Modularised fast polynomial transform algorithms for two-dimensional digital signal processing

J.-L. Wu
Y.-M. Huang

Indexing terms: Algorithms, Signal processing

Abstract: Novel, modularised, fast polynomial transform (FPT) algorithms for computing 2-D convolutions and 2-D discrete Fourier transforms are presented. In these new methods, only fast polynomial transforms, fast Fourier transforms and number theoretic transforms of the same length are required. Consequently the modularity and regularity of the proposed algorithms make them of great practical value in the concurrent implementation of the parallel-pipeline FPT for 2-D digital signal-processing tasks.

1 Introduction

Two-dimensional (2-D) digital convolution has many applications in digital signal processing, particularly in the processing of image, array and seismic signals. The 2-D digital convolution is conventionally evaluated using discrete transform methods, such as fast Fourier transforms (FFTs) and number theoretic transforms (NTTs) in column–row fashion. These discrete transform methods suffer from disadvantages, such as an excessive number of multiplications in the FFT algorithms and limitations on the convolution length and on the dynamic range in the NTTs [1].

Polynomial transforms were first developed by Nussbaumer [2] to map efficiently 2-D convolutions into 1-D convolutions and polynomial products. Since all the operations of the polynomial transforms are defined in the cyclotomic factors of the polynomial $Z^N - 1$, the 2-D convolutions can be calculated using ordinary arithmetic without multiplication, matrix transposition, dynamic range limitation and round-off error. Arambepola and Rayner [3] introduced the concept of complex mapping to transform a circular convolution into a skew-circular one and vice versa. This enables the series of polynomial transforms to be replaced by a single polynomial transform modulo $Z^N + 1$. Based on the ideas in References 2 and 3, Truong *et al.* [4] have shown that a combination of a fast polynomial transform (FPT) and the Chinese remainder theorem (CRT) can be used to compute a 2-D convolution of two $d_1 \times d_2$ complex arrays very efficiently, where $d_2 = 2^m$ and $d_1 = 2^{m-r+1}$ for $1 \leqslant r \leqslant m$. In this approach, the calculation of polynomial products is reduced to the computation of a 1-D circular convolution of length 2^n, $m - 1 \leqslant n \leqslant m - r$. Thus one has to use FFTs and FPTs with different lengths. In other words, registers, adders and multipliers of different word-lengths are required throughout the FPT stages. This length variation effect complicates the design and control of the concurrent (parallel-pipeline) FPT.

Pitas and Strintzis [5] have presented an M-D cyclic convolution algorithm, based on the factorisation of multivariable polynomial rings, which achieves the theoretically minimum number of multiplications. However, the shortcoming of length variation still exists in this approach.

Another extensively investigated area in the field of polynomial transforms is the mapping of M-D DFTs into 1-D DFTs [6–8]. This mapping is very efficient because it is accomplished using ordinary arithmetic without multiplication. However, FFTs and FPTs with different lengths have to be used because this approach is also defined in the $Z^N - 1$ polynomial ring.

Truong *et al.* [9] have recently proposed a modularised FPT structure for computing 2-D cyclic convolutions. In this new method, the 1-D cyclic polynomial convolution is decomposed into several cyclic convolutions of polynomials, all of the same length. The regularity and modularity of the new algorithm make it of great practical value in concurrent implementation of the FPT.

In this paper, we investigate and generalise the modularised FPT algorithm proposed in Reference 9 to deal with the two most basic operations of 2-D digital signal processing, namely the 2-D DFT and the 2-D digital convolution, and show that the resulting algorithms are more naturally suitable for multiprocessor and VLSI implementations.

2 Modularised FPT algorithm

The notation and definitions used in Reference 9 are adopted throughout this paper. The major contribution of Reference 9 comes from the following new decomposition of $Z^{d_2} - 1$:

$$Z^{d_2} - 1 = \prod_{k=0}^{2^r-1} (Z^{d_2/2^r} - \alpha^k) \tag{1}$$

where α is the complex 2^r-th root of unity and $d_2 = 2^m$. Based on eqn. 1, the cyclic 1-D polynomial convolution

$$C_{n_1}(Z) = \sum_{t_1=0}^{d_1-1} A_{t_1}(Z) B_{(n_1-t_1)}(Z) \bmod (Z^{d_2} - 1) \tag{2}$$

can be decomposed as

$$\begin{aligned} C^k_{n_1}(Z) &= C_{n_1}(Z) \bmod (Z^{d_2/2^r} - \alpha^k) \\ &= \sum_{t_1=0}^{d_1-1} A^k_{t_1}(Z) B^k_{(n_1-t_1)}(Z) \bmod (Z^{d_2/2^r} - \alpha^k) \end{aligned} \tag{3}$$

