \\ f(\alpha^2) &= L_0(\alpha^0) + L_1(\alpha^2) + L_2(\alpha^6) = L_0(1) + L_1(\alpha^2) + L_2(1) + L_2(\alpha^2) \\ f(\alpha^3) &= L_0(\alpha^0) + L_1(\alpha^3) + L_2(\alpha^2) = L_0(1) + L_1(1) + L_1(\alpha) + L_2(\alpha^2) \\ f(\alpha^4) &= L_0(\alpha^0) + L_1(\alpha^4) + L_2(\alpha^5) = L_0(1) + L_1(\alpha) + L_1(\alpha^2) + L_2(1) + L_2(\alpha) + L_2(\alpha^2) \\ f(\alpha^5) &= L_0(\alpha^0) + L_1(\alpha^5) + L_2(\alpha) = L_0(1) + L_1(1) + L_1(\alpha) + L_1(\alpha^2) + L_2(\alpha) \\ f(\alpha^6) &= L_0(\alpha^0) + L_1(\alpha^6) + L_2(\alpha^4) = L_0(1) + L_1(1) + L_1(\alpha^2) + L_2(\alpha) + L_2(\alpha^2), \end{aligned}$$

where α is a root of the primitive polynomial x^3+x+1. These equations can be represented in a matrix form as

$$F = \begin{pmatrix} F_0 \\ F_1 \\ F_2 \\ F_3 \\ F_4 \\ F_5 \\ F_6 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} L_1(1) \\ L_1(\alpha) \\ L_1(\alpha^2) \\ L_2(1) \\ L_2(\alpha) \\ L_2(\alpha^2) \\ f_0 \end{pmatrix} = AS.$$

Then the problem of computing the FFT of a polynomial $f(x)$ can be represented as

$$F = A \begin{pmatrix} W & \mathbf{0} & 0 \\ \mathbf{0} & W & 0 \\ 0 & 0 & 1 \end{pmatrix} \left(f_1, f_2, f_4, f_3, f_6, f_5, f_0\right)^T, \quad W = \begin{pmatrix} 1 & 1 & 1 \\ \alpha & \alpha^2 & \alpha^4 \\ \alpha^2 & \alpha^4 & \alpha^8 \end{pmatrix}. \tag{3}$$

The first stage of the algorithm is computing $\begin{pmatrix} b_{i1} \\ b_{i2} \\ b_{i3} \end{pmatrix} = W \begin{pmatrix} a_{i1} \\ a_{i2} \\ a_{i3} \end{pmatrix}$ for $i = 1, 2$, where $a_{11} = f_1$, $a_{12} = f_2$, …, $a_{23} = f_5$ (see (3)). This can be implemented using the following algorithm:

$$\begin{aligned} b_{i1} &= a_{i1} + a_{i2} + a_{i3} \\ b_{i2} &= \alpha(a_{i1} + a_{i2}) + \alpha^4(a_{i2} + a_{i3}) \\ b_{i3} &= \alpha^2(a_{i1} + a_{i3}) + \alpha^4(a_{i2} + a_{i3}), \end{aligned}$$

which requires 3 multiplications and 6 additions. At the end of the first stage one obtains the vector $S = (S_0, \ldots, S_6) = (b_{11}, b_{12}, b_{13}, b_{21}, b_{22}, b_{23}, f_0)$. The following algorithm computes the product of a binary matrix A with vector S:

$$\begin{array}{ll}
T_7 = S_3 + S_6 & F_2 = T_2 = T_7 + T_{10} \\
T_8 = S_1 + S_5 & F_3 = T_3 = S_6 + T_{12} \\
T_9 = S_1 + S_4 & F_4 = T_4 = T_1 + T_{10} \\
T_{10} = S_2 + S_5 & T_{11} = S_2 + T_3 \\
T_{12} = S_0 + T_8 & F_6 = T_6 = T_9 + T_{11} \\
F_0 = T_0 = S_0 + T_7 & F_5 = T_5 = T_6 + T_8. \\
F_1 = T_1 = T_7 + T_9 &
\end{array}$$

Thus the FFT of length 7 can be computed with $2 \times 3 = 6$ multiplications and $2 \times 6 + 13 = 25$ additions. This is smaller by one addition than in the algorithm presented in [4].

If one chooses the normal basis in (2) then all the blocks of the matrix L are circulant matrices. Thus the problem of the multiplication by this matrix can be considered as a problem of the computing a set of circular convolutions of degree $m_i \mid m$. Application of these techniques allowed us to construct the FFT algorithm of length 15 with $3 \times 5 + 1 \times 1 = 16$ multiplications and $3 \times 10 + 1 \times 2 + 45 = 77$ additions which is better than the ones presented in [4] (16 multiplications and 100 additions) and [3] (20 multiplications and 70 additions).

4 Conclusions

In this paper we suggested an algorithm for computing the FFT of a polynomial over $GF(2^m)$. The task of computing the FFT of length $n = 2^m - 1$ can be reduced to computing the circular convolutions of length $m_i \mid m$ and multiplication of a binary matrix by a vector.

References

[1] E.R. Berlekamp. *Algebraic coding theory.* New York: McGraw-Hill, 1968.

[2] S.V. Fedorenko and P.V. Trifonov. Finding roots of polynomials over finite fields. *Accepted for publication in IEEE Transactions on Communications*, 2002.

[3] E.M. Gabidulin and V.B. Afanasyev. *Coding in radioelectronics.* Moscow, Radio i Svyaz, 1986 (in Russian).

[4] T.G. Zakharova. Fourier transform evaluation in fields of characteristic 2. *Problems of Information Transmission*, 28(2): 154–167, 1992.