Regular estimates for the complexity of polynomial multiplication and truncated Fourier transform*

I. S. Sergeev†

Abstract

In the present paper polynomial multiplication circuits, which are efficient either in complexity and depth or in complexity and memory size, are proposed. Consequently, for instance, the multiplication of polynomials of total degree $n = 2^{n_1} + \ldots + 2^{n_s}$, where $n_1 > \ldots > n_s$, over a ring with invertible 2, can be implemented in $M(n_1) + \ldots + M(n_s) + O(n)$ arithmetic ring operations with depth $\max_i\{D(n_i)\} + O(\log n)$, where $M(k)$ and $D(k)$ are the complexity and depth of a modulo $x^{2^k} + 1$ multiplication circuit. As another example, the truncated DFT of order n (i.e. the DFT of order $2^{\lceil \log_2 n \rceil}$, reduced to vectors of dimension n) can be implemented by a circuit of complexity $1.5n \log_2 n + O(n)$ and memory size $n + 1$.

Keywords: *arithmetic circuits, complexity, depth, memory size, multiplication, discrete Fourier transform* (*DFT*).

Introduction

In this paper, we consider the problem of constructing circuits for multiplying polynomials with complexity regularly (uniformly, smoothly) depending on the input size (i.e. the degrees of polynomials). A typical situation is when efficient (basic) circuits for multiplying modulo $x^{2^k} + 1$ are available, but it is required to construct a circuit for multiplying polynomials of arbitrary total degree $n - 1$ with complexity that asymptotically depends on n in the same way as in the case of $n = 2^k$ (the asymptotics is naturally assumed to be a smooth function). In theoretical papers, this goal is usually achieved by modifying the basic multiplication algorithm, for example, by extending it to modules of the form $x^{s2^k} + 1$. But such techniques usually worsen the second term in the complexity estimate. (The notion "second term" is used informally here and below.) Interestingly, a better result can be achieved without interfering with the basic algorithm at all (using it as a "black box"). To do this, it is enough to be able to efficiently reduce a polynomial modulo a set of binomials $x^{2^i} + 1$ and restore it from known remainders from division by these modules. This is what we will discuss below.

*__Translated version.__ Originally published in: Prikladnaya Diskretnaya Matematika [Applied Discrete Math.]. 2011, no. 4(14), P. 72–88 (in Russian).

†Lomonosov Moscow State University, Moscow, Russia. e-mail: *isserg@gmail.com*

[0]The work was supported by RFBR, projects 11–01–00508 and 11–01–00792–a, and the fundamental research program of the Department of Mathematical Sciences of the Russian Academy of Sciences "Algebraic and combinatorial methods of mathematical cybernetics and new generation information systems" (project "Problems of optimal synthesis of control systems").

The above problem is consonant with the problem of efficient implementation of the truncated DFT (TFT — Truncated Fourier transform). The DFT of order of a power of two is computed most efficiently, therefore in [9] it is proposed to use n components of the DFT of order $2^{\lceil \log_2 n \rceil}$ instead of the DFT of arbitrary order n; the corresponding transform in [9] is called the truncated DFT.

In this paper, we consider circuits for solving the listed problems, which are efficient in terms of complexity (potentially more than one term from the asymptotic complexity of the basic circuit can be preserved) and simultaneously of depth or space (storage or memory size). The formulation of the problem of minimizing the space is borrowed from [8].

In addition, the problem of efficient implementation of polynomial multiplication via multiplication modulo $x^{2\cdot 3^k} + x^{3^k} + 1$ circuits is considered. Such circuits were proposed for multiplication by A. Schönhage [13]. Schönhage's method can be generalized to multiplication modulo $x^{(p-1)\cdot p^k} + x^{(p-2)\cdot p^k} + \ldots + x^{p^k} + 1$, where p is a prime [2]. However, for $p \neq 3$ this generalization apparently has no practical significance yet (see also [1, 7]). Such circuits are usually applied when two is non-invertible in the coefficient ring, but three is invertible (a popular example is finite fields of characteristic 2). In this case, as shown below, for multiplying polynomials of total degree $n-1$ one can construct circuits with a regular complexity estimate for $n \in \bigcup_{k\in\mathbb{N}}[2\cdot 3^k,\ 3^{k+1})$, and with a worsening of the asymptotic complexity by at most $4/3$ times for other n, provided that the basic circuits are kept intact.

Let us discuss the essence of the question in more detail, introducing the necessary concepts along the way.

Let $\mathbf{K}$ be a commutative (and associative) ring with unity. An element $\zeta \in \mathbf{K}$ is called a *primitive root* of order N if $\zeta^N = 1$ and for any prime $p|N$ the element $\zeta^{N/p} - 1$ is not a zero divisor in $\mathbf{K}$.

Discrete Fourier transform of order N is the $(\mathbf{K}^N \to \mathbf{K}^N)$-transform

$$\mathrm{DFT}_{N,\zeta}(\gamma_0, \ldots, \gamma_{N-1}) = (\gamma_0^*, \ldots, \gamma_{N-1}^*), \qquad \gamma_j^* = \sum_{i=0}^{N-1} \gamma_i \zeta^{ij},$$

where ζ is a primitive root of order N. If an element $N = 1+\ldots+1 \in \mathbf{K}$ is invertible, then there exists an inverse transform to the DFT (called *inverse DFT*) satisfying $\mathrm{DFT}_{N,\zeta}^{-1} = (1/N)\mathrm{DFT}_{N,\zeta^{-1}}$.

It is convenient to keep in mind the polynomial interpretation of the DFT as an isomorphism of rings, when the DFT input is regarded as the coefficient vector of a polynomial:

$$\mathrm{DFT}_{N,\zeta} : \mathbf{K}[x]/(x^N - 1) \to \mathbf{K}^N, \quad \mathrm{DFT}_{N,\zeta}(\Gamma(x)) = \left(\Gamma(\zeta^0), \ldots, \Gamma(\zeta^{N-1})\right).$$

For more details on the DFT, see, e.g., [7].

To analyze the efficiency of the algorithms discussed below, it is convenient to employ the standard computational model of *circuits of functional elements* [11, 16] (hereinafter, simply *circuits*). Namely, we consider circuits over the arithmetic basis $\{x \pm y,\ xy\} \cup \{ax | a \in \mathbf{K}\}$ (*arithmetic circuits*). For programmers, the concept of *straight-line program* is closer: essentially, a straight-line program (hereinafter, simply *program*) is a circuit for which the sequence of operations (numeration of circuit elements) is fixed. So, several programs can correspond to one circuit.

Several measures of circuit (program) complexity are defined in a standard way. Namely, *complexity* is the number of functional elements in a circuit (program). Circuit *depth* is the maximum number of elements in a directed input-output chain in the circuit. *Memory space* of a program is the maximum size of intermediate data (including all already computed outputs) used in subsequent iterations during the program execution. Somewhat artificially, the memory space of a circuit can be defined as the minimum memory space over all programs corresponding to a given circuit.

Essentially, complexity corresponds to the execution time of a program on a single-processor machine or the area of a microchip implementing a circuit; depth corresponds to the response time of a microcircuit or the number of parallel steps performed by a multiprocessor machine; memory space in the definition corresponds to the memory capacity allocated for storing variables when implementing a program on a computer.

When analyzing the memory space, it is convenient to represent the computation process as follows. Intermediate data are stored in memory cells (the capacity of a cell is equal to one element of the ring $\mathbf{K}$). At the beginning of work, the cells contain input data. Elementary (basic) operations are performed sequentially. Each operation uses the data that is in memory at the time of its execution. The result of an operation is stored in some cell.

A DFT of order of a power of two allows the most efficient implementation. Hence, fast algorithms efficiently multiply polynomials of total degree $n-1$ if $n=2^k$, assuming that the coefficient ring or its suitable extension admits a DFT of order of a power of two.

As is known, by the Cooley—Tukey method [3] a DFT of order 2^k can be implemented by a circuit of $k2^k$ addition-subtraction elements, $(k-2)2^{k-1}+1$ elements of multiplication by powers of a primitive root (of order 2^k), and of depth $2k-1$ (if only additive operations are taken into account, then the depth is k). This circuit can be rebuilt so as to have memory space 2^k+1 with the same complexity. By adding $k2^{k-1}$ elements of multiplication by 2, the space can be reduced to 2^k; what is meant here is that the basic transform of the circuit, the order-2 DFT: $(x,y)\to(x+y,x-y)$, can be computed with complexity 2 and space 3 in the chain $(x,y)\to(x+y,x,y)\to(x+y,x-y)$ or with complexity 3 and space 2 in the chain $(x,y)\to(x+y,y)\to(x+y,2y)\to(x+y,x-y)$.

The inverse DFT can be implemented in the same way as the forward one, by replacing ζ with ζ^{-1} and performing 2^k multiplications by the constant $2^{-k}\in\mathbf{K}$ at the end. However, some of these multiplications can be combined with internal multiplications by powers of the primitive root.