Paper 7342F (E5), first received 13th July 1989 and in final revised form 10th April 1990

The authors are with the Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan 10764, Republic of China

where $A^k_{t_1}(Z)$ and $B^k_{t_1}(Z)$ are also defined in the subring mod $(Z^{d_2/2^r} - \alpha^k)$. With the aid of the complex mapping introduced in Reference 3, eqn. 3 can be transformed into

$$\bar{C}^k_{n_1}(X) = \sum_{t_1=0}^{d_1-1} \bar{A}^k_{t_1}(X)\bar{B}^k_{(n_1-t_1)}(X) \bmod (X^{d_2/2^r} + 1) \quad (4)$$

where $Z = \beta_k X$ and β_k^{-1} is the $d_2/2^r$-th root of $(-\alpha^k)$. It is shown in Reference 2 that, since $X^{d_2/2^r} + 1$ is a cyclotomic polynomial, eqn. 4 can be computed using an FPT with X as its kernel, that is

$$\tilde{C}^k_{t_1}(X) = \tilde{A}^k_{t_1}(X)\tilde{B}^k_{t_1}(X) \bmod (X^{d_2/2^r} + 1) \quad (5)$$

where $\tilde{A}^k_{t_1}(X)$, $\tilde{B}^k_{t_1}(X)$, $\tilde{C}^k_{t_1}(X)$ are the FPTs of $\bar{A}^k_{t_1}(X)$, $\bar{B}^k_{t_1}(X)$ and $\bar{C}^k_{t_1}(X)$, respectively. Finally, by an application of the CRT, $C_{n_1}(Z)$ can be reconstructed from $C^k_{n_1}(Z)$. The polynomial products in eqn. 5 can be viewed as $2^r - 1$ skew-circular convolutions of length $d_2/2^r$ and can be computed very efficiently using the generalised FFT algorithms developed in References 10 and 11. More importantly, the equal length requirement of eqn. 5 forms the main theme of the proposed modularised FPT algorithm. The FFT-like recursive equations for the modulus reduction and the CRT reconstruction, as given in Reference 9, reduce the communication costs of the modularised FPT to an acceptable level. The overall cost (computation cost plus communication cost) of the modularised FPT is comparable with that of the row–column discrete transform approach or even with that of the conventional FPT algorithm proposed in Reference 4. However, the modularity and regularity of the modularised FPT are believed to be more naturally suitable for parallel-pipeline implementation.

3 Finite-field FPT algorithm with modularised structure

In this Section, we shall be concerned with the conditions necessary for the existence of the finite-field modularised FPT. Instead of factorising the polynomial $Z^{d_2} - 1$ in the complex field, the following finite-field factorisation is considered:

$$\langle Z^{d_2} - 1\rangle_F = \prod_{k=0}^{2^r-1} \langle (Z^{d_2/2^r} - \tilde{\alpha}^k)\rangle_F \quad (6)$$

where $\langle f(Z)\rangle_F$ denotes the coefficients of the polynomial $f(Z)$ defined in the field F and $\tilde{\alpha}$ is the 2^r-th root of unity in F. Since the FPTs are defined in the cyclotomic polynomial factors of $Z^{d_2} - 1$, each factor polynomial $\langle (Z^{d_2/2^r} - \tilde{\alpha}^k)\rangle_F$ has to be converted to a cyclotomic polynomial $(X^{d_2/2^r} + 1)$ through the mapping $Z = \beta_{1k} X$ where

$$\begin{aligned} \beta_{1k} &= (-\tilde{\alpha}^k)^{2^r/d_2} \\ &= (-1)^{2^r/d_2}(\tilde{\alpha}^{2^r/d_2})^k \\ &= bg^k \end{aligned} \quad (7)$$

with

$$b^{d_2/2^r} = -1 \quad \text{(i.e. } b \text{ is the } d_2/2^r\text{-th root of } -1 \text{ in } F)$$

and

$$g^{d_2} = \tilde{\alpha}^{2^r} = 1 \quad \text{(i.e. } g \text{ is the } d_2\text{-th root of 1 in } F) \quad (8)$$

Further, the complex multiplications used in the modulus reduction [9, eqn. 8] and in the CRT reconstruction [9, eqn. 9] are the 2^v-th root of 1 in F, say $\tilde{\alpha}_v$ (where $1 \leqslant v \leqslant r$, $0 \leqslant k \leqslant 2^r - 1$), and its inverse, respectively. Because $d_2 = 2^m$, $m \geqslant r$, it is clear that the 2^v-th root of unity is in F provided that g is in F. Thus, the modularised FPT algorithm described in Section 2, can be directly extended to the finite-field polynomial ring $\langle Z^{d_2} - 1\rangle_F$ under the condition that the d_2-th root of unity exists in F. More interestingly, the computation of the skew-circular convolution can be realised very efficiently by using the combined techniques of the polynomial transform and the NTT as proposed in References 12 and 13. The flowchart of the newly proposed algorithm is shown in Fig. 1.

