Supported by the National Science Foundation Engineering Research Center Program (NSFD CD 8803012), Industry and the University

Fast Algorithms for 2-D Circular Convolutions and Number Theoretic Transforms Based on Polynomial Transforms over Finite Rings

by X. Ran and K.J.R. Liu

TR 92-88

Fast Algorithms for 2-D Circular Convolutions and Number Theoretic Transforms Based on Polynomial Transforms Over Finite Rings

Xiaonong Ran and K. J. Ray Liu
Electrical Engineering Department
and
Systems Research Center
University of Maryland
College Park, Maryland 20742

Abstract

In this paper, we develop new fast algorithms for 2-D integer circular convolutions and 2-D Number Theoretic Transforms (NTT). These new algorithms, which offers improved computational complexity, are constructed based on polynomial transforms over Z_p; these transforms are Fourier-like transforms over $Z_p[x]$ which is the integral domain of polynomial forms over Z_p. Having defined such polynomial transforms over Z_p, we prove several necessary and sufficient conditions for their existence. We then apply the existence conditions to recognize two applicable polynomial transforms over Z_p: One is for p equal to Mersenne numbers and the other for Fermat numbers. Based on these two transforms, referred to as Mersenne Number Polynomial Transforms (MNPT) and Fermat Number Polynomial Transforms (FNPT), we develop fast algorithms for 2–D integer circular convolutions, 2-D Mersenne Number Transforms, and 2-D Fermat Number Transforms. As compared to the conventional row-column computation of 2-D NTT for 2–D integer circular convolutions and 2-D NTTs, the new algorithms give rise to reduced computational complexities by saving more than 25% or 42% in numbers of operations for multiplying 2^i, $i \geq 1$; these percentages of savings also grow with the size of the 2-D integer circular convolutions or the 2-D NTTs.

I Introduction

In the applications of image/video processing and coding, the computation of 2-D (linear) convolutions are of considerable importance [1] [2]. For instance, in the well-known subband image coding scheme, a full-band image is split into subbands by means of 2-D filter bank before the encoding operations [3]. Due to the huge number of data samples associated with these applications, direct-computation of 2-D convolutions is obviously impossible, and, thus, various fast algorithms are used [4] [5]. A popular technique for computing 2-D convolution is to convert the original 2-D convolution into a 2-D circular convolution, and to utilize the fast transform algorithms to compute the 2-D circular convolution [4]. In this paper, we will develop new transform-based fast algorithms for the computation of 2-D circular convolutions.

In the digital signal processing applications, the values of signal samples are typically represented using a finite alphabet. Because of this finite resolution, the number system of finite rings, e.g., $Z_p = \{0, 1, \ldots, p-1\}$ [6], can be used for the calculations in digital signal processing. More precisely, in the case of a convolution[1]:

$$y_i = \sum_k h_k x_{i-k},$$

where h_k and x_i assume integer values, the output y_i also assume integer values. Supposing the absolute values of y_i are upper bounded by a positive number M, then if $M < p/2$, the output, $v_i = \sum_k h_k x_{i-k} \bmod p$, of the same convolution operated on Z_p can be translated uniquely to y_i: $y_i = v_i$ if $v_i < p/2$; $y_i = v_i - p$ if $v_i \geq p/2$. Performing the calculations such as convolutions on Z_p rather than on the ordinary integer set has several advantages: (i) the residue arithmetic associated with the operations on Z_p can be implemented relatively cheaply and performed very efficiently, especially in parallel and pipeline systems [7], (ii) there are no round-off errors, and (iii) there exist very efficient algorithms (e.g., Number Theoretic Transforms (NTT)) [4] [5] [8], and more efficient algorithms, such as the ones for 2-D circular convolutions presented in this paper, can be developed.

In this paper, we focus on the development of polynomial transforms over a specific finite ring, namely, Z_p, [6] whose definition will be stated later on, and the fast algorithms

[1]The computational operations needed in a convolution are additions, subtractions, and multiplications.

associated with these transforms for 2-D circular convolutions. These polynomial transforms over Z_p are Fourier-like transforms over $Z_p[x]$ of polynomial forms [6]. An existence theorem for such Fourier-like transforms over arbitrary finite commutative rings with unity is given by Dubois and Venetsanopoulos [9] with applications for 1-D circular convolutions. However, the condition of the existence theorem in [9] is in general difficult to apply, due to the computational difficulty of various parameters. Maher later pointed out in [10] that, in most practical cases of concern, the rings may be characterized as algebraic extensions of finite rings, and that in these cases there is an existence theorem which is much easier to apply as compared to that of [9]. Maher also mentioned the computation of 2-D circular convolutions as an application for such Fourier-like transforms, motivated by polynomial transforms [11]; but the algorithm is very briefly described based on a special case, and it is not clear whether or how much one can save in terms of computational complexity by using the new technique as compared with the existing algorithms such as 2-D NTTs. In this paper, we will follow the general direction of above while concentrating on the application of 2-D circular convolutions. We will define a group of Fourier-like transforms over $Z_p[x]$, called polynomial transformed over Z_p, and will give several necessary and sufficient conditions for their existence. These conditions will be applied in a rather straight forward manner to obtain two specific groups of transforms, namely, Mersenne number polynomial transforms (MNPT) and Fermat number polynomial transforms (FNPT) which are of direct applications to the computation of 2-D circular convolutions. The complete algorithms will be provided along with the computational complexity analysis and comparisons against 2-D NTTs. New fast algorithms for 2-D NTTs are also developed based on MNPTs and FNPTs. The results of the complexity analysis show that new fast algorithms offer reduced complexities in terms of numbers of computational operations. More precisely, new fast algorithms give rise to savings on the numbers of the operation for multiplying a number by 2^i, $i \geq 1$; these savings are more than 25% or 42% (and are growing with the size of 2-D circular convolution or 2-D NTT) of the numbers of such operations in the conventional row-column computation of 2-D NTTs. These complexity savings of the new algorithms are also confirmed by a simulation experiment on the actual computing time.

The paper is organized as follows. Section II contains preliminaries. In Section III, polynomial transforms over Z_p are defined, their necessary and sufficient conditions of existence are proved, and then the MNPTs and FNPTs are introduced. Applications of the introduced transforms to the computations of 2-D circular convolutions and 2-D NTTs are included in Section IV and V. The analysis and comparisons of computational complexity are presented in Section VI. Finally, Section VII contains a summary and conclusions.

II Preliminaries

In this section, we introduce the notations, the basic definitions and a preliminary theorem.

We denote the set of all integers by Z, and the commutative ring: $\{0, 1, \cdots, p-1\}$ with addition and multiplication modulo the integer $p \geq 2$ by Z_p [6]. The equality of two numbers a and b in Z_p are denoted by $a \equiv b \bmod p$. We also use the notation $\langle a \rangle_p$ for the modulo p arithmetic on a, for example, $\langle 4 \rangle_3 = 1$. A polynomial $f(x)$ with its coefficients in Z_p is called a polynomial over Z_p. The set of all polynomials over Z_p is denoted by $Z_p[x]$ (which is an integral domain containing Z_p [6]). The equality of two polynomials $f(x)$ and $g(x)$ in $Z_p[x]$ are denoted by $f(x) \equiv g(x) \bmod p$. If $f(x), g(x) \in Z_p[x]$ are congruent modulo $h(x) \in Z_p[x]$, i.e., $f(x) - g(x) \equiv h(x)m(x) \bmod p$ for some $m(x) \in Z_p[x]$, we write

$$f(x) \equiv g(x) \text{ modd } p, h(x).$$

We will use "|" to denote the divisibility, e.g., $a|b$ means $b = ac$ for integers a, b and c; and $a|b \bmod p$ indicates $b \equiv ac \bmod p$ for a, b and c in Z_p.

The following theorem is actually a version of the Chinese Remainder Theorem[2] stated for polynomials in $Z_p[x]$, which will be referred repeatedly in the later discussions.

Theorem II.1 (Chinese Remainder Theorem on $Z_p[x]$): For a prime number p, let $m_1(x), m_2(x), \ldots, m_k(x)$ be k polynomials in $Z_p[x]$ which are coprime with each other, i.e., their greatest common divisor is 1. Define

$$m(x) \equiv m_1(x)m_2(x)\cdots m_k(x) \mod p$$
$$m(x) \equiv m_i(x)M_i(x) \mod p, \; i = 1, 2, \cdots, k.$$

[2] A proof of the Chinese Remainder Theorem for polynomials of real coefficients can be found in [5].

Then the following system of congruent equations

$$
\begin{aligned}
f(x) &\equiv b_1(x) \mod d\, p, m_1(x) \\
f(x) &\equiv b_2(x) \mod d\, p, m_2(x) \\
&\cdots\cdots \\
f(x) &\equiv b_k(x) \mod d\, p, m_k(x)
\end{aligned}
\tag{II.1}
$$

has the solution

$$
f(x) \equiv M_1(x)M_1'(x)b_1(x) + M_2(x)M_2'(x)b_2(x) + \cdots + M_k(x)M_k'(x)b_k(x) \mod d\, p, m(x) \tag{II.2}
$$

where

$$
M_i(x)M_i'(x) \equiv 1 \mod d\, p, m_i(x),\ i = 1, 2, \cdots, k. \tag{II.3}
$$

Moreover, when p is not a prime number, (II.2) is the solution of (II.1) modd $p, m(x)$ if the leading coefficients of $m(x), m_i(x)$, $1 \le i \le k$, are the unity in Z_p, and polynomials $M_i(x)M'_i(x)$, $1 \le i \le k$, defined in (II.3) exist.

Proof: Since $(m_i(x), m_j(x)) \equiv 1 \bmod p$, for $i \neq j$, we have $(M_i(x), m_i(x)) \equiv 1 \bmod p$, for all i. Thus, from Theorem A.4 in Appendix A, for each $M_i(x)$, there exists $M_i'(x)$ such that $M_i(x)M_i'(x) \equiv 1 \mod d\, p, m_i(x)$.

On the other hand, from $m(x) \equiv m_j(x)M_j(x) \bmod p$, we have $m_i(x) \mid M_j(x) \bmod p$, $i \neq j$, and, thus,

$$
\sum_{j=1}^{k} M_j(x)M_j'(x)b_j(x) \equiv M_i(x)M_i'(x)b_i(x) \equiv b_i(x) \mod d\, p, m_i(x),
$$

for $i = 1, 2, \cdots, k$, i.e., (II.2) is a solution of (II.1).

Now, suppose $f_1(x)$ and $f_2(x)$ are two solutions of (II.1). Then $f_1(x) \equiv f_2(x) \mod d\, p, m_i(x)$, $i = 1, 2, \cdots, k$. Since $(m_i(x), m_j(x)) \equiv 1 \bmod p$, $i \neq j$, then $f_1(x) \equiv f_2(x) \mod d\, p, m(x)$, i.e., the solution of (II.1) is unique modd $p, m(x)$. QED.

III Polynomial Transforms Over Finite Rings

In this section, we define polynomial transforms over Z_p and provide several necessary and sufficient conditions for their existence.

Definition III.1: Let $M(x), g(x) \in Z_p[x]$, and $\{H_m(x)\}_{m=0}^{N-1} \subset Z_p[x]$. We call

$$\overline{H}_k(x) \equiv \sum_{m=0}^{N-1} H_m(x)[g(x)]^{mk} \ \operatorname{modd} p,\ M(x),\ k = 0, 1, \cdots, N-1, \tag{III.1}$$

a polynomial transform over Z_p (or simply, a polynomial transform) of $\{H_m(x)\}_{m=0}^{N-1}$ modd $p, M(x)$. Its inverse transform is defined by

$$H_l(x) \equiv N^{-1} \sum_{k=0}^{N-1} \overline{H}_k(x)[g(x)]^{-kl} \ \operatorname{modd} p, M(x),\ l = 0, 1, \cdots, N-1, \tag{III.2}$$

where $N^{-1}N \equiv 1 \bmod p$ and $[g(x)]^{-kl}[g(x)]^{kl} \equiv 1$ modd $p, M(x)$. We denote such a polynomial transform by $(N, g(x), p, M(x))$.

Theorem III.2: (A necessary and sufficient condition for polynomial transforms over Z_p) Assuming p is prime, polynomial transform $(N, g(x), p, M(x))$ exists if and only if

$$S(q) \equiv \sum_{m=0}^{N-1} [g(x)]^{mq} = \begin{cases} N \operatorname{modd} p, M(x) & \text{if } q \equiv 0 \bmod N \\ 0 \ \operatorname{modd} p, M(x) & \text{if } q \not\equiv 0 \bmod N \end{cases}, \tag{III.3}$$

and $(p, N) = 1$. Moreover, when the leading coefficient of $M(x)$ is the unity in Z_p, the above statement is also true for p not being prime.