If $n\neq 2^k$, then "by default" the algorithm for multiplying polynomials of total degree $2^{\lceil\log_2 n\rceil}-1$ can be applied. However, the complexity of the multiplication algorithm grows irregularly with increasing n, and approximately twofold jumps occur when passing through powers of two. There are known techniques that allow smoothing the complexity function so that for any n it is asymptotically expressed in terms of n in the same way as in the case of $n=2^k$. For example, DFTs of order of powers of two can be replaced by DFTs of order $s2^l$, where $s\ll 2^l$ [14], or slightly more general multiple DFTs [15]. However, these techniques, as a rule, worsen the second-order term of the complexity function and transfer the jumps to it.

Another approach was proposed by van der Hoeven in [9, 10], where a truncated DFT is implemented that computes the DFT values on some subset of points from the set corresponding to the DFT of order of the nearest power of two from above. The *truncated DFT* (*TFT*) *of order* n is defined as a set of n components of the vector

$\mathrm{DFT}_{2^{\Lambda(n)},\zeta}(\gamma_0,\ldots,\gamma_{n-1},0,\ldots,0)$, where $\Lambda(n)=\lceil\log_2 n\rceil$.

Van der Hoeven [9] proposed to include into the TFF such components, which are values at the points $\zeta^{\rho(i)}$, $i=0,\ldots,n-1$, where $\rho(i)$ is a number whose binary notation is the reversed length-$\Lambda(n)$ binary notation of the number i.

The forward TFT circuit in the method [9] is obtained from the order-$2^{\Lambda(n)}$ DFT circuit by removing "unnecessary" operations. The inverse TFT circuit is somewhat more complicated. For both circuits, the complexity is estimated as $n\Lambda(n)+2^{\Lambda(n)}$ additive operations and $\lceil(n\Lambda(n)+2^{\Lambda(n)})/2\rceil$ multiplications by powers of a primitive root (multiplications by powers of two are ignored in the calculation, or more precisely, operations of the form $x\pm 2^s y$ are allowed). The depth of the forward TFT circuit is $\Lambda(n)$; the depth of the inverse TFT circuit is somewhat larger, but also $O(\log n)$. The memory space of this circuit (more precisely, the circuit rebuilt similarly to the above-mentioned circuit of the usual DFT), as noted in [8], is $2^{\Lambda(n)}$. In [8], the circuit of space $n+O(1)$ is proposed, but with a greater asymptotic complexity, although of the same order, $O(n\log n)$.

From the perspective of the polynomial multiplication problem, ideas close to employing the TFT were previously expressed in [4], where it was proposed to perform multiplication modulo two numbers that allow a fast multiplication algorithm and finally restore the product based on the Chinese remainder theorem, and later in [1], where it was proposed to use several modules. In terms of DFTs, this means choosing a TFT of order l, where the number l has a small (binary) weight.

In [12], it was proposed to choose the roots of polynomials $x^{2^i}+1$ of total degree n as the points of a TFT of order n.

For further analysis, it is convenient to introduce the concept of *odd DFT* (*ODFT*) *of order* N:

$$\mathrm{ODFT}_{N,\zeta}:\mathbf{K}[x]/(x^N+1)\to\mathbf{K}^N,\quad \mathrm{ODFT}_{N,\zeta}(\Gamma(x))=\left(\Gamma(\zeta^1),\Gamma(\zeta^3),\ldots,\Gamma(\zeta^{2N-1})\right),$$

where ζ is a primitive root of degree $2N$. In other words, the components of the ODFT of order N are the components of the DFT of order $2N$, distinct from the components of the DFT of order N (if we mean the polynomial interpretation of the DFT). A simple way to implement $\mathrm{ODFT}_{N,\zeta}$ is to compose a variable substitution $x\to\zeta x$ and DFT_{N,ζ^2}. Accordingly, the inverse ODFT can be implemented as a composition of $\mathrm{DFT}^{-1}_{N,\zeta^2}$ and a substitution $x\to x/\zeta$.

Let $n=2^{n_1}+\ldots+2^{n_s}$, where $n_1>\ldots>n_s$. An order-n TFT circuit [12] may be constructed from circuits of ODFTs of orders $2^{n_1},\ldots,2^{n_s}$ plus $2^{\Lambda(n)}+n$ additive operations and n multiplications. The inverse TFT circuit, in addition to circuits implementing inverse ODFTs of orders $2^{n_1},\ldots,2^{n_s}$, involves $3\cdot 2^{\Lambda(n)}+n$ additive operations and n multiplications. These complexity estimates were calculated in [12] for a slightly more general problem and can be reduced. The depth and memory space of the circuits [12] are estimated approximately the same as for [9].

Further, somewhat more accurate upper bounds on the complexity of this TFT are obtained together with the depth estimates, and it is shown that a TFT of order n can be implemented by a circuit with space n (or $n+1$) and additional complexity $O(n)$.

The issue of memory optimization can naturally be considered in the problem of polynomial multiplication. In [8], based on an efficient TFT circuit, a circuit for multiplying polynomials of degree $n/2-1$ with complexity $O(n\log n)$ and space $2n+O(1)$ is constructed, where n memory "cells" are allocated for storing the input coefficients and are not modified.

From the estimates obtained further it follows, in particular, that for multiplying polynomials of total degree $n-1$ it is possible to construct a circuit of complexity $O(n\log n)$ and memory space $2n$. The prohibition on rewriting inputs is not imposed. If we keep in mind that in fact a circuit is constructed for multiplying modulo some degree-n polynomial, then the memory space of such a circuit cannot be further reduced.

Results for the problems of computing TFT and multiplying polynomials are formulated in §1.1 and proved in §1.2. In Section 2, polynomial multiplication circuits are composed from modulo $x^{2\cdot 3^k}+x^{3^k}+1$ multiplication circuits.

1 TFT and the binary multiplication method

1.1 Main results

Let us have at our disposal circuits for order-2^k ODFTs of complexity $\Phi(k)=\Phi_A(k)+\Phi_2(k)+\Phi_C(k)$, depth $d_\Phi(k)$ and memory space $v_\Phi(k)$. Here $\Phi_A(k)$ is the number of additive elements, $\Phi_2(k)$ is the number of multiplications by powers of two, $\Phi_C(k)$ is the number of other scalar multiplications. We introduce similar notations with strokes for the parameters of circuits implementing inverse ODFTs.

We fix the notation $n=2^{n_1}+\ldots+2^{n_s}$, where $n_1>\ldots>n_s$.

Theorem 1. *The truncated DFT of order n can be implemented by a circuit*

a) of $2n+\sum_i\Phi_A(n_i)$ additive operations, $\sum_i\Phi_2(n_i)$ and $\sum_i\Phi_C(n_i)$ multiplications by powers of two and other constants, respectively, and depth $n_1+\max_i\{d_\Phi(n_i)-n_i\}+1$;

b) of $4n-2^{n_1+1}+\sum_i\Phi_A(n_i)$ additive operations, $4n-3\cdot 2^{n_1}+\sum_i\Phi_2(n_i)$ and $\sum_i\Phi_C(n_i)$ multiplications by powers of two and other constants, respectively, and space $n+\max_i\{v_\Phi(n_i)-2^{n_i}\}$.

The inverse truncated DFT of order n can be implemented by a circuit

c) of $4n-3\cdot 2^{n_1}+\sum_i\Phi'_A(n_i)$ additive operations, $2n-2^{n_1+1}+\sum_i\Phi'_2(n_i)$ and $\sum_i\Phi'_C(n_i)$ multiplications by powers of two and other constants, respectively, and depth $n_1-n_s+2s-1+\max_i\{d'_\Phi(n_i)\}$;

d) of $4n-2^{n_1+1}+\sum_i\Phi'_A(n_i)$ additive operations, $2n-2^{n_1+1}+\sum_i\Phi'_2(n_i)$ and $\sum_i\Phi'_C(n_i)$ multiplications by powers of two and other constants, respectively, and space $n+\max_i\{v'_\Phi(n_i)-2^{n_i}\}$.

The theorem follows directly from Lemmas 2 and 3 proved below.

Specific estimates can be obtained by substituting the parameters of the ODFT circuits mentioned in the introduction. For example, $\Phi_A(k)=\Phi'_A(k)=k2^k$, $\Phi_C(k)=\Phi'_C(k)=k2^{k-1}$, $\Phi_2(k)=0$, $\Phi'_2(k)=2^k$ and either $d_\Phi(k)=d'_\Phi(k)=2k$, or $v_\Phi(k)=v'_\Phi(k)=2^k+1$.