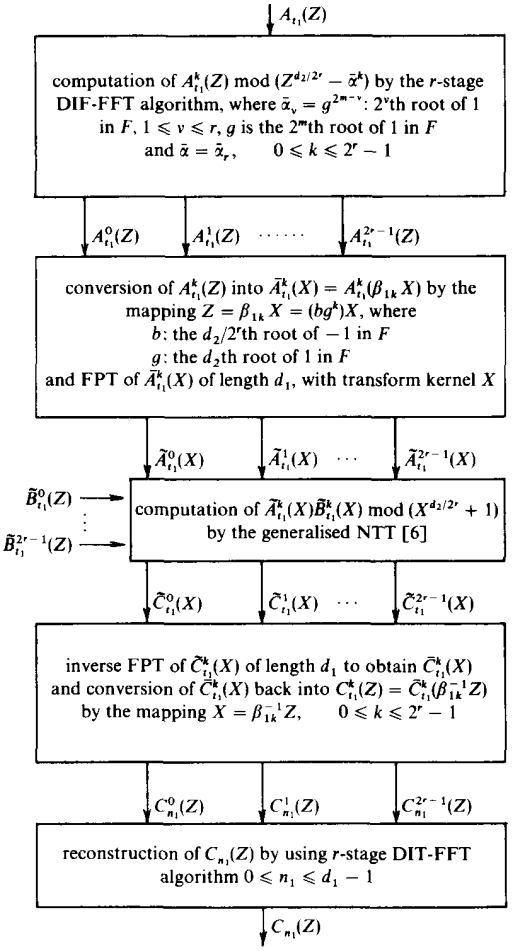

Fig. 1 *Computation of a 2-D cyclic convolution of dimension $d_1 \times d_2$, where $d_2 = 2^m$ and $d_1 = 2^{m-r+1}$, for $1 \leqslant r \leqslant m$*

To represent the convolution results unambiguously, the field F must be chosen such that

$$|F| \leqslant 2 \max(a_{t_1,t_2})\|b_{t_1,t_2}\| + 1 \quad (9)$$

where $|F|$ denotes the cardinality of F, $\max(a_{t_1,t_2})$ denotes the largest value of the 2-D array a_{t_1,t_2} and $\|b_{t_1,t_2}\|$ is the sum of the magnitudes of all the elements of b_{t_1,t_2}. Further, to guarantee the existence of the 2^m-th root of unity, good choices for F are those modulo Fermat prime $2^k + 1$ for $k = 2^n$, $n = 1, 2, 3, 4$, and those modulo primes of the form $3 \cdot 2^n + 1$ [14]. If g is a power of two then, with proper choice of F, the multiplications required in mapping, modulus reduction and CRT-

reconstruction can be replaced by bit shifts only. If the convolving sequences are 2-D Gaussian integer arrays, the proposed finite field algorithm can also be applied in the quadratic extension field of F, say F^2, and in this case the Mersenne primes are good candidates for F.

4 Two-variable FPT algorithm with modularised structure

In this Section, we combine the techniques proposed in Reference 5 and the modularised FPT algorithm to derive a novel modularised FPT algorithm for 2-D convolutions.

It is well known that the convolution of two $d_1 \times d_2$ complex sequences a_{t_1,t_2}, b_{t_1,t_2} can be represented in terms of the Z-transforms of the above sequences in a two-variable polynomial ring as

$$C(Z_1, Z_2) = A(Z_1, Z_2)B(Z_1, Z_2) \bmod Z_1^{d_1} - 1, Z_2^{d_2} - 1 \tag{10}$$

where $0 \leqslant t_i \leqslant d_i - 1$, $d_i = 2^{m_i}$ for $i = 1, 2$, and without loss of generality we assume $1 \leqslant m_1 \leqslant m_2$, $m_i \in Z^+$ (the set of positive integers). In a similar manner to Reference 3, one can rewrite eqn. 10 through the mapping $Z_1 = \beta_2 Z$, where β_2 is the d_1-th root of -1 in $\mathbb{C}$ (the complex field), as