Proof: For p being prime, "if" part: when (III.3) holds (assuming $N > 1$)

$$[g(x)]^N - 1 \equiv (g(x) - 1) \sum_{k=0}^{N-1} [g(x)]^k \equiv 0 \ \operatorname{modd} p, M(x). \tag{III.4}$$

Thus, $[g(x)]^{-1} \equiv [g(x)]^{N-1}$ modd $p, M(x)$. Now substitute (III.1) into (III.2), and define

$$R_l(x) \equiv \sum_{m=0}^{N-1} H_m(x) N^{-1} \sum_{k=0}^{N-1} [g(x)]^{k(m-l)} \ \operatorname{modd} p, M(x), \tag{III.5}$$

for $l = 0, 1, \cdots, N-1$. From the conditions in the theorem, we have

$$R_l(x) \equiv H_l(x),\ l = 0, 1, \cdots, N-1. \tag{III.6}$$

"Only if" part: Suppose $(N, g(x), p, M(x))$ exists, i.e., (III.6) holds. Then the second case of (III.3) is true. Otherwise, there is some $t, 1 \le t \le N-1$, such that $S(t) \not\equiv 0$ modd $p, M(x)$. Then (III.6) can not always hold, e.g., let $H_t(x) \equiv 1$ and $H_i(x) \equiv 0$ for $i \neq t$, then (III.5) becomes

$$R_l(x) \equiv N^{-1} \sum_{k=0}^{N-1} [g(x)]^{k(t-l)} \ \operatorname{modd} p, M(x),$$

for $l = 0, 1, \cdots, N-1$. Thus, $R_0(x) \equiv S(t) \not\equiv 0$ modd $p, M(x)$, and this is a contradiction. From the second case of (III.3) and (III.4), we have $[g(x)]^N \equiv 1$ modd $p, M(x)$, i.e., the first case of (III.3) holds. Finally, since the inverse transform exists, N^{-1} exists, thus $(p, N) = 1$, because, otherwise, $(p, N) = a > 1$, i.e., $p = b_1 a$ and $N = b_2 a$ for some b_1 and b_2; from $NN^{-1} \equiv 1 \bmod p$, we have $NN^{-1} - 1 = cp$ for some c, i.e., $b_2 a N^{-1} - 1 = cb_1 a$ or $(b_2 N^{-1} - cb_1)a = 1$ which is a contradiction.

When p is not a prime number, the whole proof goes through unless we can carry out arithmetic modulo $M(x)$ on Z_p, while these operations is guaranteed if the leading coefficient of $M(x)$ is unit of Z_p. QED.

There are two other theorems about the necessary and sufficient conditions of the existence of the polynomial transforms over Z_p, which are stated in the following and are proved in Appendix B.

Theorem III.3: Let $M(x) \equiv cb_1^{l_1}(x)b_2^{l_2}(x)...b_s^{l_s}(x)$, $b_i(x)$, $1 \leq i \leq s$, be distinct irreducible polynomials with unity leading coefficients, and c be a non-zero constant. Then $(N, g(x), p, M(x))$ exists if and only if (i) $[g(x)]^N \equiv 1$ modd $p, M(x)$, (ii) the order of $g(x)$ modd $p, b_i(x)$ is N, for $1 \leq i \leq s$.

The next theorem is also a necessary and sufficient condition like the above two, but only dealing with a special case of Theorem III.3, when $M(x)$ is a product of distinct irreducible polynomials mod p.

Theorem III.4: Let $M(x) = b_1(x)b_2(x)...b_s(x) \bmod p$, and $b_i(x)$ are distinct irreducible polynomials. Then, $(N, g(x), p, M(x))$ exists for some $g(x)$, if and only if N divides the greatest common divisor of $p^{n_1} - 1, p^{n_2} - 1, ..., p^{n_s} - 1$, where n_i is the degree of $b_i(x), 1 \leq i \leq s$.

The above necessary and sufficient conditions only tell us that if a given set of numbers and polynomials: $N, g(x), p$ and $M(x)$, satisfies certain conditions, then polynomial transform $(N, g(x), p, M(x))$ exists; they do not describe what these numbers and polynomials are. In the next two subsections, we introduce two groups of such numbers and polynomials, and then use the above conditions to show that they form polynomial transforms over Z_p.

III.A Mersenne Number Polynomial Transforms

Theorem III.5: There is $(N, x, M_N, (x^N - 1)/(x - 1))$, for each prime number N and Mersenne number $M_N = 2^N - 1$.

Proof: Since $\left(x^N - 1\right)/(x-1)$ is a factor of $x^N - 1$, $S(q) \equiv \sum_{m=0}^{N-1} x^{mq} \equiv N$ modd M_N, $(x^N - 1)/(x-1)$, when $q \equiv 0 \bmod N$. On the other hand, when $q \not\equiv 0 \bmod N$, $\{\langle mq\rangle_N\}_{m=0}^{N-1}$ is a permutation of $\{m\}_{m=0}^{N-1}$, thus $S(q) \equiv 0$ modd M_N, $\left(x^N - 1\right)/(x-1)$. We have $(M_N, N) = 1$ since $2^N - 2 \equiv 0 \bmod N$ by Fermat theorem [12], i.e., $N|(M_N - 1)$. Thus, based on Theorem III.2, we have $(N, x, M_N, (x^N - 1)/(x-1)$. QED.

The above polynomial transforms $(N, x, M_N, (x^N - 1)/(x - 1))$ are called *Mersenne Number Polynomial Transforms* (MNPT), whose applications will be described in the next two sections.

Example 1: For $N = 3$, we have $(3, x, 7, (x^3 - 1)/(x - 1))$. Given polynomial sequence $H_0(x) = x + 1$, $H_1(x) = x - 1$ and $H_2(x) = x$, the corresponding MNPT are

$$\overline{H}_k(x) \equiv \sum_{m=0}^{2} H_m(x) x^{mk} \text{ modd } 7, x^2 + x + 1,$$

for $k = 0, 1, 2$. The results are $\overline{H}_0(x) = 3x$, $\overline{H}_1(x) = 6x + 1$ and $\overline{H}_2(x) = x + 2$. It can be easily verified that

$$H_l(x) \equiv 3^{-1} \sum_{k=0}^{2} \overline{H}_k(x) x^{-kl} \text{ modd } 7, x^2 + x + 1,$$

for $l = 0, 1, 2$, where $3^{-1} \equiv 5 \bmod 7$ and $x^{-1} \equiv (6x + 6)$ modd $7, x^2 + x + 1$.

Some useful properties related to MNPT are given in Appendix C.

III.B Fermat Number Polynomial Transforms

Theorem III.8: There are $\left(2^{t-i+1}, x, F_t, x^{2^{t-i}} + 1\right)$, $0 \leq i \leq t$, and $\left(2^{t-i}, x^2, F_t, x^{2^{t-i}} + 1\right)$, $0 \leq i \leq t - 1$, for each positive number t and Fermat number $F_t = 2^{2^t} + 1$.

Proof: We prove the case for $\left(2^{t-i+1}, x, F_t, x^{2^{t-i}} + 1\right)$, $0 \leq i \leq t$; the other case can be similarly proved. When $q \equiv 0 \bmod 2^{t-i+1}$, since $\left(x^{2^{t-i+1}} - 1\right) = \left(x^{2^{t-i}} + 1\right)\left(x^{2^{t-i}} - 1\right)$, we have

$$S(q) \equiv \sum_{m=0}^{2^{t-i+1}-1} x^{mq} \equiv 2^{t-i+1} \text{ modd } F_t, x^{2^{t-i}} + 1,$$

for $0 \leq i \leq t$. When $q \not\equiv 0 \mod 2^{t-i+1}$, $S(q) \equiv \left(x^{2^{t-i+1}q} - 1\right)/(x^q - 1)$. Since $x^{2^{t-i+1}q} - 1$ has factors $x^q - 1$ and $x^{2^{t-i+1}} - 1$, if we can show that $x^q - 1$ and $x^{2^{t-i}} + 1$ are coprime, then $S \equiv 0 \text{ modd } F_t, x^{2^{t-i}} + 1$. Indeed, we have $\left(q, 2^{t-i+1}\right) \leq 2^{t-i}$ in this case. Thus, $\left(x^q - 1, x^{2^{t-i+1}} - 1\right) = x^{2^a} - 1$ with $a \leq t - i$ [12]. Therefore, $x^q - 1$ and $x^{2^{t-i}} + 1$ are coprime, because $x^{2^a} - 1$, $a \leq t - i$, and $x^{2^{t-i}} + 1$ are coprime from Corollary C.5 in Appendix C. Also we have $\left(2^{t-i+1}\right)^{-1} \equiv 2^{2^{t+1}-(t-i+1)} \mod F_t$. Based on Theorem III.2, $\left(2^{t-i+1}, x, F_t, x^{2^{t-i}} + 1\right)$ exists for $0 \leq i \leq t$. QED.

The above polynomial transforms $\left(2^{t-i+1}, x, F_t, x^{2^{t-i}} + 1\right)$, $0 \leq i \leq t$, and $\left(2^{t-i}, x^2, F_t, x^{2^{t-i}} + 1\right)$, $0 \leq i \leq t-1$, are called *Fermat Number Polynomial Transforms* (FNPT); their applications will be investigated later on in sections IV and V.

Example 2: For $t = 1$ and $i = 0$, we have $(4, x, 5, x^2 + 1)$. Given polynomial sequence $H_0(x) = x + 1$, $H_1(x) = x - 1$, $H_2(x) = x$ and $H_3(x) = 1$, the corresponding FNPT are

$$\overline{H}_k(x) \equiv \sum_{m=0}^{3} H_m(x) x^{mk} \text{ modd } 5, x^2 + 1,$$

for $k = 0, 1, 2, 3$. The results are $\overline{H}_0(x) = 3x + 1$, $\overline{H}_1(x) = 3x$ $\overline{H}_2(x) = x + 1$ and $\overline{H}_3(x) = 2x + 2$. It can be easily verified that

$$H_l(x) \equiv 4^{-1} \sum_{k=0}^{3} \overline{H}_k(x) x^{-kl} \text{ modd } 5, x^2 + 1,$$

for $l = 0, 1, 2, 3$, where $4^{-1} \equiv 4 \mod 5$ and $x^{-1} \equiv 4x \text{ modd } 5, x^2 + 1$.

Some useful properties related to FNPT are given in Appendix C.

IV Fast Algorithms for 2-D Circular Convolutions

We now use the polynomial transforms over Z_p to develope new fast algorithms for the computation of 2-D circular convolutions (CC). Let us consider a 2-D $N \times N$ CC

$$y_{l,u} = \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} h_{m,n} q_{\langle l-m \rangle_N, \langle u-n \rangle_N}, \quad l, u = 0, 1, \cdots, N-1, \qquad \text{(IV.1)}$$

of 2-D data $\{h_{i,j}\}$ and $\{q_{i,j}\}$, $i, j = 0, 1, \cdots, N-1$, which are assumed to be *integers* without loss of generality in digital signal processing practice. Apparently, this integer

2-D CC can be performed on Z_p if p is large enough; choices of p are given by

$$\frac{p}{2} > \min\left\{ \left|h_{m,n}\right|_{\max} \sum_{r=0}^{N-1}\sum_{s=0}^{N-1} \left|q_{r,s}\right|, \left|q_{r,s}\right|_{\max} \sum_{n=0}^{N-1}\sum_{m=0}^{N-1} \left|h_{m,n}\right| \right\}. \tag{IV.2}$$

This 2-D integer CC can be also written as a 1-D polynomial CC

$$Y_l(x) \equiv \sum_{m=0}^{N-1} H_m(x) Q_{\langle l-m\rangle_N}(x) \ \operatorname{modd} p, x^N - 1, \tag{IV.3}$$

where

$$\begin{aligned} H_m(x) &= \sum_{n=0}^{N-1} h_{m,n}x^n,\ m = 0, 1, \cdots, N-1, \\ Q_r(x) &= \sum_{s=0}^{N-1} q_{r,s}x^s,\ r = 0, 1, \cdots, N-1, \\ Y_l(x) &= \sum_{u=0}^{N-1} y_{l,u}x^u,\ l = 0, 1, \cdots, N-1. \end{aligned} \tag{IV.4}$$

The above conversion from 2-D CC to 1-D polynomial CC can be easily verified [13]. Thus, we can perform 2-D CC by evaluating the corresponding 1-D polynomial CC, which, in turn, can be computed using polynomials transforms over Z_p as described in the following.

IV.A Circular Convolution Property of Polynomial Transforms Over Finite Rings

Consider a 1-D polynomial CC as defined in (IV.3) with $x^N - 1$ being substituted by $M(x)$:

$$Y_l(x) \equiv \sum_{m=0}^{N-1} H_m(x) Q_{\langle l-m\rangle_N}(x) \ \operatorname{modd} p, M(x), \tag{IV.5}$$

and assume that $N, g(x), p$ and $M(x)$ form a polynomial transform $(N, g(x), p, M(x))$. Define the corresponding transformed polynomials of $\{Y_i(x)\}, \{H_i(x)\}$ and $\{Q_i(x)\}$ by $\{\overline{Y}_i(x)\}, \{\overline{H}_i(x)\}$ and $\{\overline{Q}_i(x)\}$, respectively. Then,

$$\begin{aligned} \overline{Y}_k(x) &\equiv \sum_{l=0}^{N-1} Y_l(x)[g(x)]^{lk} \ \operatorname{modd} p, M(x) \\ &\equiv \sum_{l=0}^{N-1}\sum_{m=0}^{N-1} H_m(x) Q_{\langle l-m\rangle_N}(x)[g(x)]^{(l-m)k}[g(x)]^{mk} \ \operatorname{modd} p, M(x) \\ &\equiv \sum_{m=0}^{N-1} H_m(x)[g(x)]^{mk} \sum_{l=0}^{N-1} Q_{\langle l-m\rangle_N}(x)[g(x)]^{(l-m)k} \ \operatorname{modd} p, M(x) \\ &\equiv \overline{H}_k(x)\overline{Q}_k(x) \ \operatorname{modd} p, M(x), \end{aligned}$$

where the fact: $[g(x)]^N \equiv 1 \text{ modd } p, M(x)$ from Theorem III.2 (equation (III.4)) is used to recognize the summation $\sum_{l=0}^{N-1} Q_{\langle l-m\rangle_N}(x)[g(x)]^{(l-m)k}$ as $\overline{Q}_k(x)$, for $k = 0, 1, \cdots, N-1$. Thus, in this case, the 1-D polynomial CC in (IV.5) can be computed by evaluating (i) two polynomial transforms for $\{\overline{H}_i(x)\}$ and $\{\overline{Q}_i(x)\}$, (ii) N polynomial products $\{\overline{H}_i(x)\}\{\overline{Q}_i(x)\}$ modd $p, M(x)$, and (iii) one inverse polynomial transform for $\{Y_i(x)\}$ from $\{\overline{Y}_i(x)\}$.