Now let there be circuits for multiplying polynomials modulo $x^{2^k}+1$ with coefficients over a ring in which the element 2 is invertible. The complexity, depth, and memory space of such circuits will be denoted by $M(k)=M_A(k)+M_2(k)+M_C(k)+M_N(k)$, $d_M(k)$, and $v_M(k)$, where $M_N(k)$ denotes the number of nonscalar multiplications in the circuit (the remaining notations are similar to those introduced above for ODFT circuits). Lemmas 2 and 3 also imply

Theorem 2. *For multiplication of polynomials of total degree $n-1$, one can construct a circuit*

a) of $6n - 3 \cdot 2^{n_1} + \sum_i M_A(n_i)$ *additive operations,* $2n - 2^{n_1+1} + \sum_i M_2(n_i)$, $\sum_i M_C(n_i)$ *and* $\sum_i M_N(n_i)$ *multiplications by powers of two, other constants and nonscalar multiplications, respectively, and depth* $\max_i\{d_M(n_i) - n_i\} + 2n_1 - n_s + 2s$*;*

b) of $12n - 6 \cdot 2^{n_1} + \sum_i M_A(n_i)$ *additive operations,* $10n - 8 \cdot 2^{n_1} + \sum_i M_2(n_i)$, $\sum_i M_C(n_i)$ *and* $\sum_i M_N(n_i)$ *multiplications by powers of two, other constants and nonscalar multiplications respectively, and space* $2n + \max_i\{v_M(n_i) - 2^{n_i+1}\}$.

Given that $s \le n_1 + 1$, we obtain

Corollary 1. *Let* $M(k) = f(2^k)$*, and* $f(x + y) \ge f(x) + f(y)$ *for any* $x, y \ge 1$*. Let also* $d_M(k) - k \le d_M(l) - l$ *and* $v_M(k) - 2^{k+1} \le v_M(l) - 2^{l+1}$ *for any* $k \le l$*. Then, for multiplying polynomials of total degree* $n - 1$*, there is a circuit*

a) of complexity $f(n) + 8n - 5 \cdot 2^{n_1}$ *and depth* $d_M(n_1) + 3n_1 + 2$*;*

b) of complexity $f(n) + 22n - 14 \cdot 2^{n_1}$ *and memory space* $2n + v_M(n_1) - 2^{n_1+1}$.

By choosing the number n' that is the closest from above to n and is a multiple of $2^{n_1-\alpha(n)}$, where $\alpha(n)$ is a slowly growing natural function, and by passing to a circuit multiplying polynomials of total degree at most $n' - 1$, we obtain another corollary of Theorem 2, item *a* (by the inequalities $n_1 - n_s \le \alpha(n)$, $s \le \alpha(n) + 1$):

Corollary 2. *Under the conditions of Corollary* 1*, additionally assume that* $f(x)/x \to \infty$ *for* $x \to \infty$ *and* $f(x) = x^{O(1)}$*. Then, for multiplying polynomials of total degree* $n - 1$*, there is a circuit of complexity no greater than* $(1 + o(1))f(n)$ *and depth no greater than* $d_M(\lfloor \log_2(n + o(n)) \rfloor) + o(\log n)$.

Note that the assumptions in the formulations of the corollaries are natural. First, we assume that the modulo $x^{2^k} + 1$ multiplication circuits are structured in a uniform manner. Second, we assume that the partially defined complexity function $M(k)$ of the multiplication circuits, $k \in \{2^i\}$, can be extended to a function $f(x)$ satisfying the superlinearity condition $f(x + y) \ge f(x) + f(y)$ (Corollary 1) and having nonlinear growth (Corollary 2). The first of the conditions on $f(x)$ actually follows from the second, and the second is based on the widespread assumption of nonlinearity of the multiplication complexity function. If one of the conditions cannot be satisfied, then instead of $f(x)$ we can take a suitable function $f_>(x) \ge f(x)$ satisfying both conditions.

Specific estimates can be obtained by substituting the known parameters of the circuits for multiplication modulo $x^{2^k} + 1$ — they are provided, e.g., in [7, 1, 12, 6].

In a standard way, a circuit for multiplication modulo $x^{2^k} + 1$ can be constructed from two order-2^k ODFT circuits, an inverse ODFT circuit (if a DFT of order 2^{k+1} exists in the ring under consideration) and 2^k nonscalar multiplications performed at one level. Substituting the known parameters of these circuits in Theorem 2, item *b*, we obtain the complexity bound $O(n \log n)$ while memory space is $2n$.

1.2 Auxiliary statements

Let $a(x) = \sum_{l=0}^{n-1} a_l x^l$. Formally, set $a_l = 0$ for $l \ge n$. Introduce the notations $a_{k,l}$ and $b_{k,l}$:

$$a(x) \bmod (x^{2^k} - 1) = \sum_{l=0}^{2^k-1} a_{k,l} x^l, \quad a(x) \bmod (x^{2^k} + 1) = \sum_{l=0}^{2^k-1} b_{k,l} x^l.$$

Obviously, the coefficients $a_{k,l}$ and $b_{k,l}$ are, respectively, constant-sign and alternating-sign sums of the coefficients a_i with index step 2^k:

$$a_{k,l} = a_{k+1,l} + a_{k+1,2^k+l} = \sum_{j2^k<n-l} a_{j2^k+l}, \quad b_{k,l} = a_{k+1,l} - a_{k+1,2^k+l} = \sum_{j2^k<n-l} (-1)^j a_{j2^k+l}.$$

The following lemma is used primarily in constructing memory-efficient circuits.

Lemma 1. *Let $n_{i+1} < k \le n_i$. The following formulas are valid:*

$$a_{k,l} = \sum_{j_i=0}^{2^{n_i-k}-1} \left(b_{n_i,\, j_i 2^k+l} + 2 \sum_{j_{i-1}=0}^{2^{n_{i-1}-n_i-1}-1} \Big(b_{n_{i-1},\, (2j_{i-1}+1)2^{n_i}+j_i 2^k+l} + \right.$$
$$+ 2 \sum_{j_{i-2}=0}^{2^{n_{i-2}-n_{i-1}-1}-1} \Big(b_{n_{i-2},\, (2j_{i-2}+1)2^{n_{i-1}}+(2j_{i-1}+1)2^{n_i}+j_i 2^k+l} + \dots$$
$$\left. \left. \dots + 2 \sum_{j_1=0}^{2^{n_1-n_2-1}-1} b_{n_1,\, (2j_1+1)2^{n_2}+\dots+(2j_{i-1}+1)2^{n_i}+j_i 2^k+l} \dots \Big) \Big) \right) + 2^i a_{2^{n_1}+\dots+2^{n_i}+l};$$

$$b_{n_i,l} = \sum_{j=0}^{1} (-1)^j \left(\sum_{j_{i-1}=0}^{2^{n_{i-1}-n_i-1}-1} \Big(b_{n_{i-1},\, (2j_{i-1}+j)2^{n_i}+l} + \right.$$
$$+ 2 \sum_{j_{i-2}=0}^{2^{n_{i-2}-n_{i-1}-1}-1} \Big(b_{n_{i-2},\, (2j_{i-2}+1)2^{n_{i-1}}+(2j_{i-1}+j)2^{n_i}+l} + \dots$$
$$\left. \left. \dots + 2 \sum_{j_1=0}^{2^{n_1-n_2-1}-1} b_{n_1,\, (2j_1+1)2^{n_2}+\dots+(2j_{i-2}+1)2^{n_{i-1}}+(2j_{i-1}+j)2^{n_i}+l} \dots \Big) \Big) \right) +$$
$$+ 2^{i-1} \left(a_{2^{n_1}+\dots+2^{n_{i-1}}+l} - a_{2^{n_1}+\dots+2^{n_i}+l} \right).$$

Proof. The first formula is obtained by recursively applying simple relations

$$a_{k,l} = \begin{cases} a_{k+1,l} + a_{k+1,2^k+l}, & k \notin \{n_i : i = 1, \dots, s\} \\ b_{k,l} + 2a_{k+1,2^k+l}, & k \in \{n_i : i = 1, \dots, s\} \end{cases},$$

starting from $a_{n_1+1,l} = a_l$ and taking into account that $a_l = 0$ for $l \ge n$. The second formula is obtained from the first one as $b_{n_i,l} = a_{n_i+1,l} - a_{n_i+1,2^{n_i}+l}$. □

Lemma 2. *Let $m \le n$. Then the reduction of a polynomial of degree $m-1$ modulo $x^{2^{n_i}}+1$, $i = 1, \dots, s$, can be performed by a circuit*

a) of $2(m-1)$ additive elements and depth $n_1 - n_s + 1$, where the coefficients of the remainder of division by $x^{2^{n_i}}+1$ are computed at depth $n_1 - n_i + 1$;

b) of $2(2n - 2^{n_1})$ additive elements, $4n - 3 \cdot 2^{n_1}$ multiplications by powers of two and space n.

Proof. Item *a* is proved by a simple construction [12].

It is easy to see that all sums $a_{k,l}$, where $k = n_s + 1, \ldots, n_1$, can be computed by a single circuit of complexity no greater than $m-1$, in which $a_{k,l}$ are computed at depth at most l for $k = n_1 + 1 - l$.