$$C(Z, Z_2) = A(Z, Z_2)B(Z, Z_2) \bmod Z^{d_1} + 1, Z_2^{d_2} - 1 \tag{11}$$

Expr. 11 can now be regarded as the product of two single-variable, say Z, polynomials defined in the ring modulo $Z_2^{d_2} - 1$, and the coefficients of the polynomials are in the quotient field $\mathbb{C}[Z]/Z^{d_1} + 1$ [15]. Based on the ideas of Reference 5 and the equal length decomposition given in eqn. 1, $Z_2^{d_2} - 1$ is now decomposed, in $\mathbb{C}[Z]/Z^{d_1} + 1$, as

$$Z_2^{d_2} - 1 = \prod_{k=0}^{2^r - 1} [Z_2^{d_2/2^r} - (Z^{2^{m_1 - r + 1}})^k] \qquad 1 \leqslant r \leqslant \min(m_1 + 1, m_2) \tag{12}$$

where $\min(x, y)$ stands for the minimal value of x and y. Noting the decomposition of $Z_2^{d_2} - 1$ in eqn. 12, eqn. 11 becomes

$$\begin{aligned} C^k(Z, Z_2) &\equiv C(Z, Z_2) \bmod (Z^{d_1} + 1) \\ &\quad (Z_2^{d_2/2^r} - Z^{k2^{m_1 - r + 1}}) \\ &\equiv A^k(Z, Z_2)B^k(Z, Z_2) \bmod (Z^{d_1} + 1) \\ &\quad (Z_2^{d_2/2^r} - Z^{k2^{m_1 - r + 1}}) \end{aligned} \tag{13}$$

where $A^k(Z, Z_2)$ and $B^k(Z, Z_2)$ are $A(Z, Z_2)$ and $B(Z, Z_2) \bmod (Z^{d_1} + 1)$, $(Z_2^{d_2/2^r} - Z^{k2^{m_1 - r + 1}})$, respectively. Now using the mapping

$$Z_2 = \beta_3^k X \tag{14a}$$

with

$$\beta_3 = Z^{2^{m_1 - m_2 + 1}} \tag{14b}$$

eqn. (13) can be transformed into

$$\bar{C}^k(Z, X) \equiv \bar{A}^k(Z, X)\bar{B}^k(Z, X) \bmod (Z^{d_1} + 1), (X^{d_2/2^r} - 1) \tag{15}$$

where $\bar{A}^k(Z, X) = A^k(Z, \beta_3^k X)$, $\bar{B}^k(Z, X)$ and $\bar{C}^k(Z, X)$ are defined in the same way. The evaluation of eqn. 15 is greatly simplified by noticing that this expression can be viewed as the length-$d_2/2^r$ cyclic convolution of polynomials defined in the cyclotomic polynomial field mod $Z^{d_1} + 1$. It follows from the basic definition of the FPT that such a convolution of polynomials can be computed by using an FPT with $Z^{2^{m_1 - m_2 + r + 1}}$ as its kernel.

To clarify what has been done, in the rest of this Section the novel modularised FPT algorithm will be discussed in detail. Let

$$\left.\begin{aligned} \bar{A}^k(Z, X) &= \sum_{n=0}^{d_2/2^r - 1} \bar{A}_n^k(Z)X^n \\ \bar{B}^k(Z, X) &= \sum_{n=0}^{d_2/2^r - 1} \bar{B}_n^k(Z)X^n \\ \bar{C}^k(Z, X) &= \sum_{n=0}^{d_2/2^r - 1} \bar{C}_n^k(Z)X^n \end{aligned}\right\} \tag{16}$$

and the transforms of $\bar{A}_n^k(Z)$, $\bar{B}_n^k(Z)$ and $\bar{C}_n^k(Z)$ be $\tilde{A}_n^k(Z)$, $\tilde{B}_n^k(Z)$ and $\tilde{C}_n^k(Z)$, respectively. By the convolution theorem

$$\tilde{C}_n^k(Z) = \tilde{A}_n^k(Z)\tilde{B}_n^k(Z) \bmod Z^{d_1} + 1 \tag{17}$$