Using the above circular convolution property, we develop two classes of fast algorithms for 2–D integer CC, or equivalently for 1-D polynomial CC, in the next two subsections. The first class is related to MNPT, and the second to FNPT.

IV.B Fast Algorithms for 2-D Circular Convolutions Based on MNPT

For a Mersenne number M_N, we consider a 1-D polynomial CC of length N and modulo $M_N, x^N - 1$ as given in (IV.3):

$$Y_l(x) \equiv \sum_{m=0}^{N-1} H_m(x) Q_{\langle l-m\rangle_N}(x) \text{ modd } M_N, x^N - 1. \tag{IV.6}$$

Because we do not have polynomial transform for this polynomial CC, we decompose it into two 1-D polynomial CCs as follows,

$$\begin{aligned} Y_{1,l}(x) &\equiv \sum_{m=0}^{N-1} H_{1,m}(x) Q_{1,\langle l-m\rangle_N}(x) \equiv Y_l(x) \text{ modd } M_N, \frac{x^N-1}{x-1}, \\ Y_{2,l} &\equiv \sum_{m=0}^{N-1} H_{2,m} Q_{2,\langle l-m\rangle_N} \equiv Y_l(x) \text{ modd } M_N, (x-1),\ l = 0, 1, \cdots, N-1, \end{aligned}$$

where

$$\begin{aligned} &H_{1,m}(x) \equiv H_m(x) \text{ modd } M_N, \frac{x^N-1}{x-1};\ Q_{1,r}(x) \equiv Q_r(x) \text{ modd } M_N, \frac{x^N-1}{x-1}; \\ &H_{2,m} \equiv H_m(x) \text{ modd } M_N, (x-1);\ Q_{2,r} \equiv Q_r(x) \text{ modd } M_N, (x-1), \end{aligned}$$

for $m, r = 0, 1, \ldots, N-1$. Note that $\{Y_{2,l}\}$ is simply a 1-D integer CC of length N which can be computed using Mersenne Number Transform (MNT) (see [4]). The computation of $\{Y_{1,l}(x)\}$ can be done by using MNPT $(N, x, M_N, (x^N-1)/(x-1))$ as follows,

$$\begin{aligned}
\overline{Q}_k(x) &\equiv \sum_{r=0}^{N-1} Q_{1,r}(x)x^{rk} \ \mathrm{modd}\, M_N, \frac{x^N-1}{x-1}, \\
\overline{H}_k(x) &\equiv \sum_{m=0}^{N-1} H_{1,m}(x)x^{mk} \ \mathrm{modd}\, M_N, \frac{x^N-1}{x-1}, \\
\overline{Y}_k(x) &\equiv \overline{Q}_k(x)\overline{H}_k(x) \ \mathrm{modd}\, M_N, \frac{x^N-1}{x-1}, \ k = 0, 1, \cdots, N-1, \\
Y_{1,l}(x) &\equiv N^{-1} \sum_{k=0}^{N-1} \overline{Y}_k(x)x^{-lk} \ \mathrm{modd}\, M_N, \frac{x^N-1}{x-1}, \ l = 0, 1, \cdots, N-1.
\end{aligned} \tag{IV.7}$$

We recover $\{Y_l(x)\}$ using Theorem II.1 and Lemma C.2 from $\{Y_{1,l}(x)\}$ and $\{Y_{2,l}\}$:

$$Y_l(x) \equiv Y_{1,l}(x)\left(N - \frac{x^N-1}{x-1}\right)N^{-1} + Y_{2,l}\frac{x^N-1}{x-1}N^{-1} \ \mathrm{modd}\, M_N, x^N-1, \tag{IV.8}$$

for $l = 0, 1, \ldots, N-1$. The above is the basic structure for the computation of (IV.6) and is summarized in Figure 1. The algorithms for the above each step are developed in the following.

IV.B.1 Algorithm for product of two polynomials modulo $M_N, \left(x^N-1\right)/(x-1)$

As in (IV.7), we need to compute products of two polynomials modulo $M_N, (x^N - 1)/(x-1)$. We rewrite one of them as:

$$Y(x) \equiv H(x)Q(x) \ \mathrm{modd}\, M_N, \frac{x^N-1}{x-1}, \tag{IV.9}$$

where $H(x) = \sum_{m=0}^{N-2} h_m x^m$, $Q(x) = \sum_{n=0}^{N-2} q_n x^n$, and $Y(x) = \sum_{l=0}^{N-2} y_l x^l$. Based on Lemma C.3, the following equation can be easily verified

$$\frac{x^N-1}{x-1} \equiv (x-2)\left(x-2^2\right)\cdots\left(x-2^{N-1}\right) \ \mathrm{mod}\, M_N.$$

Therefore, to compute (IV.9), we first compute

$$Y_k \equiv H_kQ_k \equiv Y(x) \ \mathrm{modd}\, M_N, x-2^k,$$

for $k = 0, 1, \ldots, N-1$, where $H_k \equiv \sum_{m=0}^{N-2} h_m 2^{mk}$, $Q_k \equiv \sum_{n=0}^{N-2} q_n 2^{nk} \ \mathrm{mod}\, M_N$. Then we get $Y(x)$ from $\{Y_k\}$ using Theorem II.1:

$$Y(x) \equiv \sum_{k=1}^{N-1} Y_k \frac{x^N-1}{(x-1)(x-2^k)}\left[\frac{x^N-1}{(x-1)(x-2^k)} \ \mathrm{modd}\, M_N, x-2^k\right]^{-1} \ \mathrm{modd}\, M_N, \frac{x^N-1}{x-1} \tag{IV.10}$$

where $(x^N-1)/((x-1)(x-2^k))$ and its inverse are given below.

Let $(x^N-1)/((x-1)(x-2^k)) \equiv a_{N-2}x^{N-2}+a_{N-3}x^{N-3}+\ldots+a_1x+a_0 \bmod M_N$. Then, from the equation

$$\begin{aligned}&\left(x^N-1\right)/(x-1) \equiv x^{N-1}+x^{N-2}+\cdots+x+1 \equiv\\&\left(a_{N-2}x^{N-2}+a_{N-3}x^{N-3}+\cdots+a_1x+a_0\right)\left(x-2^k\right) \equiv\\&a_{N-2}x^{N-1}+a_{N-3}x^{N-2}+\cdots+a_1x^2+a_0x-\\&a_{N-2}2^kx^{N-2}-a_{N-3}2^kx^{N-3}-\cdots-a_12^kx-a_02^k, \bmod M_N,\end{aligned}$$

and by equating the corresponding coefficients from both sides, we have

$$\begin{gathered}a_{N-2}\equiv 1\\a_{N-3}-2^ka_{N-2}\equiv 1,\ a_{N-3}\equiv 1+2^k\\a_{N-4}-2^ka_{N-3}\equiv 1,\ a_{N-4}\equiv 1+2^k+2^{2k}\\\cdots\cdots\\a_1\equiv 1+2^k+\cdots+2^{(N-3)k}\\a_0\equiv 1+2^k+\cdots+2^{(N-3)k}+2^{(N-2)k}\\-2^ka_0\equiv 1 \bmod M_N,\end{gathered}\tag{IV.11}$$

where the last two equations are consistent. Indeed,

$$-2^ka_0 \equiv -\left(2^k+2^{2k}+\cdots+2^{(N-1)k}\right) \equiv 1-\frac{2^{Nk}-1}{2^k-1} \equiv 1 \bmod M_N.$$

In the above, we need the fact: $(2^{Nk}-1)/(2^k-1)\equiv 0 \bmod M_N$, for $k=1,2,..,N-1$, which can be proved as follows. Since $(x^N-1,x^k-1)\equiv x-1$, $x^{Nk}-1$ has factors x^k-1 and x^N-1, thus $(x^N-1)/(x-1) \mid (x^{Nk}-1)/(x^k-1)$, or $M_N=(2^N-1)/(2-1) \mid (2^{Nk}-1)/(2^k-1)$. Now, using (IV.11), we have

$$\frac{x^N-1}{(x-1)(x-2^k)} \equiv x^{N-2}+\left(1+2^k\right)x^{N-3}+\cdots+\left(1+2^k+\cdots+2^{(N-2)k}\right)$$
$$\operatorname{modd} M_p, \frac{x^N-1}{x-1},\ k=1,2,\cdots,N-1,$$

and

$$\begin{aligned}&\left[\frac{x^N-1}{(x-1)(x-2^k)} \operatorname{modd} M_N, x-2^k\right]^{-1} \equiv\\&\left[(N-1)2^{(N-2)k}+(N-2)2^{(N-3)^{-1}k}+\cdots+2\times 2^k+1\right] \equiv\\&N^{-1}2^k\left(2^k-1\right) \bmod M_N,\end{aligned}$$

for $k = 1, 2, \ldots, N-1$. In the practical cases, one data array, say $H_m(x)$ (ref. (IV.6)), in 2–D integer CC is corresponding to the unit sample response of the filter to be implemented and thus is fixed, we can combine the above $N-1$ inverse numbers into $\{H_k\}$. Then (IV.10) becomes

$$\begin{aligned}
Y(x) &\equiv \sum_{k=1}^{N-1} Y_k \frac{x^N - 1}{(x-1)(x-2^k)} \ \operatorname{modd} M_N, \frac{x^N-1}{x-1} \\
&\equiv Y_1\Big[x^{N-2} + (1+2)x^{N-3} + \cdots + \Big(1 + 2 + \cdots + 2^{N-2}\Big)\Big] \\
&+ Y_2\Big[x^{N-2} + \Big(1+2^2\Big)x^{N-3} + \cdots + \Big(1 + 2^2 + \cdots + 2^{2(N-2)}\Big)\Big] \\
&\cdots\cdots \\
&+ Y_{N-1}\Big[x^{N-2} + \Big(1+2^{N-1}\Big)x^{N-3} + \cdots + \Big(1 + 2^{N-1} + \cdots + 2^{(N-1)(N-2)}\Big)\Big].
\end{aligned}$$

We define

$$\begin{aligned}
\overline{Y}_1 &\equiv \sum_{k=1}^{N-1} Y_k \ \operatorname{mod} M_N \\
\overline{Y}_2 &\equiv \sum_{k=1}^{N-1} 2^k Y_k \ \operatorname{mod} M_N \\
&\cdots\cdots \\
\overline{Y}_{N-1} &\equiv \sum_{k=1}^{N-1} 2^{(N-2)k} Y_k \ \operatorname{mod} M_N,
\end{aligned}$$

then

$$Y(x) \equiv \overline{Y}_1 x^{N-2} + (\overline{Y}_1 + \overline{Y}_2)x^{N-3} + \cdots + (\overline{Y}_1 + \overline{Y}_2 + \cdots + \overline{Y}_{N-1}).$$

The number of multiplications (M), additions (A), and shifts (S), i.e., multiplying by 2^i for some $i > 0$, for computing a polynomial product using the above algorithm are

$$\begin{aligned}
M &= N - 1 \\
A &= 2(N-1)(N-2) + (N-2) \\
S &= 2(N-1)(N-2).
\end{aligned}$$

IV.B.2 Algorithm for Computation of MNPT

The MNPT of $\{Q_{1,r}(x)\}$ (see (IV.7)) can be written as

$$\overline{Q}_k(x) \equiv Q_{1,0}(x) + \sum_{r=1}^{N-1} Q_{1,r}(x) x^{kr} \ \operatorname{modd} M_N, \frac{x^N-1}{x-1},$$

for $k = 0, 1, \ldots, N-1$, where the degrees of $Q_{1,r}(x)$ are no more than $N-2$, $r = 0, 1, ..., N-1$. Define the second term of the above equation as $R_k(x)$. Then, the computation of $R_0(x)$ needs $(N-2)(N-1)$ additions.

For $k \neq 0, N-1$, we first compute $R_k(x)$ modulo $M_N, x^N - 1$, then modulo $M_N, (x^N - 1)/(x-1)$. The computation needs $N^2 - 2N$ additions.

When $r \not\equiv 0 \bmod N$, $\sum_{k=0}^{N-2} x^{rk} \equiv -x^{r(N-1)}$, modd $M_N, \left(x^N - 1\right)/(x-1)$, thus

$$R_{N-1}(x) \equiv -\sum_{k=0}^{N-2}\sum_{r=1}^{N-1} Q_{1,r}(x)x^{rk} \equiv -\sum_{k=0}^{N-2} R_k(x) \ \text{modd}\, M_N, \frac{x^N-1}{x-1},$$

which requires $(N-1)(N-2)$ additions.

To get

$$\overline{Q}_k(x) \equiv Q_{1,0}(x) + R_k(x), \ \text{modd}\, M_N, \frac{x^N-1}{x-1}, \ k = 0, 1, \cdots, N-1,$$

we need to do $N(N-1)$ additions. And the total computation for $\{\overline{Q}_k(x)\}$ is $N^3 - N^2 - 3N + 4$ additions.