The desired coefficients $b_{n_i,l}$ are obtained by adding to the constructed circuit

$$\begin{cases} 0, & m \le 2^{n_i}, \\ m - 2^{n_i}, & 2^{n_i} \le m \le 2^{n_i+1}, \\ 2^{n_i}, & m \ge 2^{n_i+1}. \end{cases}$$

subtraction elements located at depth $n_1 - n_i + 1$.

Since $2^{n_i} > 2^{n_{i+1}} + \ldots + 2^{n_s}$, for the number of subtractions (for a suitable i) we obtain the estimate

$$m - 2^{n_i} + 2^{n_{i+1}} + \ldots + 2^{n_s} \le m - 1,$$

from which follows the complexity estimate $2(m-1)$ for the entire circuit.

It is sufficient to prove item b for the case $m = n$. Let us reconstruct the above circuit so that it explicitly computes only those sums $a_{k,l}$ for which "there is enough space".

Let $L_i = 2^{n_i} + \ldots + 2^{n_s}$. For any $k = n_1, \ldots, n_s + 1$, we explicitly compute the sums $a_{k,0}, \ldots, a_{k,L_i-1}$, where $n_{i-1} \ge k > n_i$, taking into account that $n - L_i$ of other "memory cells" are allocated for storing the coefficients $b_{n_j,l}$, where $j < i$.

It is convenient to divide the entire computation into stages, numbering them from n_1 to n_s in descending order. At stage k, the sums $a_{k,0}, \ldots, a_{k,L_i-1}$ are computed (where $n_{i-1} \ge k > n_i$) and, if $k \in \{n_i : i = 1, \ldots, s\}$, then the coefficients $b_{k,0}, \ldots, b_{k,2^k-1}$ are also computed.

If the sum $a_{k,l}$ is not computed explicitly, then the right-hand side of the formula (denote it $\varphi_{k,l}$) from Lemma 1 is used instead, expressing this sum through the already computed coefficients $b_{n_j,t}$, where $n_j \ge k$. Note that since $l \ge L_{i+1}$ for $n_i \ge k > n_{i+1}$, the last term $a_{2^{n_1}+\ldots+2^{n_i}+l}$ in $\varphi_{k,l}$ is zero.

Let ρ_k denote the number of terms in the formula $\varphi_{k,l}$ (this number does not depend on l, as follows from the form of the formula). Then the addition (subtraction) of $\varphi_{k,l}$ is performed in ρ_k additive operations and i multiplications by powers of two without additional space, where $n_i \ge k > n_{i+1}$.

We will illustrate the corresponding method with an example. Let us compute the transform $a \to a + (b + 2c + 4d)$. We will perform the calculation in the following order:

$$a,\ a+b,\ 2^{-1}(a+b),\ c + 2^{-1}(a+b),\ 2^{-1}(c + 2^{-1}(a+b)),$$
$$d + 2^{-1}(c + 2^{-1}(a+b)),\ 2^2(d + 2^{-1}(c + 2^{-1}(a+b))) = a + b + 2c + 4d.$$

Let us estimate the circuit complexity. Consider a stage k, where $n_{i-1} > k > n_i$. The available coefficients are $a_{k+1,0}, \ldots, a_{k+1,L_i-1}$, the remaining $a_{k+1,l}$ are expressed by the formulas $\varphi_{k+1,l}$. Note that $L_i < 2^{n_i+1} \le 2^k$. Then each of the coefficients $a_{k,l}$, where $l < L_i$, is computed as $a_{k+1,l} + \varphi_{k+1,2^k+l}$ with the complexity of ρ_{k+1} additive operations and $i-1$ multiplications by powers of two. The complexity of the stage is estimated as $L_i\rho_{k+1}$ additive operations and $(i-1)L_i$ multiplications by powers of two.

Now consider the stage $k = n_i$. The available coefficients are $a_{k+1,0}, \ldots, a_{k+1,L_i-1}$. In this case, $L_i = 2^k + L_{i+1}$. First, we compute all $b_{k,l}$ by the formulas $a_{k+1,l} - a_{k+1,2^k+l}$ for $l < L_{i+1}$ and $a_{k+1,l} - \varphi_{k+1,2^k+l}$ for the remaining l. In this case, the coefficients $b_{k,l}$ "overwrite" $a_{k+1,l}$, $l \ge L_{i+1}$. Then we compute $a_{k,l}$ for $l < L_{i+1}$ as $2a_{k+1,l} - b_{k,l}$. The

complexity of the stage is estimated as $2L_{i+1} + (2^k - L_{i+1})\rho_{k+1}$ additive operations and $(i-1)2^k - (i-2)L_{i+1} = (i-1)L_i - (2i-3)L_{i+1}$ multiplications by powers of two.

Let us find ρ_k. From the formula of Lemma 1 it follows directly that

$$\rho_{n_i} = 2^{n_1-n_i-(i-1)} + 2^{n_2-n_i-(i-2)} + \ldots + 2^{n_{i-1}-n_i-1} + 1, \tag{1}$$

and $\rho_k = 2^{n_i-k}\rho_{n_i}$ for $n_{i+1} < k < n_i$.

Let us estimate the total additive complexity of the computations. The sum of the complexity estimates for stages $n_i - 1, \ldots, n_{i+1}$ does not exceed

$$C_i = L_{i+1}\rho_{n_i}(1 + 2 + \ldots + 2^{n_i-n_{i+1}-2}) + 2L_{i+2} + (2^{n_{i+1}} - L_{i+2})2^{n_i-n_{i+1}-1}\rho_{n_i} \le$$
$$\le \rho_{n_i}2^{n_i-n_{i+1}-1}(L_{i+1} - L_{i+2} + 2^{n_{i+1}}) - \rho_{n_i}L_{i+1} + 2L_{i+2} = \rho_{n_i}(2^{n_i} - L_{i+1}) + 2L_{i+2}.$$

For $i \ge 1$, the last expression is at most $2^{n_i}\rho_{n_i}$, given that $L_{i+1} > 2L_{i+2}$ and $\rho_{n_i} \ge 1$.

For the complexity of all stages except stage n_1, using (1), we derive the bound

$$C_1 + \ldots + C_{s-1} = \sum_{i=1}^{s-1} 2^{n_i}\rho_{n_i} = \sum_{i=1}^{s-1}(2^{n_1-(i-1)} + 2^{n_2-(i-2)} + \ldots + 2^{n_i}) <$$
$$< (2^{n_1} + 2^{n_1-1} + \ldots) + (2^{n_2} + 2^{n_2-1} + \ldots) + \ldots + (2^{n_s} + 2^{n_s-1} + \ldots) < 2n.$$

Finally, estimating the complexity of stage n_1 as $2L_2 = 2(n - 2^{n_1})$, we obtain the assertion of item b in terms of additive complexity.

The number of multiplications by powers of two at stages $n_i - 1, \ldots, n_{i+1}$ we estimate roughly as

$$D_i = (n_i - n_{i+1} - 1)iL_{i+1} + iL_{i+1} - (2i-1)L_{i+2} \le$$
$$\le i(n_i - n_{i+1})L_{i+1} \le i2^{n_i-n_{i+1}-1}2^{n_{i+1}+1} = i2^{n_i}.$$

Then for the total number of multiplications at all stages, except stage n_1, we have the estimate

$$D_1 + \ldots + D_{s-1} = 2^{n_1} + 2 \cdot 2^{n_2} + 3 \cdot 2^{n_3} + \ldots + (s-1) \cdot 2^{n_{s-1}} =$$
$$= n + (n - 2^{n_1}) + (n - 2^{n_1} - 2^{n_2}) + \ldots <$$
$$< n + (n - 2^{n_1}) + (1/2)\,(n - 2^{n_1}) + (1/2)^2\,(n - 2^{n_1}) + \ldots = n + 2(n - 2^{n_1}).$$

By adding $n - 2^{n_1}$ multiplications by 2 at step n_1, we arrive to the final estimate. □

Lemma 3. *Reconstruction of a polynomial of degree $n-1$ from given remainders modulo $x^{2^{n_i}} + 1$, $i = 1, \ldots, s$, can be performed by a circuit*

a) of $4n - 3 \cdot 2^{n_1}$ additive elements, $2(n - 2^{n_1})$ multiplications by powers of two and depth at most $n_1 - n_s + 2s - 1$;

b) of $2(2n - 2^{n_1})$ additive elements, $2(n - 2^{n_1})$ divisions by 2 and space n.

Proof. We apply Lemma 1 to express the differences

$$h_{i,l} = a_{2^{n_1}+\ldots+2^{n_i-1}+l} - a_{2^{n_1}+\ldots+2^{n_i}+l}, \tag{2}$$

where $i = 1, \ldots, s-1$ and $l = 0, \ldots, L_i - 1$, in terms of the coefficients $b_{n_j,t}$. Note that the subtrahend coefficient is zero for $l \ge L_{i+1}$.

