Explicit Cook-Toom Algorithm for Linear Convolution

Yuke Wang
Department of Computer Science and Engineering
Florida Atlantic University
Boca Raton, FL 33431, USA
yuke@cse.fiu.edu

Keshab Parhi
Department of Electrical and Computer Engineering
University of Minnesota
Minneapolis, Minnesota, USA
parhi@ece.umn.edu

Abstract
The short length linear convolution, conventionally computed by the Cook-Toom algorithm, is important since it is the building block of large convolution algorithms. To compute the linear convolution of $N$ and $M$ points, the Cook-Toom algorithm computes the Lagrange interpolation at $L = N + M - 1$ real numbers. However, the computation is often tedious and has only been carried out for special integers. In this paper, we present an explicit general formula for linear convolutions which calculates the interpolation at $L = 2$ general non-zero points. We further investigate the linear convolution from VLSI implementation point of view.

§1 Introduction
The fast and exact convolution is one of the most important digital signal processing problems [1][2][3]. There are two kinds of convolutions: the linear convolution, which is also called the noncyclic convolution or aperiodic convolution, and the cyclic convolution. Most fast convolution algorithms compute a cyclic convolution, while most applications call for a linear convolution [4]. On the other hand, cyclic convolutions can be computed by finding the linear convolutions and reducing them by modulo $x^N - 1$. Hence efficient ways of computing linear convolutions also lead to efficient ways of computing cyclic convolutions.

The linear convolution can be computed directly using $MN$ multiplications and $(M - 1)(N - 1)$ additions while the cyclic convolution can be computed in $N^2 + N$ multiplications and $N(N - 1)$ additions. These numbers grow very fast as $N$ and $M$ increase. Hence much effort has gone into developing alternative and more efficient ways of implementing convolutions. As pointed out in [2], the conventional approach for speeding up the calculation of convolutions is based on the fast Fourier transform (FFT). The speed advantage offered by the FFT algorithm can be very large for long convolutions and the method is by far the most commonly used for fast computation of convolutions. However, there are several drawbacks of the FFT approach, which relate mainly to the use of sines and cosines and to the need for complex arithmetic, even if the convolutions are real. Most fast convolution algorithms, such as those based on the FFT, apply only to periodic functions and therefore compute only cyclic convolutions. In order to overcome the limitations of the FFT method, many other fast algorithms have been proposed.

An alternative way to compute the convolution is by converting the one-dimensional convolution into a multi-dimensional convolution [5]. Each of the one-dimensional convolutions in this multidimensional convolution is then performed using efficient short-length algorithms. The calculation of a convolution by cyclic or linear nesting of small length convolutions has many desirable attributes. It requires fewer arithmetic operations than the FFT approach for sequence lengths of up to 200 samples and does not require complex arithmetic with sines and cosines [2]. In recent years, such algorithms have been implemented in VLSI circuits [6].

However, as pointed out in [2], the large algorithms will be good only if the small algorithms are good. Thus it is important to find the best possible small convolution algorithms [2]. In the literature, short length linear convolution algorithms are derived for the length of $2 \times 2 \times 3 \times 3 \times 4 \times 4$. The $2 \times 2$ linear convolution may seem to be too small a problem for any practical application. In fact, a good algorithm for $2 \times 2$ linear convolution is a good building block from which one can construct more elaborate algorithms. Hence such short length convolution algorithms are of practical importance [page 70 [2]].

The short-length linear convolutions are computed by the Cook-Toom algorithm. When the Cook-Toom algorithm is used to compute the linear convolution of $N$ and $M$ points, one has to choose $L = N + M - 1$ real points and compute the function values at those points, after which the Lagrange interpolation formula is used to find the polynomial resulting from the linear convolution. However, the computation involving Lagrange interpolation is often tedious and has only been carried out for special integers. Finally, when applying the Cook-Toom algorithm, significant reduction of operation counts occurs if the $L$ numbers are carefully chosen [7]. A better algorithm can be produced if we only choose $L - 1$ distinct numbers. However, there is no general explicit algorithm on how to choose the numbers and to calculate the convolution.