Similarly for the inverse MNPT (see (IV.7)), we have $\sum_{k=0}^{N-2} x^{-rk} \equiv -x^{-r(N-1)}$ modd $M_N, (x^N - 1)/(x-1)$, for $r \not\equiv 0 \bmod N$. Combining the factor N^{-1} into $\overline{H}_k(x)$, the computation of inverse MNPT is also $N^3 - N^2 - 3N + 4$ additions.

Now consider the reconstruction of $Y_l(x)$ using Theorem II.1 (see (IV.8)). Define

$$T(x) \equiv -x^{N-2} - 2x^{N-3} \cdots - (N-3)x^2 - (N-2)x - (N-1) \ \text{modd}\, M_N, \frac{x^N-1}{x-1}.$$

It can be verified that

$$(x-1)T(x)N^{-1} \equiv \begin{cases} 0 & \text{modd}\, M_N, x-1 \\ 1 & \text{modd}\, M_N, \left(x^N-1\right)/(x-1) \end{cases},$$

$$\frac{x^N-1}{x-1}N^{-1} \equiv \begin{cases} 1 & \text{modd}\, M_N, x-1 \\ 0 & \text{modd}\, M_N, \left(x^N-1\right)/(x-1) \end{cases}.$$

Thus

$$Y_l(x) \equiv Y_{1,l}(x)(x-1)T(x)N^{-1} + Y_{2,l}\frac{x^N-1}{x-1}N^{-1} \ \text{modd}\, M_N, x^N - 1,$$

and, after combining $T(x)N^{-1}$ into $\{\overline{H}_k(x)\}$, N^{-1} into $\{H_{2,m}\}$,

$$Y_l(x) \equiv Y_{1,l}(x)(x-1) + Y_{2,l}\frac{x^N-1}{x-1} \ \text{modd}\, M_N, x^N - 1, \ l = 0, 1, \cdots, N-1.$$

And this reconstruction requires $2N(N-1)$ additions.

IV.B.3 Example

In this subsection, we provide an example of computing 2-D integer CC with the above fast algorithm based on MNPT. Referring (IV.1), the 2-D data array are given in the following,

$$\{h_{m,n}\}: \begin{array}{ccc} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{array} \qquad \{q_{r,s}\}: \begin{array}{ccc} 0 & 1 & -1 \\ 1 & -1 & 0 \\ -1 & 0 & 1 \end{array},$$

and we want to compute

$$y_{\ell,u} = \sum_{m=0}^{2} \sum_{n=0}^{2} h_{m,n} q_{\langle \ell - m\rangle_3, \langle u-n\rangle_3},$$

for $\ell, u = 0, 1, 2$. We choose $N = 3$ and $M_3 = 7$ for the modulo arithmetic, which satisfy the requirement (IV.2). Now we use MNPT to compute $y_{\ell,u}$ mod 7 as follows, where the notations are based on the ones used before and, thus, their definitions can be located in the above description of the algorithm.

$$N^{-1}H_{2,m}: \begin{array}{ccc} 2 & 5 & 5 \end{array} \qquad Q_{2,r}: \begin{array}{ccc} 0 & 0 & 0 \end{array}$$

$$H_{1,m}(x): \begin{array}{cc} 0 & 1 \\ -1 & 0 \\ -1 & -1 \end{array} \qquad Q_{1,r}: \begin{array}{cc} 1 & 2 \\ 1 & -1 \\ -2 & -1 \end{array}$$

$$\overline{Q}_k(x): \begin{array}{cc} 0 & 0 \\ 3 & -1 \\ 0 & 0 \end{array} \qquad \overline{H}_k(x): \begin{array}{cc} -2 & 0 \\ 0 & 1 \\ 2 & 2 \end{array}$$

$$T(x)N^{-1} \equiv (-x-2)5 \equiv 2(x+2) \bmod 7$$

$$\left(N^{-1}\right)^2 T(x)\overline{H}_k(x): \begin{array}{cc} 2 & 1 \\ 4 & 3 \\ -1 & 5 \end{array}$$

To compute $\overline{Y}_k(x) \equiv \overline{Q}_k(x)(N^{-1})^2 T(x)\overline{H}_k(x)$ modd $7, x^2+x+1$, for $k = 0, 1, 2$, we have

$$\frac{x^3-1}{(x-1)(x-2)} \equiv x+1+2 \text{ modd } 7, x^2+x+1,$$

$$\frac{x^3-1}{(x-1)(x-4)} \equiv x+1+2^2 \text{ modd } 7, x^2+x+1,$$

$$N^{-1}2(2-1) \equiv 3 \bmod 7; \; N^{-1}4(4-1) \equiv 4 \bmod 7.$$

Referring to Subsection *IV.B.1*, for $k = 0$ and $k = 2$, we have $\overline{Y}_k(x) \equiv 0$ modd $7, x^2 + x + 1$, since the corresponding $Q_1 \equiv 0$, $Q_2 \equiv 0$ all modd $7, x^2 + x + 1$. For $k = 1$, $\overline{Y}_1(x) \equiv 2(x+1+2) + (-1)(x+1+2^2) \equiv x+1$ modd $7, x^2+x+1$, since $Q_1 \equiv 1$, $Q_2 \equiv -1$, $3H_1 \equiv 2$, $4H_2 \equiv 1$, and $Y_1 \equiv 2$, $Y_2 \equiv -1$. Then using the inverse MNPT on $\overline{Y}_k(x)$, we get

$$Y_{1,\ell}(x): \begin{array}{cc} 1 & 1 \\ 0 & -1 \\ -1 & 0 \end{array}.$$

Obviously, we also have $Y_{2,\ell}$: 0 0 0. Applying Theorem II.1,

$$Y_\ell(x) \equiv Y_{1,\ell}(x)(x+1) + Y_{2,\ell}(x^2+x+1) \text{ modd } 7, x^3 - 1,$$

for $\ell = 0, 1, 2$, from which we obtain

$$\{y_{\ell,u}\}: \begin{array}{ccc} -1 & 0 & 1 \\ 0 & 1 & -1 \\ 1 & -1 & 0 \end{array}.$$

IV.C Fast Algorithms for 2-D Circular Convolutions Based on FNPT

For a Fermat number F_t, we consider a 1-D polynomial CC (ref. (IV.3)) of length $N = 2^{t+1}$ and modulo $F_t, x^N - 1$:

$$Y_l(x) \equiv \sum_{m=0}^{N-1} H_m(x) Q_{\langle l-m \rangle_N}(x) \text{ modd } F_t, x^N - 1.$$

Define

$$\begin{aligned} Y_{1,l}(x) &\equiv \sum_{m=0}^{N-1} H_{1,m}(x) Q_{1,\langle l-m \rangle_N}(x) \equiv Y_l(x) \text{ modd } F_t, x^{2^t} + 1, \\ Y_{2,l}(x) &\equiv \sum_{m=0}^{N-1} H_{2,m}(x) Q_{2,\langle l-m \rangle_N}(x) \equiv Y_l(x) \text{ modd } F_t, x^{2^t} - 1, \\ &\qquad l = 0, 1, \cdots, N-1, \end{aligned} \tag{IV.12}$$

where

$$H_{1,m}(x) \equiv H_m(x) \text{ modd } F_t, x^{2^t} + 1,\ Q_{1,r}(x) \equiv Q_r(x) \text{ modd } F_t, x^{2^t} + 1,$$
$$H_{2,m}(x) \equiv H_m(x) \text{ modd } F_t, x^{2^t} - 1,\ Q_{2,r}(x) \equiv Q_r(x) \text{ modd } F_t, x^{2^t} - 1,$$
$$r, m = 0, 1, \cdots, N-1.$$

$\{Y_l(x)\}$ can be computed from $\{Y_{1,l}(x)\}$ and $\{Y_{2,l}(x)\}$ by Theorem II.1:

$$Y_l(x) \equiv Y_{1,l}(x)2^{2^t-1}\left(x^{2^t}-1\right) + Y_{2,l}(x)2^{2^{t+1}-1}\left(x^{2^t}+1\right) \operatorname{modd} F_t, x^{2^{t+1}}-1,$$
$$l = 0, 1, \cdots, N-1.$$

We use FNPT $(2^{t+1}, x, F_t, x^{2^t}+1)$ to compute $\{Y_{1,l}\}$:

$$\begin{aligned}
\overline{Q}_k(x) &\equiv \sum_{r=0}^{N-1} Q_{1,r}(x)x^{rk} \operatorname{modd} F_t, x^{2^t}+1,\\
\overline{H}_k(x) &\equiv \sum_{m=0}^{N-1} H_{1,m}(x)x^{mk} \operatorname{modd} F_t, x^{2^t}+1,\\
\overline{Y}_k(x) &\equiv \overline{Q}_k(x)\overline{H}_k(x) \operatorname{modd} F_t, x^{2^t}+1,\ k=0,1,\cdots,N-1,\\
Y_{1,l}(x) &\equiv N^{-1}\sum_{k=0}^{N-1} \overline{Y}_k(x)x^{-lk} \operatorname{modd} F_t, x^{2^t}+1,\ l=0,1,\cdots,N-1,
\end{aligned} \tag{IV.13}$$

where $N^{-1} \equiv (2^{t+1})^{-1} \equiv 2^{2^{t+1}-(t+1)} \bmod F_t$. For $\{Y_{2,l}\}$, we recognize that $\{Y_{2,l}\}_{l=0}^{N-1}$ corresponds to an $N \times (N/2)$ 2–D integer array with each row expressed in a polynomial form, and is the result of an $N \times (N/2)$ 2-D integer CC (ref. (IV.12)); this $N \times (N/2)$ 2-D integer CC can be also expressed as an $(N/2) \times N$ 2-D integer CC and, thus can be written as a 1–D polynomial (of degree $N-1$) CC of length $N/2$, denoted as follows,

$$Y_l'(x) \equiv \sum_{m=0}^{2^t-1} H_m'(x)Q_{\langle l-m\rangle_{2^t}}'(x) \operatorname{modd} F_t, x^{2^{t+1}}-1,$$
$$l = 0, 1, \cdots, 2^t-1,$$

where the data array corresponding to $\{Y'_l(x)\}$, $\{H'_m(x)\}$ and $\{Q'_i(x)\}$ are just the transposes of those for $\{Y_{2,l}(x)\}$, $\{H_{2,m}(x)\}$ and $\{Q_{2,i}(x)\}$, respectively. Again, define

$$Y_{1,l}'(x) \equiv \sum_{m=0}^{2^t-1} H_{1,m}'(x)Q_{1,\langle l-m\rangle_{2^t}}'(x) \equiv Y_l'(x) \operatorname{modd} F_t, x^{2^t}+1,$$
$$Y_{2,l}'(x) \equiv \sum_{m=0}^{2^t-1} H_{2,m}'(x)Q_{2,\langle l-m\rangle_{2^t}}'(x) \equiv Y_l'(x) \operatorname{modd} F_t, x^{2^t}-1,$$
$$l = 0, 1, \cdots, 2^t-1.$$

$\{Y'_{1,l}(x)\}$ can be computed by FNPT $(x^2, 2^t, F_t, x^{2^t}+1)$:

$$\begin{aligned}
\overline{Q'_k}(x) &\equiv \sum_{r=0}^{2^t-1} Q'_{1,r}(x)x^{2rk} \ \operatorname{modd} F_t, x^{2^t}+1, \\
\overline{H'_k}(x) &\equiv \sum_{m=0}^{2^t-1} H'_{1,m}(x)x^{2mk} \ \operatorname{modd} F_t, x^{2^t}+1, \\
\overline{Y'_k}(x) &\equiv \overline{Q'_k}(x)\overline{H'_k}(x) \ \operatorname{modd} F_t, x^{2^t}+1,\ k = 0, 1, \cdots, 2^t-1, \\
Y'_{1,l}(x) &\equiv \left(2^t\right)^{-1} \sum_{k=0}^{2^t-1} \overline{Y'_k}(x)x^{-2lk} \ \operatorname{modd} F_t, x^{2^t}+1,\ l = 0, 1, \cdots, 2^t-1.
\end{aligned} \tag{IV.14}$$

$\{Y'_{2,l}(x)\}$ corresponds a $2^t \times 2^t$ 2-D integer CC, and, thus, has a similar computation process as described above. The computation process is shown in Figure 2; the algorithm for each step are developed in the following.