To construct a circuit from item a, compute the auxiliary quantities $c_{i,l}$ and $d_{i,l}$, defined by the equalities:

$$d_{1,l} = b_{n_1,l}, \quad c_{i,l} = \sum_{j=0}^{2^{n_{i-1}-n_i-1}-1} d_{i-1,\,j2^{n_i+1}+l}, \quad d_{i,l} = b_{n_i,l} + 2c_{i,2^{n_i}+l},$$

where $i \geq 2$. Then compute the required coefficients $h_{i,l}$ via the formulas $h_{i,l} = 2^{1-i}(b_{n_i,l} - c_{i,l} + c_{i,2^{n_i}+l})$, and in the case $i = 1$ simply $h_{1,l} = b_{n_1,l}$.

The computational complexity implied in the formulas for $c_{i,l}$, $i = 2, \ldots, s$ and $l = 0, \ldots, 2^{n_i+1} - 1$, is estimated as

$$\sum_{i=2}^{s} 2^{n_i+1}(2^{n_{i-1}-n_i-1} - 1) = \sum_{i=2}^{s}(2^{n_{i-1}} - 2^{n_i+1}) < n - 2(n - 2^{n_1}).$$

The complexity implied in the formulas for $d_{i,l}$, $i = 2, \ldots, s-1$ and $l = 0, \ldots, 2^{n_i} - 1$, is estimated as $2^{n_2} + \ldots + 2^{n_{s-1}} < n - 2^{n_1}$ additive operations and the same number of multiplications by 2. To complete the computation of $h_{i,l}$, we need to perform another $2(2^{n_2} + \ldots + 2^{n_s}) = 2(n - 2^{n_1})$ additive operations and $n - 2^{n_1}$ multiplications by powers of two.

It is easy to verify that $c_{i,l}$ is computed at depth $n_1 - n_i + i - 3$ and, consequently, $h_{i,l}$ — at depth $n_1 - n_i + i$.

From the computed $h_{i,l}$, the coefficients of the polynomial $a(x)$ are easily reconstructed. The coefficients $a_{2^{n_1}+\ldots+2^{n_{i-1}}+l}$ simply coincide with $h_{i,l}$ for $l \geq L_{i+1}$. In particular, in the case $i = s$, all coefficients are known. This allows us to sequentially determine the missing coefficients $a_{2^{n_1}+\ldots+2^{n_{i-1}}+l}$, $l = 0, \ldots, L_{i+1} - 1$, in descending order of i, directly by formulas (2) with the total complexity

$$L_s + L_{s-1} + \ldots + L_2 = 2^{n_2} + 2 \cdot 2^{n_3} + 3 \cdot 2^{n_4} + \ldots < 2(n - 2^{n_1}).$$

The depth of these calculations obviously does not exceed $s - 1$.

Adding up all the estimates, we obtain the assertion of item a.

To prove item b, we construct a circuit that sequentially in descending order of i transforms each coefficient $b_{n_i,l}$ according to the formula of Lemma 1 into the corresponding difference $h_{i,l}$.

It is easy to see that the transformation of each coefficient $b_{n_i,l}$ is performed in σ_i additive operations and $i - 1$ divisions by 2, where σ_i is the number of coefficients $b_{n_j,t}$ on the right-hand side of the second formula of Lemma 1. It can be directly verified that

$$\sigma_i = 2^{n_1-n_i-(i-2)} + 2^{n_2-n_i-(i-3)} + \ldots + 2^{n_{i-1}-n_i}.$$

The additive complexity of computing the differences $h_{i,l}$ can now be estimated as

$$\sum_{i=2}^{s} 2^{n_i}\sigma_i = \sum_{i=2}^{s}(2^{n_1-(i-2)} + 2^{n_2-(i-3)} + \ldots + 2^{n_{i-1}}) <$$
$$(2^{n_1} + 2^{n_1-1} + \ldots) + (2^{n_2} + 2^{n_2-1} + \ldots) + \ldots + (2^{n_s} + 2^{n_s-1} + \ldots) < 2n.$$

The number of divisions by 2 is estimated as

$$\sum_{i=2}^{s}(i-1)2^{n_i} = 2^{n_2} + 2 \cdot 2^{n_3} + 3 \cdot 2^{n_4} + \ldots < 2(n - 2^{n_1}).$$

The final part of the circuit is the same as in item a. □

2 Ternary multiplication method

2.1 Main results

Further, unless otherwise explicitly stated, we assume $n = 2\,(3^{n_1} + 3^{n_2} + \ldots + 3^{n_s})$, where $n_1 > n_2 > \ldots > n_s$. Note that $n \in [2 \cdot 3^{n_1},\, 3^{n_1+1})$.

Let us consider a method of using circuits for multiplying polynomials modulo $x^{2\cdot 3^k} + x^{3^k} + 1$ with coefficients over a ring in which the element 3 is invertible. The complexity, depth, and memory space of such circuits will be denoted by $M(k) = M_A(k) + M_3(k) + M_C(k) + M_N(k)$, $d_M(k)$, and $v_M(k)$, where $M_A(k)$ denotes the number of additive operations; $M_3(k)$ — the number of multiplications by powers of three; $M_C(k)$ — the number of other scalar multiplications; $M_N(k)$ — the number of nonscalar multiplications.

From Lemmas 5 and 6 given below, it follows

Theorem 3. *For multiplication of polynomials of total degree at most $n-1$, one can construct a circuit*

a) of $5.5n - 5\cdot 3^{n_1} + \sum_i M_A(n_i)$ additive operations, $1.5n - 3^{n_1+1} + \sum_i M_3(n_i)$, $\sum_i M_C(n_i)$ and $\sum_i M_N(n_i)$ multiplications by powers of three, other constants and nonscalar multiplications, respectively, and depth $4n_1 + \max_i\{d_M(n_i) - 2n_i\} - 2n_s + s + 1$. In the case of a ring of characteristic 2, the additive complexity estimate may be reduced by $0.5n$;

b) of $15.5n - 19 \cdot 3^{n_1} + \sum_i M_A(n_i)$ additive operations, $4.75n - 8.5 \cdot 3^{n_1} + \sum_i M_3(n_i)$, $\sum_i M_C(n_i)$ and $\sum_i M_N(n_i)$ multiplications by powers of three, other constants and nonscalar multiplications, respectively, and space $2n + \max_i\{v_M(n_i) - 4 \cdot 3^{n_i}\}$.

Corollary 3. *Let $M(k) = f(2\cdot 3^k)$, where for any $x, y \geq 1$ we have $f(x+y) \geq f(x)+f(y)$. Let also $d_M(k) - 2k \leq d_M(l) - 2l$ and $v_M(k) - 4 \cdot 3^k \leq v_M(l) - 4 \cdot 3^l$ for any $k \leq l$.*

Then for multiplying polynomials of total degree at most $n-1$ there is a circuit

a) of complexity $f(n) + 7n - 8 \cdot 3^{n_1}$ and depth $d_M(n_1) + 3n_1 + 2$, and in the case of a ring of characteristic 2 the circuit complexity is bounded by $f(n) + 5(n - 3^{n_1})$;

b) of complexity $f(n) + 20.25n - 27.5 \cdot 3^{n_1}$ and memory space $2n + v_M(n_1 - 4 \cdot 3^{n_1})$, and in the case of a ring of characteristic 2 the circuit complexity is bounded by $f(n) + 15.5n - 19 \cdot 3^{n_1}$.

In the general case, namely for $n \in \bigcup_i [3^i,\, 2\cdot 3^i)$, we are not able to obtain a complexity estimate of the form $(1+o(1))f(n)$ without modifying the basic multiplication algorithm. But it is possible to prove a weaker estimate $(4/3 + o(1))f(n)$.

Corollary 4. *Let $f(x)/x \to \infty$ while $x \to \infty$ and $f(x) = x^{O(1)}$ be satisfied additionally under the conditions of Corollary 3. Then, for multiplying polynomials of total degree at most $n-1$, one can construct a circuit of complexity no greater than*

$$\begin{cases} (1 + o(1))f(n), & 2 \cdot 3^k \leq n < 3^{k+1}, \\ \left(2 - 3^k/n + o(1)\right) f(n), & 3^k \leq n < 3^{k+1}/2, \\ \left(2 \cdot 3^k/n + o(1)\right) f(n), & 3^{k+1}/2 < n < 2 \cdot 3^k. \end{cases}$$

and depth at most $d_M(\lfloor \log_3(2n + o(n)) \rfloor - 1) + o(\log n)$.

Proof. In the first case, the construction is the same as in Corollary 2. In the third case, employ a circuit of multiplication of polynomials of total degree at most $2 \cdot 3^k - 1$.

In the case $n \in [3^k, 3^{k+1}/2)$, consider the nearest number n' from above, which is a multiple of $2 \cdot 3^{k-\alpha(n)}$, where $\alpha(n)$ is a slowly growing natural function. If $n' > 3^{k+1}/2$, then act as in the third case.