In this paper, we present an explicit general formula for linear convolutions which calculates the interpolation at $L = 2$ general non-zero points. We further investigate the short-length convolution from VLSI implementation point of view. If the linear convolution algorithms are implemented in VLSI circuits, their operation counts can be reduced from the operation counts for software implementations, due to the fact that multiplying by numbers such as $2^4$ is available for free in hardware while it is not in software.

The rest of this paper is organized as follows. In Section 2, we review all the short-length linear convolutions studied in the literature. In Section 3, we introduce a general formula for linear convolution which calculates the interpolation at $L = 2$ general non-zero points. Short-length linear convolutions are derived as examples of the general formula. In Section 4, we conclude the paper.
§11 Background and Previous Algorithms

The linear convolution $s_{i}$ of two sequences $d_{m}$ and $g_{n}$ of length $M$ and $N$ respectively can be written compactly as a polynomial product, $\deg D(x) = N - 1$ and $\deg G(x) = M - 1$.

$$S(x) = D(x)G(x) = \sum_{i=0}^{N-M+1} s_{i}x^{i}$$

(1)

The coefficients $s_{i}$ are given by $s_{i} = \sum_{m=0}^{N-M-2} d_{m}g_{i+m}$, $i = 0, 1, ..., N + M - 2$.

**Cook-Toom Algorithm** If $r_{0}, r_{1}, ..., r_{N+M-2}$ are $L = N + M - 1$ distinct real numbers, the Cook-Toom algorithm calculates $D(r_{i}), G(r_{i}),$ and $S(r_{i}) = D(r_{i})G(r_{i})$. $S(x)$ can be interpolated from $S(r_{i}), S'(r_{i}), ..., S(N+M-2)$ using Lagrange's interpolation formula

$$S(x) = \sum_{i=0}^{L-1} S(r_{i}) \prod_{j \neq i}^{L}(x - r_{j}) / \prod_{j \neq i}^{L}(r_{i} - r_{j})$$

(2)

Explicitly, the above equation can be solved using the inverse Vandermonde matrix of the form shown in Formula (2). An explicit formula for calculating the inverse of the Vandermonde matrix can be found in [9].

$$\begin{bmatrix}
    S_{0} \\
    \vdots \\
    S_{L-1}
\end{bmatrix} = \begin{bmatrix}
    r_{0} & r_{1}^2 & \cdots & r_{L-1}^{L-1} \\
    r_{1} & r_{1}^2 & \cdots & r_{L-1}^{L-1} \\
    \vdots & \vdots & \ddots & \vdots \\
    r_{L-1} & r_{L-1}^2 & \cdots & r_{L-1}^{L-1}
\end{bmatrix}^{-1} \begin{bmatrix}
    s_{0} \\
    \vdots \\
    s_{L-1}
\end{bmatrix}$$

When applying the Cook-Toom algorithm, significant reduction of operation counts occurs if the numbers $r_{0}, r_{1}, ..., r_{N+M-2}$ are carefully chosen [7]. A better algorithm can be produced if we only choose $L - 1$ distinct numbers and compute the convolution using the following formula.

$$S(x) = ([D(x)G(x) \bmod \prod_{i=0}^{L-1}(x - r_{i})] + g_{N-M} \cdot \prod_{i=0}^{L-1}(x - r_{i})) \bmod (x - \infty)$$

(3)

The convolution using Formula (3) is done using Chinese Remainder Theorem (CRT) [2][8] which requires long calculation. In the literature, only a few specific algorithms have been provided. Computing $S(x)$ by the above two stages shown in Formula (3) is denoted by $S(x) = D(x)G(x) \bmod \prod_{i=0}^{L-1}(x - r_{i})(x - \infty)$.

**Algorithm 1** We compute the $2 \times 2$ linear convolution of two polynomials $d(x) = d_{1}x + d_{0}$ and $g(x) = g_{1}x + g_{0}$. The result is $s(x) = s_{0}x + s_{1}x^{2} + s_{2}x^{3} + s_{3}x^{4}$.

This example has been shown in [2][4][7].