IV.C.1 Algorithm for product of two polynomials modulo $F_{t+i}, x^{2^t}+1$

In (IV.13) and (IV.14), we need to compute products of two polynomials modulo $F_{t+i}, x^{2^t}+1$, for $i \geq 0$. We write one of them in the following form for convenience:

$$Y(x) \equiv H(x)Q(x) \ \operatorname{modd} F_{t+i}, x^{2^t}+1,\ i \geq 0,$$

where $H(x) = \sum_{m=0}^{2^t-1} h_m x^m$; $Q(x) = \sum_{n=0}^{2^t-1} q_n x^n$; $Y(x) = \sum_{l=0}^{2^t-1} y_l x^l$. Define

$$H'(x) = \sum_{m=0}^{2^t-1} h_m x^m 2^{2^i m};\ Q'(x) = \sum_{n=0}^{2^t-1} q_n x^n 2^{2^i n};\ Y'(x) = \sum_{l=0}^{2^t-1} y_l x^l 2^{2^i l}.$$

Since $(2^{2^i})^{2^t} \equiv -1 \bmod F_{t+i}$, we have $Y'(x) \equiv H'(x)Q'(x)$, $\operatorname{modd} F_{t+i}, x^{2^t}-1$, which can be computed using FNT with kernel $2^{2^{i+1}}$, length 2^t, and modulus F_{t+i} [4]. Define the FNT as

$$\begin{aligned}
\overline{H}_k &\equiv \sum_{n=0}^{2^t-1} h_n \left(2^{2^i}\right)^n \left(2^{2^{i+1}}\right)^{nk} \equiv \sum_{n=0}^{2^t-1} h_n \left(2^{2^i}\right)^{n(2k+1)} \ \bmod F_{i+i}, \\
\overline{Q}_k &\equiv \sum_{s=0}^{2^t-1} q_s \left(2^{2^i}\right)^{s(2k+1)} \ \bmod F_{i+i},\ k = 0, 1, \cdots, 2^t-1,
\end{aligned} \tag{IV.15}$$

which can be rewritten as

$$\overline{Q}_k \equiv \sum_{s=0}^{2^{t-1}-1} q_{2s}\left(2^{2^i}\right)^{2s(2k+1)} + \left(2^{2^i}\right)^{2k+1} \sum_{s=0}^{2^{t-1}-1} q_{2s+1}\left(2^{2^i}\right)^{2s(2k+1)} \mod F_{t+i},$$

$$\overline{Q}_{k+2^{t-1}} \equiv \sum_{s=0}^{2^{t-1}-1} q_{2s}\left(2^{2^i}\right)^{2s(2k+1)} - \left(2^{2^i}\right)^{2k+1} \sum_{s=0}^{2^{t-1}-1} q_{2s+1}\left(2^{2^i}\right)^{2s(2k+1)} \mod F_{t+i},$$

$$k = 0, 1, \cdots, 2^{t-1} - 1.$$

Thus $\{\overline{Q}_k\}$ can be computed recursively with $t2^t$ additions and $t2^{t-1}$ shifts.

Then we compute $\overline{Y}_k \equiv \overline{H}_k\overline{Q}_k \mod F_{t+i}$, $k = 0, 1, \cdots, 2^t - 1$, and

$$y_u\left(2^{2^i}\right)^u \equiv \left(2^t\right)^{-1} \sum_{k=0}^{2^t-1} \overline{Y}_k\left(2^{2^{i+1}}\right)^{-ku} \mod F_{t+i},\ u = 0, 1, \cdots, 2^t - 1.$$

If we combine $(2^t)^{-1}$ into $\{\overline{H}_k\}$, the above equation can be written as

$$y_u \equiv \sum_{k=0}^{2^t-1} \overline{Y}_k\left(2^{2^i}\right)^{-u(2k+1)} \mod F_{t+i},\ u = 0, 1, \cdots, 2^t - 1, \qquad \text{(IV.16)}$$

which can be decomposed into

$$y_{2u} \equiv \sum_{k=0}^{2^{t-1}-1} \left(\overline{Y}_k + \overline{Y}_{k+2^{t-1}}\right)\left(2^{2^i}\right)^{-2u(2k+1)} \mod F_{t+i},$$

$$y_{2u+1} \equiv \sum_{k=0}^{2^{t-1}-1} \left(\overline{Y}_k - \overline{Y}_{k+2^{t-1}}\right)\left(2^{2^i}\right)^{-(2k+1)}\left(2^{2^i}\right)^{-2u(2k+1)} \mod F_{t+i},$$

$$u = 0, 1, \cdots, 2^{t-1} - 1.$$

To compute $\overline{Y}_k + \overline{Y}_{k+2^{t-1}}$ and $(\overline{Y}_k - \overline{Y}_{k+2^{t-1}})(2^{2^i})^{-(2k+1)}$, for $k = 0, 1, ..., 2^{t-1} - 1$, we need to do 2^t additions and 2^{t-1} shifts. The above process can be done again in the 2nd stage for y_{2u} and y_{2u+1}, and the additional computations are also 2^t additions and 2^{t-1} shifts. There are t stages. Thus the total computation is $t2^t$ additions and $t2^{t-1}$ shifts.

IV.C.2 Algorithm for Computation of FNPT

For $(2^{t-i+1}, x, F_t, x^{2^{t-i}} + 1)$, $i = 0, 1, ..., t$, the FNPT of $\{\overline{Q}_r(x)\}$ is

$$\overline{Q}_k(x) \equiv \sum_{r=0}^{2^{t-i+1}-1} Q_r(x)x^{rk} \operatorname{modd} F_t, x^{2^{t-i}} + 1,\ k = 0, 1, \cdots, 2^{t-i+1} - 1,$$

which can be decomposed into the following two cases

$$\overline{Q}_k(x) \equiv \sum_{r=0}^{2^{t-i}-1} Q_{2r}(x)x^{2rk} + x^k \sum_{r=0}^{2^{t-i}-1} Q_{2r+1}(x)x^{2rk} \ \mathrm{modd}\, F_t, x^{2^{t-i}}+1,$$
$$\overline{Q}_{k+2^{t-i}}(x) \equiv \sum_{r=0}^{2^{t-i}-1} Q_{2r}(x)x^{2rk} - x^k \sum_{r=0}^{2^{t-i}-1} Q_{2r+1}(x)x^{2rk} \ \mathrm{modd}\, F_t, x^{2^{t-i}}+1, \qquad \text{(IV.17)}$$
$$k = 0, 1, \cdots, 2^{t-i}-1,$$

where we have used the fact $x^{2^{t-i+1}} \equiv 1, x^{2^{t-i}} \equiv -1$ modd $F_t, x^{2^{t-i}}+1$. (IV.17) is the first stage of decoding for the computation of $\{\overline{Q}_r(x)\}$. Similarly, we can perform further decomposition on the summations in (IV.17): $\sum_{r=0}^{2^{t-i}-1} Q_{2r}(x)x^{2rk}$, $\sum_{r=0}^{2^{t-i}-1} Q_{2r+1}(x)x^{2rk}$. For each stage, we need $2 \times 2^{2(t-i)}$ additions. Since there are $t-i+1$ stages, the total computation is $(t-i+1) \times 2 \times 2^{2(t-i)}$ additions.

For $(2^{t-i}, x^2, F_t, x^{2^{t-i}}+1)$, $i = 0, 1, ..., t-1$, the FNPT of $\{\overline{Q}_r(x)\}$ is

$$\overline{Q}_k(x) \equiv \sum_{r=0}^{2^{t-i}-1} Q_r(x)x^{2rk}, \ \mathrm{modd}\, F_t, x^{2^{t-i}}+1,\ k = 0, 1, \cdots, 2^{t-i}-1,$$

which is similar to the 1st term of the right-hand-side of (IV.17), and, thus, it can be computed in the same way. The total computation is $(t-i)2^{2(t-i)}$ additions.

V Fast Algorithms for 2-D Number Theoretic Transforms

In this section, we develope fast algorithms for direct computation of 2-D MNTs and 2-D Fermat Number Transforms (FNT) [4] [8] by using MNPTs and FNPTs. The traditional way for computing 2-D NTTs of a 2-D data array is to apply the corresponding 1-D NTTs to each row and then to each column (or vice versa) of the 2-D data array. The computational complexity comparisons of the new technique presented in this section and the traditional row-column scheme is given in the next section.

V.A Fast Algorithm of 2-D Mersenne Number Transforms Using MNPT

Consider a 2–D MNT of size $N \times N$, where N is a prime number,

$$\overline{Q}_{k_1,k_2} \equiv \sum_{n_1=0}^{N-1} \sum_{n_2=0}^{N-1} q_{n_1,n_2} 2^{k_1 n_1} 2^{k_2 n_2} \ \mathrm{mod}\, M_N,\ k_1, k_2 = 0, 1, \cdots, N-1.$$

Define

$$Q_{n_1}(x) \equiv \sum_{n_2=0}^{N-1} q_{n_1,n_2} x^{n_2} \bmod M_N,\ n_1 = 0, 1, \cdots, N-1,$$
$$\overline{Q}_{k_1}(x) \equiv \sum_{n_1=0}^{N-1} Q_{n_1}(x) 2^{n_1 k_1} \ \text{modd}\, M_N, x^N - 1,\ k_1 = 0, 1, \cdots, N-1,$$

then

$$\overline{Q}_{k_1,k_2} \equiv \overline{Q}_{k_1}(x) \ \text{modd}\, M_N, x - 2^{k_2},\ k_2 = 0, 1, \cdots, N-1.$$

When $k_2 = 0$,

$$\overline{Q}_{k_1,0} \equiv \sum_{n_1=0}^{N-1} (Q_{n_1}(x) \ \text{modd}\, M_N, x-1) 2^{n_1 k_1} \bmod M_N,\ k_1 = 0, 1, \cdots, N-1,$$

which is a 1–D MNT.

When $k_2 \neq 0$, define

$$Q'_{n_1}(x) \equiv Q_{n_1}(x) \ \text{modd}\, M_N, \frac{x^N - 1}{x - 1},\ n_1 = 0, 1, \cdots, N-1,$$
$$\overline{Q'}_{k_1}(x) \equiv \sum_{n_1=0}^{N-1} Q_{n_1}(x) 2^{n_1 k_1} \ \text{modd}\, M_N, \frac{x^N - 1}{x - 1},\ k_1 = 0, 1, \cdots, N-1,$$

then

$$\overline{Q}_{k_1,k_2} \equiv \overline{Q'}_{k_1}(x) \ \text{modd}\, M_N, x - 2^{k_2},\ k_2 = 1, 2, \cdots, N-1.$$

Since $\{\langle k_2 k_1 \rangle_N\}$ is a permutation of $\{k_1\}$ when $k_2 \neq 0$,

$$\overline{Q'}_{\langle k_1 k_2 \rangle_N}(x) \equiv \sum_{n_1=0}^{N-1} Q'_{n_1}(x) 2^{n_1 k_1 k_2} \equiv \sum_{n_1=0}^{N-1} Q'_{n_1}(x) x^{n_1 k_1} \ \text{modd}\, M_N, \frac{x^N - 1}{x - 1},$$
$$k_1 = 0, 1, \cdots, N-1,$$

which is a MNPT.[3]

[3] The computation of $\overline{Q}_{\langle k_1 k_2 \rangle_N, k_2} \equiv \overline{Q'}_{\langle k_1 k_2 \rangle_N}(x) \ \text{modd}\, M_N, x - 2^{k_2}$, $k_2 = 1, 2, \cdots, N-1$, is p reconstruction processes of (IV.10).

V.B Computation of 2–D Fermat Number Transforms Using FNPT

Consider a 2–D FNT of size $2^{t+1} \times 2^{t+1}$

$$\overline{Q}_{k_1,k_2} \equiv \sum_{n_1=0}^{2^{t+1}-1} \sum_{n_2=0}^{2^{t+1}-1} q_{n_1,n_2} 2^{k_1 n_1} 2^{k_2 n_2} \mod F_t,\ k_1, k_2 = 0, 1, \cdots, 2^{t+1} - 1.$$

Define

$$Q_{n_1}(x) = \sum_{n_2=0}^{2^{t+1}-1} q_{n_1,n_2} x^{n_2},\ n_1 = 0, 1, \cdots, 2^{t+1} - 1,$$

$$\overline{Q}_{k_1}(x) \equiv \sum_{n_2=0}^{2^{t+1}-1} Q_{n_1}(x) 2^{k_1 n_1} \ \mathrm{modd}\, F_t, x^{2^{t+1}} - 1,\ k_1 = 0, 1, \cdots, 2^{t+1} - 1,$$

then

$$\overline{Q}_{k_1,k_2} \equiv \overline{Q}_{k_1}(x) \ \mathrm{modd}\, F_t, x - 2^{k_2},\ k_1, k_2 = 0, 1, \cdots, 2^{t+1}.$$

When $k_2 = 2u + 1, u = 0, 1, ..., 2^t - 1$

$$Q'_{n_1}(x) \equiv Q_{n_1}(x) \ \mathrm{modd}\, F_t, x^{2^t} + 1,\ n_1 = 0, 1, \cdots, 2^{t+1} - 1,$$

$$\overline{Q'}_{<k_1 k_2>_{2^{t+1}}}(x) \equiv \sum_{n_1=0}^{2^{t+1}-1} Q'_{n_1}(x) 2^{k_1 k_2 n_1} \equiv \sum_{n_1=0}^{2^{t+1}-1} Q'_{n_1}(x) x^{k_1 n_1} \ \mathrm{modd}\, F_t, x^{2^t} + 1,$$
$$k_1 = 0, 1, \cdots, 2^{t+1} - 1.$$

When $k_2 = 2u, u = 0, 1, ..., 2^t - 1$

$$\overline{Q'}_{k,2u}(x) \equiv \sum_{n_1=0}^{2^{t+1}-1} \sum_{n_2=0}^{2^t-1} \left(q_{n_1,n_2} + q_{n_1,n_2+2^t}\right) 2^{k_1 n_1} 2^{2u n_2} \mod F_t,$$
$$u, k_1 = 0, 1, \cdots, 2^{t+1} - 1.$$

Define

$$Q_{n_2}(x) = \sum_{n_1=0}^{2^{t+1}-1} \left(q_{n_1,n_2} + q_{n_1,n_2+2^t}\right) x^{n_1},\ n_2 = 0, 1, \cdots, 2^t - 1,$$

$$\overline{Q}_{2u}(x) \equiv \sum_{n_2=0}^{2^t-1} Q_{n_2}(x) 2^{2u n_2} \ \mathrm{modd}\, F_t, x^{2^{t+1}} - 1,\ u = 0, 1, \cdots, 2^t - 1.$$

Then

$$\overline{Q}_{k_1,2u} \equiv \overline{Q}_{2u}(x) \ \mathrm{modd}\, F_t, x - 2^{k_1},\ k_1 = 0, 1, \cdots, 2^{t+1} - 1.$$

In this case, for $k_1 = 2v+1, v = 0, 1, .., 2^t - 1$, we define

$$Q'_{n_2}(x) \equiv Q_{n_2}(x) \ \text{modd}\, F_t, x^{2^t}+1,\ n_2 = 0, 1, \cdots, 2^t - 1,$$

$$\overline{Q'}_{<2uk_1>_{2^{t+1}}} \equiv \sum_{n_2=0}^{2^t-1} Q'_{n_2}(x) x^{2un_2} \ \text{modd}\, F_t, x^{2^t}+1,\ u = 0, 1, \cdots, 2^t - 1,$$

where in last equation is FNPT $(2^t, x^2, F_t, x^{2^t}+1)$. Then we compute[4]

$$\overline{Q}_{k_1, <2uk_1>_{2^{t+1}}} \equiv \overline{Q'}_{<2uk_1>_{2^{t+1}}}(x) \ \text{modd}\, F_t, x - 2^{k_1},$$

for $k_1 = 2v+1, v = 0, 1, .., 2^t - 1$. For $k_1 = 2v, v = 0, 1, .., 2^t - 1$,

$$\overline{Q}_{2v,2u}(x) \equiv \sum_{n_1=0}^{2^t-1} \sum_{n_2=0}^{2^t-1} \left[\left(q_{n_1,n_2} + q_{n_1,n_2+2^t}\right) + \left(q_{n_1+2^t,n_2} + q_{n_1+2^t,n_2+2^t}\right)\right] 2^{2vn_1} 2^{2un_2} \ \text{mod}\, F_t,$$
$$v, u = 0, 1, \cdots, 2^t - 1,$$

which is a 2–D FNT of size $2^t \times 2^t$, and can be computed similarly.