Otherwise, if $n' < 3^{k+1}/2$, compute the product modulo $x^{2\cdot 3^i} + x^{3^i} + 1$, where $i = k-1, \ldots, k-\alpha(n)$, by a circuit of complexity $f(3^k - 3^{k-\alpha(n)}) + O(n)$ and depth $O(\alpha(n))$ by the method of Theorem 3.

In fact, this gives us $3^k - 3^{k-\alpha(n)}$ linear equations for the coefficients of the desired product. To obtain the remaining $n'' = n' - 3^k + 3^{k-\alpha(n)}$ equations, note that $n'' = L + U$, where

$$2L,\, 2U \in \{0\} \cup \left(\bigcup_{j=1}^{\alpha(n)} [2 \cdot 3^{k-j},\, 3^{k-j+1}) \right) \cap \{2 \cdot 3^{k-\alpha(n)}\mathbb{N}\}.$$

Indeed, any natural number can be represented as a sum of two numbers whose base-3 notation consists only of zeros and ones. All such numbers are contained in the set $\{0\} \cup \bigcup_{j=1}^{\infty} [3^j,\, 3^{j+1}/2)$.

By separately multiplying the lower parts (of the polynomials) of total degree $2L-2$ and the higher parts of total degree $2U-2$, we determine L lower and U higher coefficients of the product. This can be done with complexity $f(2L) + f(2U) + O(L+U)$ and depth $O(\alpha(n))$ according to Theorem 3.

Next, by Lemma 5, find the remainders from dividing the known part of the desired product by the polynomials $x^{2\cdot 3^i} + x^{3^i} + 1$, $i = k-1, \ldots, k-\alpha(n)$, from which determine the remainders from dividing the unknown part $a(x)x^L$, where $\deg a < n' - L - U$, by the same polynomials. All this is performed with complexity $O(n)$ and depth $O(\alpha(n))$.

The polynomial $f(x)$ is reconstructed via a slightly modified method of Lemma 6 (see below; this modification is easy to construct), also with complexity $O(n)$ and depth $O(\alpha(n))$. □

Constructions and complexity characteristics of multiplication modulo $x^{2\cdot 3^k} + x^{3^k} + 1$ circuits are provided, e.g., in [7, 1, 12, 5, 6].

2.2 Auxillary statements

Let $a(x) = \sum_{l=0}^{n-1} a_l x^l$. Formally set $a_l = 0$ for $l \geq n$. Let us introduce the notation

$$a(x) \bmod (x^{3^k} - 1) = \sum_{l=0}^{3^k-1} a_{k,l} x^l, \quad a(x) \bmod (x^{2\cdot 3^k} + x^{3^k} + 1) = \sum_{r=0}^{1} \sum_{l=0}^{3^k-1} b_{k,r,l} x^l.$$

The coefficients $a_{k,l}$ and $b_{k,r,l}$ are connected by simple relations:

$$a_{k,l} = a_{k+1,l} + a_{k+1,3^k+l} + a_{k+1,2\cdot 3^k+l}, \quad b_{k,r,l} = a_{k+1,r3^k+l} - a_{k+1,2\cdot 3^k+l}.$$

These are the basis of

Lemma 4. *Let $n_{i+1} < k \le n_i$. The following formulas hold:*

$$
\begin{aligned}
a_{k,l} = \sum_{j_i=0}^{3^{n_i-k}-1} \Big(& b_{n_i,\,0,\,j_i3^k+l} + b_{n_i,\,1,\,j_i3^k+l} + \\
& + 3 \sum_{j_{i-1}=0}^{3^{n_{i-1}-n_i-1}-1} \Big(b_{n_{i-1},\,0,\,(3j_{i-1}+2)3^{n_i}+j_i3^k+l} + b_{n_{i-1},\,1,\,(3j_{i-1}+2)3^{n_i}+j_i3^k+l} + \\
& \quad + 3 \sum_{j_{i-2}=0}^{3^{n_{i-2}-n_{i-1}-1}-1} \Big(b_{n_{i-2},\,0,\,(3j_{i-2}+2)3^{n_{i-1}}+(3j_{i-1}+2)3^{n_i}+j_i3^k+l} + \\
& \qquad + b_{n_{i-2},\,1,\,(3j_{i-2}+2)3^{n_{i-1}}+(3j_{i-1}+2)3^{n_i}+j_i3^k+l} + \dots \\
& \quad \dots + 3 \sum_{j_1=0}^{3^{n_1-n_2-1}-1} \big(b_{n_1,\,0,\,(3j_1+2)3^{n_2}+\dots+(3j_{i-1}+2)3^{n_i}+j_i3^k+l} + \\
& \qquad + b_{n_1,\,1,\,(3j_1+2)3^{n_2}+\dots+(3j_{i-1}+2)3^{n_i}+j_i3^k+l} \big) \dots \Big) \Big) \Big) + 3^i a_{2(3^{n_1}+\dots+3^{n_i})+l};
\end{aligned}
$$

$$
\begin{aligned}
b_{n_i,r,l} = \sum_{j=0}^{1} (-1)^j \Bigg(& \sum_{j_{i-1}=0}^{3^{n_{i-1}-n_i-1}-1} \Big(b_{n_{i-1},\,0,\,(3j_{i-1}+r+j(2-r))3^{n_i}+l} + \\
& \qquad + b_{n_{i-1},\,1,\,(3j_{i-1}+r+j(2-r))3^{n_i}+l} + \\
& \quad + 3 \sum_{j_{i-2}=0}^{3^{n_{i-2}-n_{i-1}-1}-1} \Big(b_{n_{i-2},\,0,\,(3j_{i-2}+2)3^{n_{i-1}}+(3j_{i-1}+r+j(2-r))3^{n_i}+l} + \\
& \qquad + b_{n_{i-2},\,1,\,(3j_{i-2}+2)3^{n_{i-1}}+(3j_{i-1}+r+j(2-r))3^{n_i}+l} + \dots \\
& \dots + 3 \sum_{j_1=0}^{3^{n_1-n_2-1}-1} \big(b_{n_1,\,0,\,(3j_1+2)3^{n_2}+\dots+(3j_{i-2}+2)3^{n_{i-1}}+(3j_{i-1}+r+j(2-r))3^{n_i}+l} + \\
& + b_{n_1,\,1,\,(3j_1+2)3^{n_2}+\dots+(3j_{i-2}+2)3^{n_{i-1}}+(3j_{i-1}+r+j(2-r))3^{n_i}+l} \big) \dots \Big) \Big) \Big) + \\
& + 3^{i-1} \left(a_{2(3^{n_1}+\dots+3^{n_{i-1}})+r3^{n_i}+l} - a_{2(3^{n_1}+\dots+3^{n_i})+l} \right).
\end{aligned}
$$

Proof. The proof is completely analogous to the proof of Lemma 1, we only use the relations

$$
a_{k,l} = \begin{cases} a_{k+1,l} + a_{k+1,3^k+l} + a_{k+1,\,2\cdot3^k+l}, & k \notin \{n_i : i = 1, \dots, s\} \\ b_{k,0,l} + b_{k,1,l} + 3a_{k+1,\,2\cdot3^k+l}, & k \in \{n_i : i = 1, \dots, s\} \end{cases},
$$

taking into account that $a_{n_1+1,l} = a_l$, and, moreover, $a_l = 0$ for $l \ge n$. The second formula is obtained from the first as $b_{n_i,r,l} = a_{n_i+1,\,r3^{n_i}+l} - a_{n_i+1,\,2\cdot3^{n_i}+l}$. □

Lemma 5. *Let $m \le 2n$, and n, n_i be as in Theorem 3. Then the reduction of a polynomial of degree at most $m-1$ modulo $x^{2\cdot3^{n_i}} + x^{3^{n_i}} + 1$, $i = 1, \dots, s$, can be performed by a circuit*

a) of $2(m-1)$ additive elements and depth $2(n_1 - n_s) + 1$, where the coefficients of the remainder of division by $x^{2\cdot3^{n_i}} + x^{3^{n_i}} + 1$ are computed at depth $2(n_1 - n_i + 1)$. Moreover, in the case of a ring of characteristic 2, the complexity of the circuit us at most $1.5(m-1)$;

b) of $6n - 8 \cdot 3^{n_1}$ *additive elements,* $13n/8 - 11 \cdot 3^{n_1}/4$ *multiplications by powers of three, and space* n*.*

Proof. All sums $a_{k,l}$, where $k = n_s + 1, \ldots, n_1$, can be computed by a single circuit of complexity at most $m - 3^{n_s+1}$, in which the depth of computation of $a_{k,l}$ does not exceed $2(n_1 + 1 - k)$.