Let $r_{0} = 0, r_{1} = -1, r_{2} = 2, r_{3} = \infty$, then we have

$$
\begin{align*}
G_{0} &= g_{0} \\
D_{0} &= d_{0} \\
G_{1} &= g_{1} + g_{0} \\
D_{1} &= d_{1} + d_{0} \\
G_{2} &= g_{1} - g_{0} \\
D_{2} &= d_{1} - d_{0}
\end{align*}
$$

According to (3), we have the following:

$$S(x) = L_{0}(x)S_{0}(x + 1) + L_{1}(x)S_{1}(x + 1) + L_{2}(x)S_{2}(x + 1) + L_{3}(x)S_{3}(x + 1)$$

where $L_{0}(x) = \operatorname{mod}(x, L_{0}(x))$, $L_{1}(x) = \operatorname{mod}(x + 1)$, i.e., $L_{0}(x) = 1$, $L_{1}(x) = 1$. Finally, we have

$$s(x) = S_{0}(x + 1) - S_{1}(x + 2) + S_{2}(x + 2) + S_{3}(x + 3)$$

$$= \frac{G_{0}D_{0}}{2} + G_{1}D_{1} + G_{2}D_{2} + G_{3}D_{3}$$

The matrix representation of the above process is as follows.

$$\begin{bmatrix}
    s_{0} \\
    s_{1} \\
    s_{2} \\
    s_{3}
\end{bmatrix} = \begin{bmatrix}
    1 & 0 & 0 & \frac{G_{0}D_{0}}{2} \\
    1 & -1 & 1 & G_{1}D_{1} \\
    0 & 0 & 1 & G_{2}D_{2} \\
    0 & 0 & 0 & G_{3}D_{3}
\end{bmatrix}\begin{bmatrix}
    d_{0} \\
    d_{1} \\
    d_{2} \\
    d_{3}
\end{bmatrix}$$

(4)

**Algorithm 2** We compute the $2 \times 3$ linear convolution of two polynomials $d(x) = d_{2}x^{2} + d_{1}x + d_{0}$ and $g(x) = g_{1}x + g_{0}$.

This example has been shown in [2][7].

Let $r_{0} = 0, r_{1} = -1, r_{2} = 1, r_{3} = \infty$, then we have

$$
\begin{align*}
G_{0} &= g_{0} \\
D_{0} &= d_{0} \\
G_{1} &= g_{1} + g_{0} \\
D_{1} &= d_{1} + d_{0} \\
S_{0} &= S_{0}D_{0} \\
S_{1} &= S_{1}D_{1} \\
S_{2} &= S_{2}D_{2} \\
S_{3} &= S_{3}D_{3}
\end{align*}
$$

According to (3), we have the following:

$$S(x) = L_{0}(x)S_{0}(x + 1) + L_{1}(x)S_{1}(x + 1) + L_{2}(x)S_{2}(x + 1) + L_{3}(x)S_{3}(x + 1)$$

where $L_{0}(x) = \operatorname{mod}(x, L_{0}(x))$, $L_{1}(x) = \operatorname{mod}(x + 1)$, $L_{2}(x) = \operatorname{mod}(x + 2)$, $L_{3}(x) = \operatorname{mod}(x + 3)$.

Expanding the above $S(x)$, we can obtain the coefficients $s_{i}$ as follows.

$$\begin{bmatrix}
    s_{0} \\
    s_{1} \\
    s_{2} \\
    s_{3}
\end{bmatrix} = \begin{bmatrix}
    1 & 0 & 0 & \frac{G_{0}D_{0}}{2} \\
    1 & -1 & 1 & G_{1}D_{1} \\
    0 & 0 & 1 & G_{2}D_{2} \\
    0 & 0 & 0 & G_{3}D_{3}
\end{bmatrix}\begin{bmatrix}
    d_{0} \\
    d_{1} \\
    d_{2} \\
    d_{3}
\end{barray}$$

(5)

**Algorithm 3** We compute the $3 \times 3$ linear convolution of two polynomials $g(x) = g_{2}x^{2} + g_{1}x + g_{0}$, $d(x) = d_{3}x^{3} + d_{2}x^{2} + d_{1}x + d_{0}$.

This example has been shown in [2][4][7].