VI Computational Complexity and Comparisons

The computational complexities for the algorithms in the last two sections will be described in terms of numbers of *multiplications* (M), *additions* (A) and *shifts* (S), where shifts are the operations of multiplying a number by 2^i for some $i \geq 1$. Note that this definition of shifts is slightly different from the regular one which corresponds to multiplying a number by 2. One shift $(\times 2^i)$ here is actually i consecutive regular shifts. Therefore, we will treat shifts and additions as if they are in the same category in the following for simplicity.

Now we summarize the results of the analysis of computational complexities as follows, whose details can be found in the appendixes. To compute an $N \times N$ 2-D integer CC using MNPT, we need to perform the following numbers of operations:

$$\begin{aligned} M &= N^2, \\ S &= 2N^3 - 4N^2 + 2, \\ A &= 4N^3 - N^2 - 10N + 8. \end{aligned} \tag{IV.1}$$

[4]These operations are 2^t computing processes of the form (IV.15).

To compute a $2^{t+1} \times 2^{t+1}$, $t \geq 1$, 2-D integer CC using FNPT, we have the computational complexity:

$$\begin{aligned} M &= 1 + \sum_{q=0}^{t} 3 \times 2^{2q}, \\ S &= \sum_{q=0}^{t} 3 \times q \times 2^{2q}, \\ A &= \sum_{q=0}^{t} \left(3 \times q \times 2^{2q+2} + 2^{2q+4}\right). \end{aligned} \tag{IV.2}$$

We compare the above computational complexities with those for computing the same 2-D integer CCs using the corresponding NTTs with row-column scheme in Table 1 and Table 2. In Table 1, the first two columns under "Parameters" are the sizes N of 2-D integer CCs and the Mersenne number $M_N = 2^N - 1$ used. The next two groups of three columns are the computational complexities for MNPT and 2-D MNT with row-column scheme, respectively. Notice that both algorithms need the same number of multiplications; using MNPT saves some numbers of shifts, listed in column "S" of the last three columns, and needs more numbers of additions which are listed as negative numbers in the last column "A". However, the saving on shifts for MNPT are much larger and growing faster with N than the corresponding spending on additions in terms of numbers of operations. To get an approximate overall comparison between these two algorithm, we subtract the extra-spendings on additions from the saving on shifts for MNPT and enter the resulting numbers in the last column under "S+A" along with their percentages with respect to the corresponding numbers of shifts for the row-column scheme. With the above simplification, we conclude that using MNPT saves more than 42% of shifts as compared with the other algorithm.

In Table 2, which is similar to Table 1 in style, we compare the computational complexities of FNPT and 2-D FNT with row-column scheme. The numbers of multiplications and additions are the same for both algorithms, while less numbers of shifts are needed for FNPT; the savings are more than 25% and are growing when t is increasing (see Table 2).

The computational complexity of an $N \times N$ 2–D MNT using MNPT is

$$\begin{aligned} S &= N^3 - 2N^2 + 1, \\ A &= 2N^3 - N^2 - 4N + 4, \end{aligned} \tag{IV.3}$$

where S is half of the S in (IV.1), and A is less than half of the A in (IV.1) by an amount of $0.5N^2 - N$. The computational complexities[5], S and A, for computing the same $N \times N$ 2–D MNT using row-column scheme are halves of those for $N \times N$ 2-D integer CCs, and thus can be obtained directly from Table 1. The conclusion for the computational complexity comparison is similar to the one for $N \times N$ 2-D integer CCs.

The computational complexity of a $2^{t+1} \times 2^{t+1}$, $t \geq 1$, 2-D FNT using FNPT are exactly halves of those in (IV.2), and the savings of using FNPT with respect to the row-column scheme are the same as those percentages in Table 2. We have conducted a simulation experiment in which 2-D FNTs are programed in FORTRAN with the algorithm of FNPT and with the row-column scheme. We use general integer multiplications in FORTRAN to realize the shift operations in the algorithms. Since the general integer multiplication are more time-consuming than addition, the difference of the compuing-times for the above two programs to complete the same 2-D FNT can be an indicator for the difference of shifts used in the two algorithms. The resulting compuing-times, on a personal computer, as a function of the sizes of 2-D FNTs are shown in Figure 3 which indicates that using FNPT reduces compuing-times by about 50%. This matches with the above computational complexity assessment for these cases.

VII Summary and Conclusions

In this paper, we developed new fast algorithms for 2-D integer circular convolutions and 2-D NTTs. These new algorithms are constructed based on polynomial transforms over Z_p introduced here. Several necessary and sufficient conditions for the existence of polynomial transforms over Z_p are stated and proved. By applying these existence conditions, we have obtained two important polynomial transforms over Z_p: MNPT and FNPT, based on which we then developed fast algorithms for 2–D integer CCs, 2-D MNTs and 2-D FNTs. Comparing to the conventional row-column computation of 2-D NTTs for 2–D integer CCs and 2-D NTTs, the new algorithms save more than 25% or 42% of numbers

[5]There are no multiplications for computation of NTTs.

of operations for multiplying 2^i, $i \geq 1$; these percentages of savings also grow with the size of the 2-D integer CCs or the 2-D NTTs. These complexity savings of the new algorithms are also indicated by the computing-time results of a simulation experiment on computer.

Acknowledgment

The first author would like to thank Professor S. C. Zhou of Chongqing University, Chongqing, P. R. C., for his introduction of fast transforms, and for his teaching and advising.

Appendix A

This appendix contains some rather basic theorems and definitions required in the proofs of many lemmas and theorems presented in this paper.

Theorem A.1: For any two polynomials $f(x)$ and $g(x)$ in $Z_p[x]$, there exist polynomials: $m(x), n(x)$ in $Z_p[x]$, such that

$$m(x)f(x) + n(x)g(x) \equiv (f(x), g(x)) \bmod p, \tag{A.1}$$

where $(f(x), g(x))$ denotes the greatest common divisor of $f(x)$ and $g(x)$, (its leading coefficient is assumed to be unity in Z_p) [12].

Theorem A.2: Any polynomial in $Z_p[x]$ can be expressed as a product of a constant and irreducible polynomials in $Z_p[x]$, where the irreducible polynomials are of unity leading coefficients [12].

Theorem A.3: Let $a(x), b(x), c(x)$ be given polynomials in $Z_p[x]$. The congruent equation

$$a(x)p_1(x) + b(x)p_2(x) \equiv c(x) \bmod p \tag{A.2}$$

has solutions $p_1(x), p_1(x)$ in $Z_p[x]$, if and only if $(a(x), b(x))|c(x) \bmod p$.

Proof: The "Only if" part is obvious.

"If" part: Supposing $(a(x), b(x))|c(x)$, then $c(x) \equiv c_1(x)(a(x), b(x)) \bmod p$ for some polynomial $c_1(x)$. From Theorem A.1, we have $m(x), n(x)$, such that

$$c_1(x)m(x)a(x) + c_1(x)n(x)b(x) \equiv c_1(x)(a(x), b(x)) \bmod p$$

where we have used the fact: if $f_1(x) \equiv g_1(x)$, $f_2(x) \equiv g_2(x)$ mod p, then $f_1(x)f_2(x) \equiv g_1(x)g_2(x)$ mod p [12]. Therefore, equation (A.2) has solutions: $p_1(x) \equiv c_1(x)m(x)$, $p_2(x) \equiv c_1(x)n(x)$ mod p. QED.

Theorem A.4: The following first order congruent equation

$$a(x)p(x) \equiv b(x) \text{ modd } p, m(x) \tag{A.3}$$

has solution $p(x)$, if and only if $(a(x), m(x))|b(x)$ mod p.

Proof: By definition, (A.3) has solution if and only if $a(x)p(x) - m(x)p_1(x) \equiv b(x)$ mod p has solutions $p(x), p_1(x)$. Then by Theorem A.3, (A.3) has solution if and only if $(a(x), m(x))|b(x)$ mod p. QED.

Definition A.5: (Complete Residue System) Let the degree of a given polynomial $M(x)$ in $Z_p[x]$ be n. Then every polynomial is congruent, modd $p, M(x)$, to one of the following polynomials

$$a_1 + a_2x + \cdots + a_nx^{n-1},\ 0 \leq a_i \leq p-1,\ 1 \leq i \leq n, \tag{A.4}$$

which express p^n polynomials, among which no two of them are congruent $p, M(x)$. We call these p^n polynomials the complete residue system modd $p, M(x)$.

Definition A.6: (Reduce Residue System) A reduced residue system is derived from a complete residue system by discarding those polynomials which are not coprime with $M(x)$ mod p.

Definition A.7: Euler Function $\varphi(p, M(x))$ is defined to be the number of polynomials in the reduced residue system modd $p, M(x)$.

Theorem A.8: (Euler Theorem on Z_p) If $(M(x), f(x)) \equiv 1$ mod p, then

$$[f(x)]^{\varphi(p,M(x))} \equiv 1 \text{ modd}\, p, M(x).$$

Proof: Let $f_1(x), f_2(x), \cdots, f_{\varphi(p,M(x))}(x)$ be a reduced residue system modd $p, M(x)$ in some order. Then clearly $f(x)f_1(x), f(x)f_2(x), \cdots, f(x)f_{\varphi(p,M(x))}(x)$ modd $p, M(x)$ is also a reduced residue system, modd $p, M(x)$ [12]. Therefore,

$$\prod_{i=1}^{\varphi(p,M(x))} f_i(x) \equiv \prod_{i=1}^{\varphi(p,M(x))} f_i(x)f(x) \text{ modd}\, p, M(x),$$

i.e.,

$$\left[[f(x)]^{\varphi(p,M(x))}-1\right]\prod_{i=1}^{\varphi(p,M(x))} f_i(x)\equiv 0 \ \operatorname{modd} p, M(x).$$

Thus, $[f(x)]^{\varphi(p,M(x))} \equiv 1 \operatorname{modd} p, M(x)$. QED.

If $M(x)$ is an irreducible polynomial of degree n, mod p, then $\varphi(p, M(x)) = p^n - 1$, and we have the following Fermat theorem [12].

Theorem A.9 : Let $f(x)$ be a polynomial not divisible by $M(x)$ mod p. Then $[f(x)]^{p^n-1} \equiv 1 \operatorname{modd} p, M(x)$. Given any polynomial $f(x)$, we have $[f(x)]^{p^n} \equiv f(x) \operatorname{modd} p, M(x)$ and, in particular, $x^{p^n} \equiv x \operatorname{modd} p, M(x)$.

Definition A.10: The least positive integer l such that $[f(x)]^l \equiv 1 \operatorname{modd} p, M(x)$ is called the order of $f(x)$ modd $p, M(x)$.

Theorem A.11: Let $f(x)$ have order l modd $p, M(x)$. If $[f(x)]^m \equiv 1 \operatorname{modd} p, M(x)$, then $l|m$.

Proof: Supposing the theorem is not true, then we have two integers q, r, such that $m = ql + r$ where $q \geq 0$ and $0 < r < l$, and $1 \equiv [f(x)]^m \equiv [f(x)]^{ql+r} \equiv [f(x)]^r$ modd $p, M(x)$. This contradicts the definition of l. QED.

Theorem A.12 [12]: The number of roots of $f(X) \equiv 0 \operatorname{modd} p, M(x)$ does not exceed the degree of $f(X)$.[6]

Theorem A.13: For an irreducible polynomial $M(x)$ in $Z_p[x]$, if the order of a given polynomial $g(x)$ is l modd $p, M(x)$, then there are only $\varphi(l)$[7] distinct polynomials with order l modd $p, M(x)$.

Proof: From Theorem II.12, there are at most l different solutions for $[f(x)]^l \equiv 1$ modd $p, M(x)$. But we have the following l different solutions: $g(x), [g(x)]^2, ..., [g(x)]^l$. Thus they are the solutions of $[f(x)]^l \equiv 1$ modd $p, M(x)$.