The coefficients $b_{n_i,r,l}$ of the remainder of division by $x^{2\cdot 3^{n_i}} + x^{3^{n_i}} + 1$ are obtained by adding to the constructed circuit the subtraction elements located at depth of 1 relative to $a_{n_i+1,l}$, in the amount of

$$\begin{cases} 0, & m \le 2 \cdot 3^{n_i}, \\ 2(m - 2 \cdot 3^{n_i}), & 2 \cdot 3^{n_i} \le m \le 3^{n_i+1}, \\ 2 \cdot 3^{n_i}, & m \ge 3^{n_i+1}. \end{cases}$$

The number of subtractions in the last two cases can also be estimated from above as $m - 3^{n_i}$. Since $3^{n_i} > 2(3^{n_{i+1}} + \ldots + 3^{n_s})$, for the number of subtractions (for a suitable i) we obtain the estimate

$$m - 3^{n_i} + 2\left(3^{n_{i+1}} + \ldots + 3^{n_s}\right) \le m - 1,$$

from which follows the estimate $2(m-1)$ for the complexity of the entire circuit.

If the characteristic is 2, then one of the two coefficients in each pair $b_{n_i,0,l}$, $b_{n_i,1,l}$ can be used to compute $a_{n_i,l}$ (except for the case $i = s$). For the complexity of the circuit in this case we have the estimate $m - 3^{n_s+1} + 0.5(m-1) + 3^{n_s} < 1.5(m-1)$. Item *a* is proven.

The proof of item *b* is sufficient to carry out for the case $m = n$. Let us reconstruct the circuit from the previous item so that it explicitly computes only those sums $a_{k,l}$ for which "there is enough memory".

Let $L_i = 2(3^{n_i} + \ldots + 3^{n_s})$. For any $k = n_1, \ldots, n_s + 1$, explicitly compute the sums $a_{k,0}, \ldots, a_{k,L_i-1}$, where $n_{i-1} \ge k > n_i$, taking into account that $n - L_i$ of the other "memory cells" are allocated for storing the coefficients $b_{n_j,r,l}$, where $j < i$.

The entire calculation can be divided into stages numbered from n_1 to n_s in descending order. At stage k, the sums $a_{k,0}, \ldots, a_{k,L_i-1}$ are computed and, if $k \in \{n_i : i = 1, \ldots, s\}$, then the coefficients $b_{k,r,l}$, $r \in \{0, 1\}$, $l = 0, \ldots, 3^k - 1$ are computed as well.

If the sum $a_{k,l}$ is not computed explicitly, then the right-hand side of the formula (denote it by $\psi_{k,l}$) from Lemma 4 is used instead, expressing $a_{k,l}$ through the known coefficients $b_{n_j,q,t}$, where $n_j \ge k$. Note that since $l \ge L_{i+1}$ for $n_i \ge k > n_{i+1}$, the last term $a_{2(3^{n_1}+\ldots+3^{n_i})+l}$ in $\psi_{k,l}$ is zero.

Let ρ_k denote the number of variables in the formula $\psi_{k,l}$ (this number does not depend on l, as follows from the form of the formula). Then the addition (subtraction) of $\psi_{k,l}$ is performed in ρ_k additive operations and i multiplications by powers of three without additional space, where $n_i \ge k > n_{i+1}$.

Let's estimate the complexity of the circuit. Consider stage k, where $n_{i-1} > k > n_i$. The available coefficients are $a_{k+1,0}, \ldots, a_{k+1,L_i-1}$, the remaining $a_{k+1,l}$ are expressed by the formulas $\psi_{k+1,l}$. Note that $L_i < 3^{n_i+1} \le 3^k$. Then each of the coefficients $a_{k,l}$, where $l < L_i$, can be computed as $a_{k+1,l} + \psi_{k+1,3^k+l} + \psi_{k+1,2\cdot 3^k+l}$ with complexity $2\rho_{k+1}$ additive operations and $i-1$ multiplications by powers of three (multiplications by powers of three

can be combined when adding two formulas of type ψ). The complexity of the stage is estimated as $2L_i\rho_{k+1}$ additive operations and $(i-1)L_i$ multiplications by powers of three.

Let us consider the case $k = n_i$. The available coefficients are $a_{k+1,0}, \ldots, a_{k+1,L_i-1}$. In this case, $L_i = 2 \cdot 3^k + L_{i+1}$. First, we compute all $b_{k,r,l}$ using the formulas $a_{k+1,r3^k+l} - a_{k+1,2\cdot 3^k+l}$ for $l < L_{i+1}$ and $a_{k+1,r3^k+l} - \psi_{k+1,2\cdot 3^k+l}$ for other l. In this case, the coefficients $b_{k,r,l}$ overwrite $a_{k+1,l}$, $l < 2 \cdot 3^k$. Then we compute $a_{k,l}$ for $l < L_{i+1}$ by the formulas $3a_{k+1,2\cdot 3^k+l} + b_{k,0,l} + b_{k,1,l}$. The complexity of this step is estimated as $4L_{i+1} + 2(3^k - L_{i+1})\rho_{k+1}$ additive operations and $2(i-1)3^k - (2i-3)L_{i+1}$ multiplications by powers of three. This estimate is also valid in the case $k = n_1$, since we can assume $\rho_{n_1+1} = 0$.

Let us find ρ_k. From the formula of Lemma 4 it follows directly that

$$\rho_{n_i} = 2\left(3^{n_1-n_i-(i-1)} + 3^{n_2-n_i-(i-2)} + \ldots + 3^{n_{i-1}-n_i-1} + 1\right), \tag{3}$$

and $\rho_k = 3^{n_i-k}\rho_{n_i}$ for $n_{i+1} < k < n_i$.

Let us estimate the total additive complexity of the computations. The sum of the complexities of the stages $n_i - 1, \ldots, n_{i+1}$ is at most

$$\begin{aligned} C_i = 2L_{i+1}\rho_{n_i}(1 + 3 + \ldots + 3^{n_i-n_{i+1}-2}) + 4L_{i+2} + 2(3^{n_{i+1}} - L_{i+2})3^{n_i-n_{i+1}-1}\rho_{n_i} \le \\ \le 2\rho_{n_i}3^{n_i-n_{i+1}-1}(\frac{1}{2}L_{i+1} - L_{i+2} + 3^{n_{i+1}}) - \rho_{n_i}L_{i+1} + 4L_{i+2} = \\ = 2\rho_{n_i}3^{n_i-n_{i+1}-1}(2 \cdot 3^{n_{i+1}} - \frac{1}{2}L_{i+2}) - \rho_{n_i}L_{i+1} + 4L_{i+2} = \\ = \rho_{n_i}(4 \cdot 3^{n_i-1} - L_{i+2}3^{n_i-n_{i+1}-1} - L_{i+1}) + 4L_{i+2}. \end{aligned}$$

For $i \ge 1$ the last expression does not exceed $4 \cdot 3^{n_i-1}\rho_{n_i}$, if we take into account that $L_{i+1} > 3L_{i+2}$ and $\rho_{n_i} \ge 1$.

For the complexity of all stages except stage n_1, using (3), we obtain the estimate

$$\begin{aligned} C_1 + \ldots + C_{s-1} = 4\sum_{i=1}^{s-1} 3^{n_i-1}\rho_{n_i} = 8\sum_{i=1}^{s-1}(3^{n_1-i} + 3^{n_2-(i-1)} + \ldots + 3^{n_i-1}) < \\ < 8\left((3^{n_1-1} + 3^{n_1-2} + \ldots) + (3^{n_2-1} + 3^{n_2-2} + \ldots) + \ldots + (3^{n_s-1} + 3^{n_s-2} + \ldots)\right) < 2n. \end{aligned}$$

Finally, estimating the complexity of stage n_1 as $4L_2 = 4(n - 2 \cdot 3^{n_1})$, we obtain the assertion of item b in terms of additive complexity.

The number of multiplications by powers of three at stages $n_i, \ldots, n_{i+1}+1$ is estimated as

$$\begin{aligned} D_i = (n_i - n_{i+1} - 1)iL_{i+1} + 2(i-1)3^{n_i} - (2i-3)L_{i+1} = \\ = i(n_i - n_{i+1})L_{i+1} + (i-1)(L_i - 4L_{i+1}). \end{aligned}$$

The sum of the first terms can be estimated as

$$\begin{aligned} \sum_{i=1}^{s} i(n_i - n_{i+1})L_{i+1} \le \sum_{i=1}^{s} i3^{n_i-n_{i+1}-1}(3^{n_{i+1}+1}/2) = \frac{1}{2}\sum_{i=1}^{s} i3^{n_i} < \\ < \frac{1}{2}\left((3^{n_1} + 3^{n_2} + \ldots) + (3^{n_2} + 3^{n_3} + \ldots) + \ldots\right) < \\ < \frac{1}{2}\left(\frac{n}{2} + (n/2 - 3^{n_1}) + \frac{1}{3}(n/2 - 3^{n_1}) + \ldots\right) = \frac{n}{4} + \frac{3}{4}(n/2 - 3^{n_1}). \end{aligned}$$

We will roughly estimate the sum of the second terms as

$$(L_2 - 4L_3) + 2(L_3 - 4L_4) + 3(L_4 - 4L_5) + \ldots \le L_2 = n - 2 \cdot 3^{n_1}.$$

Summing the last two estimates, we complete the proof of item *b*. □

Lemma 6. *Reconstruction of a polynomial of degree $n-1$ from given remainders from division by polynomials $x^{2\cdot 3^{n_i}} + x^{3^{n_i}} + 1$, $i = 1, \ldots, s$, can be performed by a circuit*

a) of $3.5n - 5 \cdot 3^{n_1}$ additive elements, $1.5(n - 2 \cdot 3^{n_1})$ multiplications by powers of three and depth $2(n_1 - n_s) + s + 1$;

b) of $3.5n - 3^{n_1+1}$ additive elements, $1.5(n - 2 \cdot 3^{n_1})$ divisions by 3 and memory space n.