Let $r_{0} = 0, r_{1} = -1, r_{2} = 1, r_{3} = 2, r_{4} = \infty$, then we have

$$
\begin{align*}
G_{0} &= g_{0} \\
D_{0} &= d_{0} \\
G_{1} &= g_{1} + g_{0} \\
D_{1} &= d_{1} + d_{0} \\
S_{0} &= S_{0}D_{0} \\
S_{1} &= S_{1}D_{1} \\
S_{2} &= S_{2}D_{2} \\
S_{3} &= S_{3}D_{3}
\end{align*}
$$

According to (3), we have the following:

$$S(x) = L_{0}(x)S_{0}(x + 1) + L_{1}(x)S_{1}(x + 1) + L_{2}(x)S_{2}(x + 1) + L_{3}(x)S_{3}(x + 1)$$

Expanding this equation, we obtain the following coefficients represented by the matrix form.

$$\begin{bmatrix}
    s_{0} \\
    s_{1} \\
    s_{2} \\
    s_{3}
\end{bmatrix} = \begin{bmatrix}
    G_{0} + S_{0}G_{0} & G_{0}D_{0} & S_{0}D_{0} & G_{0}D_{3} \\
    G_{1} + S_{1}G_{0} & G_{1}D_{0} & S_{1}D_{0} & G_{1}D_{3} \\
    G_{2} + S_{2}G_{0} & G_{2}D_{0} & S_{2}D_{0} & G_{2}D_{3} \\
    G_{3} + S_{3}G_{0} & G_{3}D_{0} & S_{3}D_{0} & G_{3}D_{3}
\end{bmatrix}\begin{bmatrix}
    d_{0} \\
    d_{1} \\
    d_{2} \\
    d_{3}
\end{barray}$$

(6)

3280
§III New Convolution Algorithms

§3.1 A New General Convolution Algorithm

The Cook-Toom algorithm finds the interpolation using Lagrange’s interpolation formula by interpolating at \( L = N + M - 1 \) distinct real numbers. Its improved version using Formula (3) based on Chinese Remainder Theorem needs interpolation at \( L - 2 = N + M - 3 \) real numbers. In the following, we introduce a new algorithm which only needs to interpolate at \( L - 2 = N + M - 3 \) non-zero real numbers. Moreover, our new algorithm is in explicit format as in Formula (2). We introduce Algorithm 4 without the proof due to space limitations. It can be proved based on the recently proposed New Chinese Remainder Theorems [8].

Algorithm 4 The linear convolution \( S(x) = D(x)G(x) = \sum_{k=0}^{M+N-2} s_k \cdot x^k = \sum_{k=0}^{M} s_k \cdot x^k \) of two sequences \( D(x) = \sum_{k=0}^{N-1} d_k \cdot x^k \) and \( G(x) = \sum_{k=0}^{M-1} g_k \cdot x^k \) of length \( N \) and \( M \) can be found by the following formula, where \( r_1, \ldots, r_k \) are \( L - 2 = N + M - 3 = k \) distinct non-zero numbers.

\[
\begin{align*}
D(x)G(x) &= d_0 + \sum_{k=1}^{L-2} d_k \cdot \frac{d_0 \cdot G(r_k) - d_k \cdot G(r_{k-1})}{r_k - r_{k-1}} \\
&= d_0 + \sum_{k=1}^{L-2} d_k \cdot \frac{D(r_k)G(r_k) - d_k \cdot G(r_k)}{r_k - r_{k-1}}
\end{align*}
\]

(7)

The above Formula (7) is an explicit formula for the linear convolution. Based on (7), we can derive the short length linear convolution algorithms simply by substituting \( k = N + M - 3 \).

(i) \( 2 \times 2 \) convolution, \( k = N + M - 3 = 1 \)

\[
\begin{align*}
s_0 &= 1 \quad 0 \quad 0 \\
s_1 &= 0 \quad 1 \quad -r_1 \\
s_2 &= 0 \quad 0 \quad 1 \\
&\quad \frac{d_0 \cdot g_0 - d_1 \cdot g_1}{r_1 - r_0}
\end{align*}
\]

(8)

(ii) \( 2 \times 3 \) convolution, \( k = N + M - 3 = 2 \)

\[
\begin{align*}
s_0 &= 1 \quad 0 \quad 0 \\
s_1 &= 0 \quad (-r_0) \quad (-r_1) \\
s_2 &= 0 \quad 1 \quad -(r_1 + r_2) \\
s_3 &= 0 \quad 0 \quad 1 \\
&\quad \frac{d_0 \cdot g_0 - d_1 \cdot g_1}{r_1 - r_0} \\
&\quad \frac{D(r_1)G(r_1) - d_1 \cdot g_{r_1}}{r_1 - r_2}
\end{align*}
\]