Denote the order of $[g(x)]^r$ as l', for $1 \leq r \leq l$. Since $[g(x)]^{rl} \equiv 1$ modd $p, M(x)$, $l' \mid rl$. Assuming $(r, l) = 1$, then $l' \mid l$. But $[g(x)]^{rl'} \equiv 1$ modd $p, M(x)$, then $l \mid rl'$, and $l \mid l'$. Therefore, $l = l'$.

[6] Note: $f(X)$ is a polynomial in X, and a polynomial $g(x)$ is called a root of $f(X) \equiv 0 \operatorname{modd} p, M(x)$, if $f(g(x)) \equiv 0 \operatorname{modd} p, M(x)$.

[7] $\varphi(l)$ is the Euler function which is defined to be the number of positive integers smaller than l and being coprime with l.

Conversely, if the order of $[g(x)]^r$ is l, then $(r,l)=1$. Otherwise, $(r,l)=d>1$, then, $[g(x)]^{rl/d} \equiv 1$ modd $p, M(x)$, and this is a contradiction. QED.

Theorem A.14: Let $l \mid (p^n - 1)$. For an irreducible polynomial $M(x)$ of degree n in $Z_p[x]$, the number of distinct polynomials with order l modd $p, M(x)$, is $\varphi(l)$.

Proof: Let $\phi(l)$ be the number of polynomials of order l modd $p, M(x)$ in the reduced residue system modd $p, M(x)$. Then $\sum_{l|p^n-1} \phi(l) = p^n - 1$ from Theorem A.9 and A.11. It can be easily verified that $\sum_{l|p^n-1} \varphi(l) = p^n - 1$ [13]. Thus, $\sum_{l|p^n-1} (\phi(l) - \varphi(l)) = 0$. Therefore, $\phi(l) = \varphi(l)$, since, by Theorem A.13, $\phi(l) \le \varphi(l)$. QED.

Appendix B

In this appendix, two additional necessary and sufficient conditions for the existence of polynomial transforms over Z_p are proved, where p is assumed to be a prime number.

Proof of Theorem III.3: The decomposition expression of $M(x)$ is supported by Theorem A.2. From (i), we get $S(0) \equiv N$ modd $p, M(x)$. We denote the degree of $b_i(x)$ by n_i, for $1 \le i \le s$. Obviously, $n_i \ge 1$. From (ii) and Theorems A.9 and A.11, we have $N \mid p^{n_i} - 1$, i.e., $p^{n_i} - 1 = Nk_i$. Thus, p does not divide N, for otherwise, $N = pk_0$ for some k_0, and then, $p^{n_i} - 1 = pk_ik_0$, which is not true. When $q \not\equiv 0$ mod N, $([g(x)]^q - 1)S(q) = [g(x)]^{qN} - 1 \equiv 0$ modd $p, M(x)$. From (ii), we also have $([g(x)]^q - 1, M(x)) \equiv 1$, mod p. Thus, $S(q) \equiv 0$ modd $p, M(x)$, if $q \not\equiv 0$ mod N. Therefore, $(N, g(x), p, M(x))$ exists, from Theorem III.2.

On the other hand, when $(N, g(x), p, M(x))$ exists, from (III.3) and (III.4), we have $[g(x)]^N \equiv 1$ modd $p, M(x)$, $[g(x)]^N \equiv 1$ modd $p, b_i(x)$, and $S(q) \equiv 0$ modd $p, b_i(x)$, for $q \not\equiv 0$ mod N. Thus, $[g(x)]^q - 1 \not\equiv 0$, modd $p, b_i(x)$, for $q \not\equiv 0$ mod N for otherwise, $p \mid N$, and this contradicts the requirement $(p,N) = 1$. QED.

Proof of Theorem III.4: Suppose $(N, g(x), p, M(x))$ exists for some $g(x)$. Then, from Theorems III.3, A.9 and A.11, we know $N \mid (p^{n_i} - 1), 1 \le i \le s$. Conversely, from N divides the greatest common divisor of $p^{n_1} - 1, p^{n_2} - 1, ..., p^{n_s} - 1$, we have $N \mid (p^{n_i} - 1), 1 \le i \le s$. From Theorem A.14, for each i, there is $g_i(x)$ with order N modd $p, b_i(x)$. Thus, we have $g(x)$, such that, $g(x) \equiv g_i(x)$, modd $p, b_i(x)$, from Theorem II.1. Since $[g_i(x)]^N \equiv 1$, modd

$p, b_i(x)$, $[g(x)]^N \equiv 1$, modd $p, b_i(x)$. Thus $[g(x)]^N \equiv 1$, modd $p, M(x)$. Then based on Theorem III.3, we proved this Theorem. QED.

Appendix C

The lemmas in this appendix are related to MNPT and FNPT, and will be used in the development of the fast algorithms for 2-D integer CCs.

Lemma C.1: $x-1$ and $(x^N-1)/(x-1)$ are coprime, mod N.

Proof: Suppose that this is not true, i.e., they have a common non-trivial factor, then $(x^N-1)/(x-1)$ must have the factor $x-1$. Thus $(x^N-1)/(x-1) = x^{N-1}+x^{N-2}+\ldots+x+1 \equiv N \equiv 0$ modd $M_N, x-1$ which is a contradiction since $M_N|N$ can not hold. QED.

Lemma C.2:

$$\frac{x^N-1}{x-1} \equiv \begin{cases} N & \operatorname{modd} M_N, x-1 \\ 0 & \operatorname{modd} M_N, \left(x^N-1\right)/(x-1) \end{cases},$$

$$N - \frac{x^N-1}{x-1} \equiv \begin{cases} 0 & \operatorname{modd} M_N, x-1 \\ N & \operatorname{modd} M_N, \left(x^N-1\right)/(x-1) \end{cases}.$$

Proof: Obvious. QED.

Lemma C.3: Let $\{\alpha_i\}_{i=1}^k \subset Z_p$ be k distinct solutions of the equation

$$f(x) \equiv x^n + a_{n-1}x^{n-1} + \cdots + a_0 \equiv 0, \bmod p,$$

and $(\alpha_i - \alpha_j, p) = 1$ for $i \neq j$. Then

$$f(x) \equiv (x-\alpha_1)(x-\alpha_2)\cdots(x-\alpha_k)f_k(x) \bmod p,$$

where $f_k(x)$ is a polynomial of degree $n-k$ and with the leading coefficient 1, and p is not necessarily a prime number.

Proof: Since $f(\alpha_1) \equiv 0 \bmod p$, then

$$f(x) \equiv f(x) - f(\alpha_1) \equiv x^n - \alpha_1^n + a_{n-1}\left(x^{n-1} - \alpha_1^{n-1}\right) + \cdots + a_1(x-\alpha_1) \equiv (x-\alpha_1)f_1(x) \bmod p,$$

where $f_1(x)$ is a polynomial of degree $n-1$ and with the leading coefficient 1. Since we also have $f(\alpha_2) \equiv 0 \bmod p$, and $(\alpha_1 - \alpha_2, p) = 1$, then $f_1(\alpha_2) \equiv 0 \bmod p$. By repeating the above argument, the Lemma is proved. QED.

Lemma C.4:

$$x^{2^{t+1}}-1 \equiv \left(x-2^0\right)\left(x-2^1\right)\left(x-2^2\right)\cdots\left(x-2^{2^{t+1}-1}\right) \bmod F_t.$$

Proof: It can be easily shown that the order of 2 modulo F_t is 2^{t+1}. Thus

$$2^0, 2^1, 2^2, \cdots, 2^{2^{t+1}-1} \tag{C.1}$$

are 2^{t+1} incongruent numbers mod F_t. Let $p|F_t$ be a prime number. Then the order of 2 modulo p is also 2^{t+1} [14]. Thus, the 2^{t+1} numbers in (C.1) are also incongruent modulo p, i.e., $(2^i-2^j, F_t)=1$, for $i \neq j$, $i,j=0,1,\ldots,2^{t+1}-1$. Since the 2^{t+1} numbers in (C.1) are solutions of the equation $x^{2^{t+1}}-1 \equiv 0 \bmod F_t$, from Lemma C.3 we proved the lemma. QED.

In (C.1), the numbers with even power: $2^0, 2^2, \cdots 2^{2^{t+1}-2}$ are also solutions of $x^{2^t}-1 \equiv 0 \bmod F_t$. Thus, from $x^{2^{t+1}}-1=\left(x^{2^t}-1\right)\left(x^{2^t}+1\right)$, we have the following corollary

Corollary C.5:

$$x^{2^t}+1 \equiv (x-2)\left(x-2^3\right)\cdots\left(x-2^{2^{t+1}-1}\right) \equiv \sum_{k_0=0}^{2^t-1}\left(x-2^{2k_0+1}\right) \bmod F_t,$$

and $(x^{2^t}+1, x^{2^t}-1) \equiv 1 \bmod F_t$. More generally,

$$x^{2^{t-i}}+1 \equiv \sum_{k_i=0}^{2^{t-i}-1}\left(x-2^{(2k_i+1)2^i}\right) \bmod F_t,$$

and

$$\left(x^{2^{t-i}}+1, x^{2^{t-i}}-1\right) \equiv 1 \bmod F_t,$$

for $0 \leq i \leq t$.

Lemma C.6:

$$2^{2^{t+1}-1}\left(x^{2^{t-i}}+1\right) \equiv \begin{cases} 1 & \operatorname{modd} F_t, x^{2^{t-i}}-1 \\ 0 & \operatorname{modd} F_t, x^{2^{t-i}}+1 \end{cases},$$

$$2^{2^t-1}\left(x^{2^{t-i}}-1\right) \equiv \begin{cases} 0 & \operatorname{modd} F_t, x^{2^{t-i}}-1 \\ 1 & \operatorname{modd} F_t, x^{2^{t-i}}+1 \end{cases}.$$

Proof: Because that

$$x^{2^{t-i}}+1 \equiv 2 \ \operatorname{modd}\ F_t, x^{2^{t-i}}-1,$$

$$x^{2^{t-i}}-1 \equiv -2 \equiv 2^{2^t+1} \ \operatorname{modd}\ F_t, x^{2^{t-i}}+1,$$

and $2, 2^{2^t+1}$ have inverse $2^{2^{t+1}-1}, 2^{2^t-1}$, respectively, mod F_t. QED.

References

[1] W. Pratt, *Digital Image Processing.* New York: Wiley, 1978.

[2] A. N. Netravali and B. G. Haskell, *Digital Pictures.* New York: Plenum Press, 1988.

[3] J. W. Woods, *Subband Image Coding.* Boston: Kluwer Academic Publishers, 1991.

[4] R. E. Blahut, *Fast Algorithms for Digital Signal Processing.* Menlo Park, CA: Addison-Wesley Publishing Company, 1985.

[5] F. L. J. G. Proakis, C. M. Rader and C. L. Nikias, *Advanced Digital Signal Processing.* New York: Macmillan Company, 1992.

[6] G. Birkhoff and S. M. Lane, *A Survey of Modern Algebra.* New York, London: Macmillan Company, 1968.

[7] J. B. Martens, "Number theoretic transforms for the calculation of convolutions," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-31, pp. 969–978, Aug. 1983.

[8] J. H. McClellan and C. M. Rader, *Number Theory in Digital Signal Processing.* Englewood Cliffs, New Jersey: Prentice-Hall, 1979.

[9] E. Bubois and A. N. Venetsanopoulos, "The discrete fourier transform over finite rings with application to fast convolution," *IEEE Trans. Compu.*, vol. C-27, pp. 586–593, July 1978.

[10] D. P. Maher, "On fourier transforms over extension of finite rings," *IEEE Trans. Computers*, vol. C-29, pp. 331–333, Apr. 1980.

[11] H. J. Nussbaumer and P. Quandelle, "Computation of convolutions and discrete fourier transform by polynomial transforms," *IBM J. Res. Develop.*, vol. 22, pp. 134–144, Mar. 1978.

[12] L. K. Hua, *Introduction to Number Theory.* Berlin, Heidelberg, New York: Springer-Verlag, 1982.

[13] H. J. Nussbaumer, *Fast Fourier Transform and Convolution Algorithms.* Berlin, New York: Springer-Verlag, 1982.

[14] J.-Y. Jang, *Number Theory Transform.* Beijing: Science Publishing Institute, 1980.

[15] J.-Y. Jang, "Polynomial transforms and their applications in the calculation of convolutions," *Computational Math.*, vol. 3, pp. 230–241, Aug. 1983.

[16] E. Bubois and A. N. Venetsanopoulos, "Convolution using a conjugate symmetry property for the generalized discrete fourier transform," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-26, pp. 165–170, Apr. 1978.

[17] B. Arambela and P. J. W. Rayner, "Discrete transformations over polynomial rings with application in computing multi-dimensional convolutions," *IEEE Trans. Acust., Speech, Signal Processing*, vol. ASSP-28, pp. 407–414, Aug. 1980.

[18] D.-X. Z. Qi Suen and Z.-Q. Sen, *Fast Number Theory Transforms.* Beijing: Science Publishing Institute, 1980.

[19] A. V. Oppenheim and R. W. Schafer, *Digital Signal Processing.* Prentice-Hall, 1975.

[20] J.-L. Wu and Y. M. Huang, "Two-variable modularized fast polynomial transform algorithm for 2-d discrete fourier transforms," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 2, pp. 84–87, Mar. 1992.

[21] H. J. Nussbaumer, "Fast polynomial transform algorithms for digital convolution," *IEEE Trans. Acust., Speech, Signal Processing*, vol. ASSP-28, pp. 205–215, Apr. 1980.