Proof. From Lemma 4 express the differences

$$h_{i,r,l} = a_{2(3^{n_1}+\ldots+3^{n_{i-1}})+r3^{n_i}+l} - a_{2(3^{n_1}+\ldots+3^{n_i})+l}, \tag{4}$$

where $i = 1, \ldots, s-1$, $r \in \{0, 1\}$ and $l = 0, \ldots, L_i - 1$, in terms of the coefficients $b_{n_j,q,t}$. Note that the subtrahend coefficient is zero for $l \ge L_{i+1}$.

To construct the circuit from item *a*, compute the auxiliary quantities $c_{i,l}$ and $d_{i,l}$, defined by the equalities

$$d_{1,l} = b_{n_1,0,l} + b_{n_1,1,l}, \quad c_{i,l} = \sum_{j=0}^{3^{n_{i-1}-n_i-1}-1} d_{i-1,\, j3^{n_i+1}+l}, \quad d_{i,l} = b_{n_i,0,l} + b_{n_i,1,l} + 3c_{i,2\cdot 3^{n_i}+l},$$

where $i \ge 2$. The desired coefficients $h_{i,r,l}$ for $i \ge 2$ are expressed by the formulas $h_{i,r,l} = 3^{1-i}(b_{n_i,r,l} - c_{i,r3^{n_i}+l} + c_{i,2\cdot 3^{n_i}+l})$, and for $i = 1$ simply as $h_{1,r,l} = b_{n_1,r,l}$.

The computational complexity implied in the formulas for $c_{i,l}$, $i = 2, \ldots, s$ and $l = 0, \ldots, 3^{n_i+1} - 1$ is estimated as

$$\sum_{i=2}^{s} 3^{n_i+1}(3^{n_{i-1}-n_i-1} - 1) = \sum_{i=2}^{s}(3^{n_{i-1}} - 3^{n_i+1}) < \frac{n}{2} - 3(n/2 - 3^{n_1}) = 3^{n_1+1} - n.$$

The complexity implied in the formulas for $d_{i,l}$, $i = 1, \ldots, s-1$ and $l = 0, \ldots, 3^{n_i} - 1$ is estimated as $3^{n_1} + 2(3^{n_2} + \ldots + 3^{n_{s-1}}) < n - 3^{n_1}$ additive operations and $3^{n_2} + \ldots + 3^{n_{s-1}} < n/2 - 3^{n_1}$ multiplications by 3. To complete the computation of $h_{i,r,l}$, we need to perform another $4(3^{n_2} + \ldots + 3^{n_s}) = 2(n - 2 \cdot 3^{n_1})$ additive operations and $n - 2 \cdot 3^{n_1}$ multiplications by powers of three.

It is easy to verify that $c_{i,l}$ is computed at depth $2(n_1 - n_i) - 1$ and, consequently, $h_{i,r,l}$ — at depth $2(n_1 - n_i) + 2$.

From the computed $h_{i,r,l}$, the coefficients of $a(x)$ are easily reconstructed. The coefficients $a_{2(3^{n_1}+\ldots+3^{n_{i-1}})+r3^{n_i}+l}$ coincide with $h_{i,r,l}$ for $l \ge L_{i+1}$. In particular, in the case $i = s$, all coefficients are known. This allows us to determine sequentially in descending order of i the remaining coefficients $a_{2(3^{n_1}+\ldots+3^{n_{i-1}})+r3^{n_i}+l}$, $l = 0, \ldots, L_{i+1} - 1$, from formulas (4) with complexity

$$L_s + L_{s-1} + \ldots + L_2 = 2(3^{n_2} + 2 \cdot 3^{n_3} + 3 \cdot 3^{n_4} + \ldots) < 1.5(n - 2 \cdot 3^{n_1})$$

additive operations. The depth of these computations obviously does not exceed $s - 1$.

Summing all the estimates, we obtain the assertion of item *a*.

To prove item b, we construct a circuit that sequentially in descending order of i transforms each coefficient $b_{n_i,r,l}$ according to Lemma 4 into the corresponding difference $h_{i,r,l}$.

It is easy to see that the transformation of each coefficient $b_{n_i,r,l}$ is performed in τ_i additive operations and $i-1$ divisions by 3, where τ_i is the number of coefficients $b_{n_j,q,t}$ on the right-hand side of the second formula of Lemma 4. It can be directly verified that

$$\tau_i = 4\left(3^{n_1-n_i-(i-1)} + 3^{n_2-n_i-(i-2)} + \ldots + 3^{n_{i-1}-n_i-1}\right).$$

The additive complexity of computing the differences $h_{i,r,l}$ can now be estimated as

$$\sum_{i=2}^{s} 2\cdot 3^{n_i}\tau_i = \sum_{i=2}^{s} 8\left(3^{n_1-(i-1)} + 3^{n_2-(i-2)} + \ldots + 3^{n_{i-1}-1}\right) <$$
$$8\left((3^{n_1-1} + 3^{n_1-2} + \ldots) + (3^{n_2-1} + 3^{n_2-2} + \ldots) + \ldots + (3^{n_s-1} + 3^{n_s-2} + \ldots)\right) < 2n.$$

The number of divisions by 3 is estimated as

$$\sum_{i=2}^{s} 2(i-1)3^{n_i} = 2\left(3^{n_2} + 2\cdot 3^{n_3} + 3\cdot 3^{n_4} + \ldots\right) < 1.5(n - 2\cdot 3^{n_1}).$$

The final part of the circuit is the same as in item a. □

References

[1] Bernstein D. J. Fast multiplication and its applications // in: Algorithmic Number Theory, MSRI Publ. 2008. V. 44. P. 325–384.

[2] Cantor D., Kaltofen E. On fast multiplication of polynomials over arbitrary algebras // Acta Inf. 1991. V. 28. no. 7. P. 693–701.

[3] Cooley J., Tukew J. An algorithm for the machine calculation of complex Fourier series // Math. Comp. 1965. V. 19. P. 297–301.

[4] Crandall R., Fagin B. Discrete weighted transforms and large-integer arithmetic // Math. of Comput. 1994. V. 62. P. 305–324.

[5] Gashkov S. B., Sergeev I. S. The complexity and depth of Boolean circuits for multiplication and inversion in some fields $GF(2^n)$ // Moscow Univ. Math. Bulletin. 2009. V. 64, no. 4. P. 139–143.

[6] Gashkov S. B., Sergeev I. S. Fast Fourier transform algorithms // in: "Discrete mathematics and its applications". Part V. Moscow: Izd. IPM RAN, 2009. P. 3–23. (in Russian)

[7] von zur Gathen J., Gerhard J. Modern computer algebra. Cambridge: Cambridge University Press, 1999. 768 p.

[8] Harvey D., Roche D. S. An in-place truncated Fourier transform and application to polynomial multiplication // Proc. ISSAC 2010 (Munich, Germany). NY: ACM Press, 2010. P. 325–329.

[9] van der Hoeven J. The truncated Fourier transform and applications // Proc. ISSAC 2004 (Santander, Spain). NY: ACM Press, 2004. P. 290–296.

[10] van der Hoeven J. Notes on the truncated Fourier transform // Tech. report. Univ. Paris-Sud, Orsay, France, 2005.

[11] Lupanov O. B. Asymptotic estimates of the complexity of control systems. Moscow: Izd. MGU, 1984. 138 p. (in Russian)

[12] Mateer T. Fast Fourier algorithms with applications // Ph. D. Thesis. Clemson University, 2008.

[13] Schönhage A. Schnelle Multiplikation von Polynomen über Körpern der Charakteristik 2 // Acta Inf. 1977. V. 7. P. 395–398.

[14] Schönhage A. Asymptotically fast algorithms for the numerical multiplication and division of polynomials with complex coefficients // Proc. EuroCAM–82 (Marseille, France). LNCS. V. 144. Berlin, Heidelberg, NY: Springer, 1982. P. 3–15.

[15] Sergeev I. S. Regularization of some estimates of the complexity of polynomial multiplication // Proc. Youth Scientific School on Discrete Math. and its Appl. (Moscow, 2009). Part II. Moscow: Izd. IPM RAN, 2009. P. 26–32. (in Russian)

[16] Yablonskii S. V. Introduction to discrete mathematics. Moscow: Nauka, 1986. 384 p. (in Russian)