(9)

(iii) \( 3 \times 3 \) convolution, \( k = N + M - 3 = 3 \)

\[
\begin{align*}
s_0 &= 1 \quad 0 \quad 0 \\
s_1 &= 0 \quad r_0 \quad r_0 \\
s_2 &= 0 \quad (r_0 + r_1) \quad -(r_1 + r_2) \\
s_3 &= 0 \quad 1 \quad 1 \\
s_4 &= 0 \quad 0 \quad 0 \\
&\quad \frac{d_0 \cdot g_0 - d_1 \cdot g_1}{r_1 - r_0} \\
&\quad \frac{D(r_0)G(r_0) - d_1 \cdot g_{r_0}}{r_1 - r_2} \\
&\quad \frac{D(r_1)G(r_1) - d_1 \cdot g_{r_1}}{r_1 - r_2}
\end{align*}
\]

(10)

IV) \( 3 \times 4 \) convolution, \( k = N + M - 3 = 4 \)

\[
\begin{align*}
s_0 &= 1 \quad 0 \quad 0 \\
s_1 &= 0 \quad 0 \quad 0 \\
s_2 &= 0 \quad r_0 \quad r_0 \\
s_3 &= 0 \quad (r_0 + r_1) \quad -(r_1 + r_2) \\
s_4 &= 0 \quad 1 \quad 1 \\
s_5 &= 0 \quad 0 \quad 0 \\
&\quad \frac{d_0 \cdot g_0 - d_1 \cdot g_1}{r_1 - r_0} \\
&\quad \frac{D(r_0)G(r_0) - d_1 \cdot g_{r_0}}{r_1 - r_2} \\
&\quad \frac{D(r_1)G(r_1) - d_1 \cdot g_{r_1}}{r_1 - r_2}
\end{align*}
\]

(V) \( 4 \times 4 \) convolution, \( k = N + M - 3 = 5 \)

\[
\begin{align*}
s_0 &= 1 \quad 0 \quad 0 \\
s_1 &= 0 \quad 0 \quad 0 \\
s_2 &= 0 \quad r_0 \quad r_0 \\
s_3 &= 0 \quad (r_0 + r_1) \quad -(r_1 + r_2) \\
s_4 &= 0 \quad 1 \quad 1 \\
s_5 &= 0 \quad 0 \quad 0 \\
&\quad \frac{d_0 \cdot g_0 - d_1 \cdot g_1}{r_1 - r_0} \\
&\quad \frac{D(r_0)G(r_0) - d_1 \cdot g_{r_0}}{r_1 - r_2} \\
&\quad \frac{D(r_1)G(r_1) - d_1 \cdot g_{r_1}}{r_1 - r_2}
\end{align*}
\]

§3.2 Relationship with Previous Algorithms

Compared to the general algorithm shown in Formula (2), the algorithm based on Formula (7) has fewer operations due to the fact that the numbers in the first and the last row, and the first column in the matrix of Formula (7) are 0.

By assigning the numbers \( r_1, \ldots, r_k \) to certain values, the matrix derived based on (8)-(10) share a close relationship with the matrix in (4)-(6) derived by early algorithms.

(i) \( 2 \times 2 \) convolution, \( k = N + M - 3 = 1 \), \( r_1 = -1 \).

\[
\begin{align*}
s_0 &= 1 \quad 0 \quad 0 \\
s_1 &= 0 \quad 1 \quad -r_1 \\
s_2 &= 0 \quad 0 \quad 1 \\
&\quad \frac{d_0 \cdot g_0 - d_1 \cdot g_1}{r_1 - r_0}
\end{align*}
\]

(ii) \( 2 \times 3 \) convolution, \( k = N + M - 3 = 2 \), \( r_1 = -1 \), \( r_2 = 1 \).