[22] H. J. Nussbaumer, "New polynomial transforms for multi-dimensional dft's and convolutions," *IEEE Trans. Acust., Speech, Signal Processing*, vol. ASSP-29, pp. 74–84, Feb. 1981.

[23] X. Li and G. Qian, "Block size considerations for multidimensional convolution and correlation," *IEEE Trans. Signal Processing*, vol. 40, pp. 1271–1273, May 1992.

Parameters		MNPT			2-D MNT (row-column scheme)			Savings by using MNPT		
N	M_N	M	S	A	M	S	A	S	A	S+A
3	7	9	20	77	9	48	72	28	-5	23 (48%)
5	31	25	152	733	25	320	400	168	-33	135 (42%)
7	127	49	492	1261	49	1008	1176	516	-85	431 (43%)
13	8191	169	3720	8497	169	7488	8112	3768	-385	3383 (45%)
17	131071	289	8672	19201	286	17408	18496	8736	-705	8031 (46%)
19	524287	361	12276	26893	361	24624	25992	12348	-901	11447 (46%)

Table 1: Comparison of computation complexity for computing 2–D integer circular convolutions using MNPT and 2-D Mersenne Number Transform (MNT) with row-column scheme.

Parameters			FNPT			2-D FNT (row-column scheme)			Savings by using FNPT
t	F_t	2^{t+1}	M	S	A	M	S	A	S
1	5	4	16	12	128	16	16	128	4 (25%)
2	17	8	64	108	768	64	160	768	52 (33%)
3	257	16	256	684	4096	256	1088	4096	404 (37%)
4	65537	32	1024	3756	20480	1024	6272	20480	2516 (40%)

Table 2: Comparison of computation complexity for computing 2–D integer circular convolutions using FNPT and 2–D Fermat Number Transform (FNT) with row-column scheme.

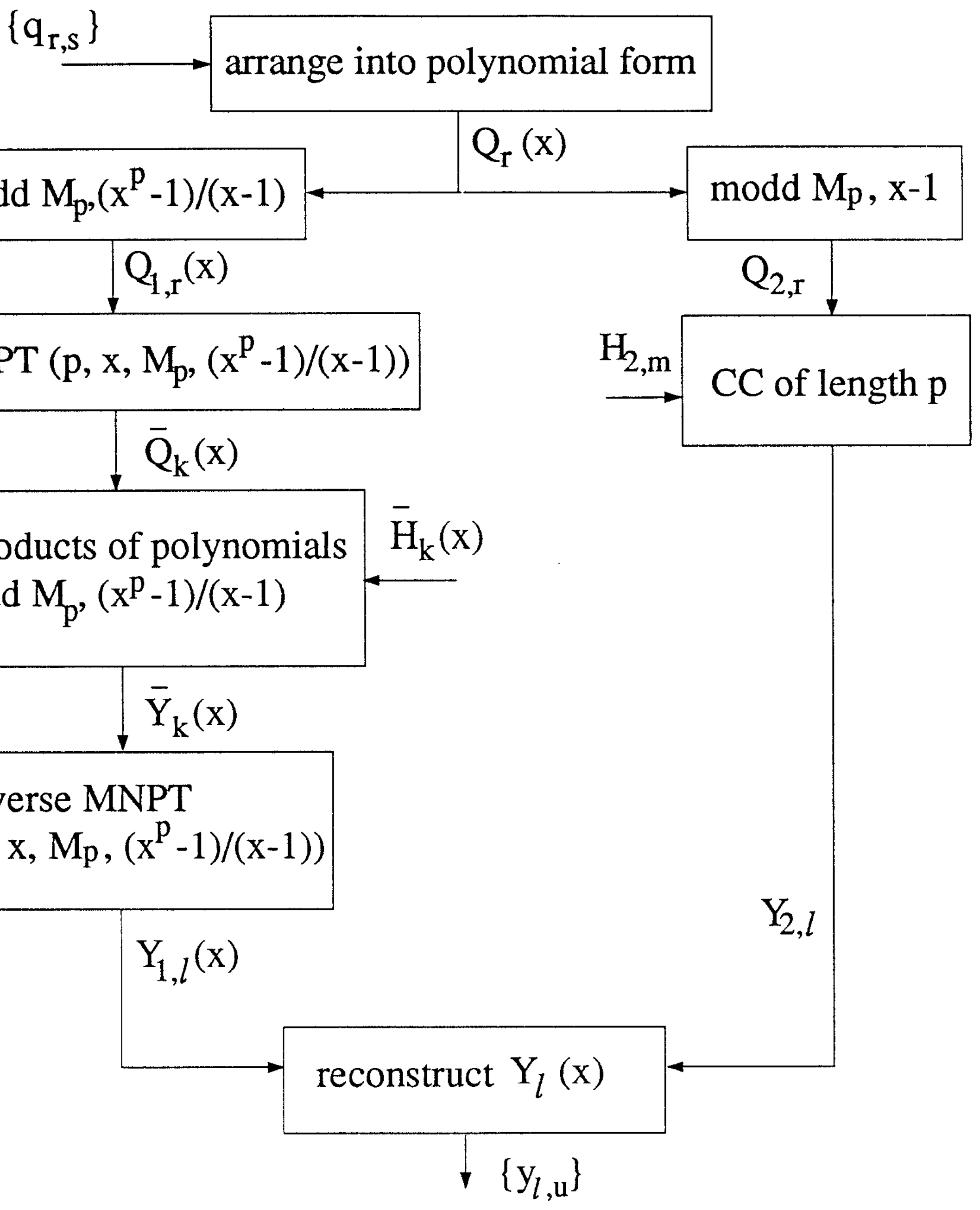
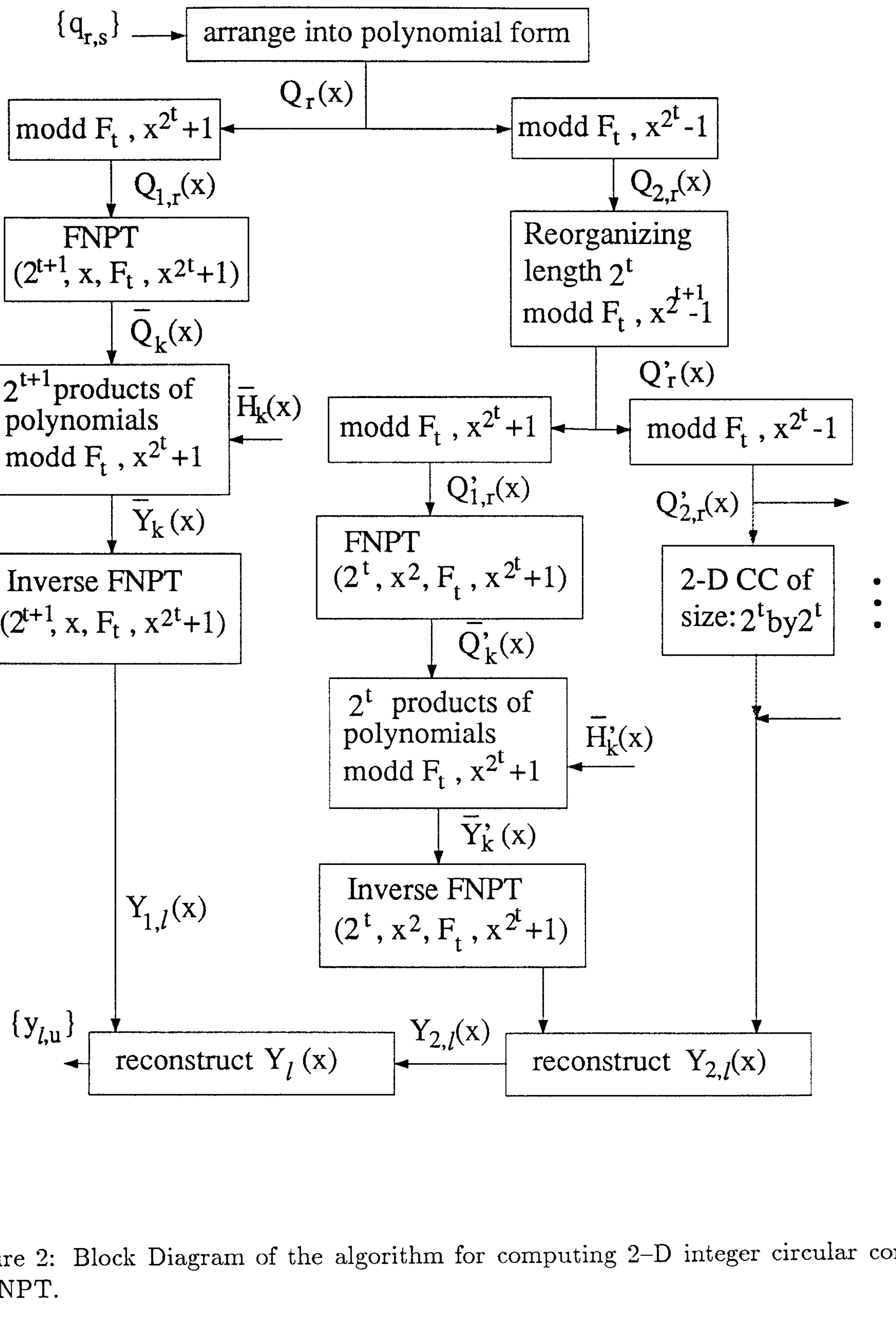


Figure 1: Block diagram of the algorithm for computing 2–D integer circular convolution using MNPT.



$\{q_{r,s}\}$
arrange into polynomial form
$Q_r(x)$
modd F_t, $x^{2^t}+1$
modd F_t, $x^{2^t}-1$
$Q_{1,r}(x)$
$Q_{2,r}(x)$
FNPT $(2^{t+1}, x, F_t, x^{2^t}+1)$
Reorganizing length 2^t modd F_t, $x^{2^{t+1}}-1$
$\bar{Q}_k(x)$
$Q'_r(x)$
2^{t+1} products of polynomials modd F_t, $x^{2^t}+1$
$\bar{H}_k(x)$
modd F_t, $x^{2^t}+1$
modd F_t, $x^{2^t}-1$
$\bar{Y}_k(x)$
$Q'_{1,r}(x)$
$Q'_{2,r}(x)$
Inverse FNPT $(2^{t+1}, x, F_t, x^{2^t}+1)$
FNPT $(2^t, x^2, F_t, x^{2^t}+1)$
2-D CC of size: 2^t by 2^t
$\bar{Q}'_k(x)$
2^t products of polynomials modd F_t, $x^{2^t}+1$
$\bar{H}'_k(x)$
$\bar{Y}'_k(x)$
Inverse FNPT $(2^t, x^2, F_t, x^{2^t}+1)$
$Y_{1,l}(x)$
$\{y_{l,u}\}$
$Y_{2,l}(x)$
reconstruct $Y_l(x)$
reconstruct $Y_{2,l}(x)$

Figure 2: Block Diagram of the algorithm for computing 2–D integer circular convolution using FNPT.

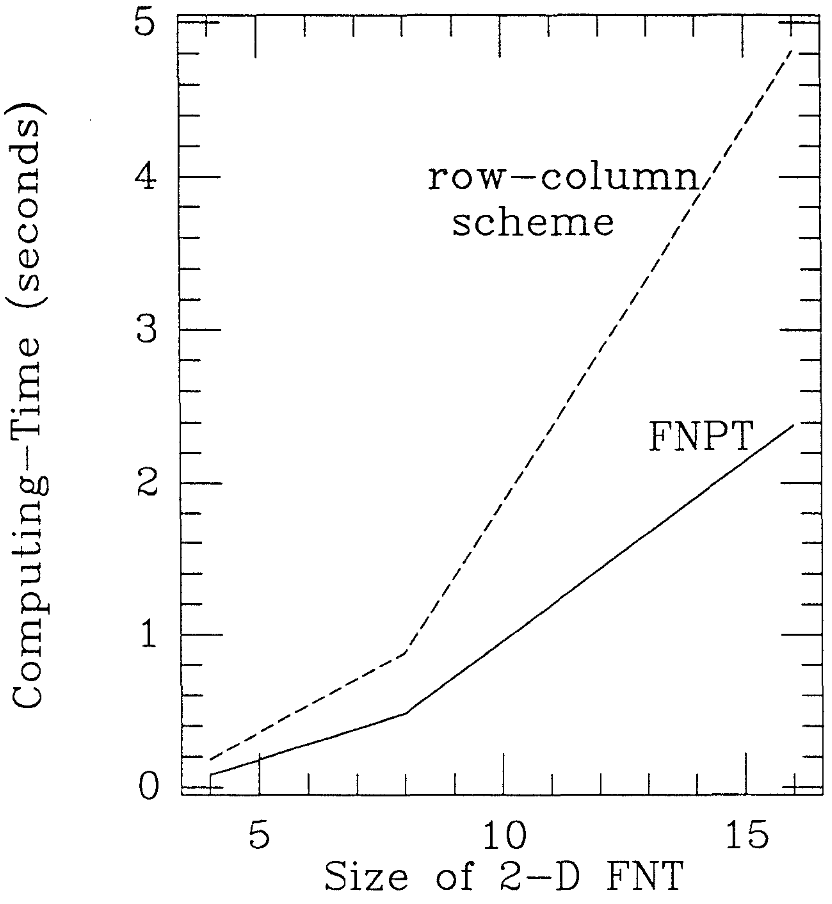


Figure 3: Computing-time comparison for calculation of 2-D Fermat Number Transforms (FNT) using FNPT and the row-column scheme.