\[
\begin{align*}
s_0 &= 1 \quad 0 \quad 0 \\
s_1 &= 0 \quad 1 \quad -r_1 \\
s_2 &= 0 \quad 0 \quad 1 \\
s_3 &= 0 \quad 0 \quad 0 \\
&\quad \frac{d_0 \cdot g_0 - d_1 \cdot g_1}{r_1 - r_0} \\
&\quad \frac{D(r_1)G(r_1) - d_1 \cdot g_{r_1}}{r_1 - r_2} \\
&\quad \frac{D(r_2)G(r_2) - d_1 \cdot g_{r_2}}{r_1 - r_2}
\end{align*}
\]

(iii) \( 3 \times 3 \) convolution, \( k = N + M - 3 = 3 \), \( r_1 = -1 \), \( r_2 = 1 \), \( r_3 = 2 \).

\[
\begin{align*}
s_0 &= 1 \quad 0 \quad 0 \\
s_1 &= 0 \quad 2 \quad -1 \\
s_2 &= 0 \quad -1 \quad 2 \\
s_3 &= 0 \quad 1 \quad 1 \\
&\quad \frac{d_0 \cdot g_0 - d_1 \cdot g_1}{r_1 - r_0} \\
&\quad \frac{d_2 \cdot g_2}{g_2 d_2}
\end{align*}
\]

On the other hand, the new general Formula (7) and the specific Formulae (8)-(10) point to many different alternatives for the choice of the \( r_1, \ldots, r_k \) which lead to the least operation counts. For example, based on Formula (8), we know that \( r_1 = -1 \) is not the only choice that needs 3 additions. If we choose \( r_1 = 1 \), then we have the following calculation requiring only 3 additions as well. In general, we can choose \( r_1, \ldots, r_k \) so that the number of 0’s in the matrix of Formula (7) is maximized.
\[
\begin{pmatrix}
1 & 0 & 0 & d_{4n} \\
0 & 1 & -\gamma & d_{4n} + d_{4n}\eta + d_{4n}
\end{pmatrix}
\begin{pmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & -\gamma & d_{4n} + d_{4n}\eta + d_{4n}
\end{pmatrix}
\]

We define the computation of two numbers \( d \ast g \) to be a multiplication if both factors can take on arbitrary real values, but it is not a multiplication if one of the factors can only be a rational number [page 99, [2]]. It is well known that for convolution based on Cook-Toom algorithm \( M + N - 1 = L \), multiplications are needed. The same result holds for Formula (7).

There is no general formula available to calculate the number of additions accurately. For specific short length linear convolutions of size \( 2 \times 2 \), \( 2 \times 3 \), \( 3 \times 3 \) algorithms, the number of additions needed are 3, 7, and 20 respectively [3, page 83].

There are two kinds of additions - pre-additions and post-additions. Pre-additions are used to derive \( D(t) \), while the post-additions are to derive \( s \) after \( D(t) \ast G(t) \) are known. Traditionally, multiplications by rational numbers are implemented by successive additions. The following from [2] is an example for \( 3 \times 3 \) convolution, where \( 2S_0 \) is counted as one addition.

\[
\begin{pmatrix}
2 & 0 & 0 & 0 & 0 \\
1 & -2 & -2 & -1 & 2 \\
-2 & 1 & 3 & 0 & -1 \\
1 & -1 & -1 & 1 & -2 \\
0 & 0 & 0 & 0 & 1
\end{pmatrix}
\begin{pmatrix}
S_0 \\
S_1 \\
S_2 \\
S_3 \\
S_4
\end{pmatrix}
\]

This way of counting is not correct for VLSI implementation of convolution algorithms since \( 2S_0 \) is available by simply shifting the input one bit position. If a number \( 2S_0 \) is counted as available, then the additions needed for short length linear convolutions can be reduced to 6 and \( 5x3+8=16 \) additions for \( 2 \times 3 \) and \( 3 \times 3 \) linear convolution respectively, based on the Formula (9) and (10).

**§ IV Conclusions**

A general algorithm for linear convolution is presented which needs interpolation at \( N + M - 3 \) points rather than \( N + M - 1 \) points by the conventional Cook-Toom algorithm. The algorithm is given in an explicit formula, which can provide more choices for the selection of the \( N + M - 3 \) points. The operation count for short-length convolutions can be reduced if they are implemented in VLSI circuits comparing to software implementation.

**References**