Hence, the inverse transform of $\tilde{C}_n^k(Z)$ is $\bar{C}_n^k(Z)$ and $C^k(Z, Z_2)$ can be obtained from $\bar{C}^k(Z, X)$ by changing the variable X back into $\beta_3^{-k} Z_2$, i.e. $C^k(Z, Z_2) = \bar{C}^k(Z, \beta_3^{-k} Z_2)$. $C(Z, Z_2)$ can be reconstructed from $C^k(Z, Z_2)$ using a FFT-like structure as described later, and the final convolution result $C(Z_1, Z_2)$ is obtained from $C(Z, Z_2)$ through the inverse mapping $Z = \beta_2^{-1} Z_1$. Before describing the modulus reduction and CRT reconstruction structure, let us first consider the mapping $Z_2 = \beta_3^k X = Z^{k2^{m_1 - m_2 + 1}} X$. If $m_1 - m_2 + 1 \geqslant 0$, the multiplication by β_3^k (or β_3^{-k}) for each polynomial amounts to $(k2^{m_1 - m_2 + 1})$-word right (or left) shifts of the polynomial followed by a sign inversion of the overflowed words. This gives a good situation for hardware implementation. For case where $m_1 - m_2 + 1 < 0$, the input 2-D arrays can be decomposed into several subarrays of reduced size by using conventional FPT approaches such that the condition $m_1 - m_2 + 1 \geqslant 0$ is satisfied for each subarray. The prescribed algorithm can then be applied to complete the computation.

We now return to the modulus reduction and the CRT reconstruction algorithms. From expr. 13, the modulus reduction of $A(Z, Z_2)$, $A^k(Z, Z_2)$ can be written as

$$\begin{aligned} A^k(Z, Z_2) &\equiv A(Z, Z_2) \bmod (Z^{d_1} + 1) \\ &\quad (Z_2^{d_2/2^r} - Z^{k2^{m_1 - r + 1}}) \\ &= \sum_{n=0}^{d_2/2^r - 1} a_n^k Z_2^n \end{aligned} \tag{18}$$

To compute a_n^k efficiently we define

$$\begin{aligned} A_v^k(Z, Z_2) &= A(Z, Z_2) \bmod (Z^{d_1} + 1) \\ &\quad (Z_2^{d_2/2^v} - Z^{k2^{m_1 - v + 1}}) \\ &= \sum_{n=0}^{d_2/2^v - 1} a_n^k(v) Z_2^n \end{aligned} \tag{19}$$

where $0 \leqslant v \leqslant r$ and $0 \leqslant 2^v - 1$. Notice that $A^k(Z, Z_2) = A_r^k(Z, Z_2)$ and $A(Z, Z_2) = A_0^0(Z, Z_2)$, i.e. $a_n^k = a_n^k(r)$ for $0 \leqslant n \leqslant d_2/2^r - 1$ and $0 \leqslant k \leqslant 2^r - 1$. Since $(Z^{2^{m_1 - v + 1}})^{(k + 2^{v-1})} = -Z^{k2^{m_1 - v + 1}}$ and $Z_2^{d_2/2^v} - Z^{k2^{m_1 - v + 1}}$ is a factor of $Z_2^{d_2/2^{v-1}} - Z^{k2^{m_1 - (v-1) + 1}}$, the coefficients $a_n^k(v)$ in eqn. 19 can be shown to satisfy the following recursive equations

$$a_n^k(v) = a_n^k(v - 1) + \text{shift-}R(k2^{m_1 - v + 1})[a_{n + d_2/2^v}^k(v - 1)] \tag{20a}$$

and

$$a_n^{k+2^{\nu-1}}(\nu) = a_n^k(\nu-1) - \text{shift-}R(k2^{m_1-\nu+1}) \times [a_{n+d_2/2^\nu}^k(\nu-1)] \quad (20b)$$

for $0 \leqslant k \leqslant 2^{\nu-1}-1$ and $0 \leqslant n \leqslant d_2/2^\nu - 1$, where the function shift-$R(x)(y)$ stands for x-word right shifts of the polynomial y followed by a sign inversion of the overflowed words. By virture of eqns. 20, $a_n^k(r)$ can be obtained by an iteration on $1 \leqslant \nu \leqslant r$ and this flow structure is similar to that of the decimation-in-time FFT.

The reconstruction of $C(Z, Z_2)$ from $C^k(Z, Z_2)$ is the inverse problem of modulus reduction discussed above. To accomplish this, we define

$$C(Z, Z_2) = \sum_{n=0}^{d_2-1} c_n Z_2^n = \sum_{n=0}^{d_2-1} \left[\sum_{m=0}^{d_1-1} c_{n,m} Z^m\right] Z_2^n \quad (21a)$$

and

$$C_\nu^k(Z, Z_2) \equiv C(Z, Z_2) \bmod (Z^{d_1}+1)(Z_2^{d_2/2^\nu} - Z^{k2^{m_1-\nu+1}}) = \sum_{n=0}^{d_2/2^\nu-1} c_n^k(\nu) Z_2^n \quad 1 \leqslant \nu \leqslant r,\ 0 \leqslant k \leqslant 2^\nu - 1 \quad (21b)$$

At this point, $c_n^k = c_n^k(r)$ are known and c_n are wanted. Note that $C_r^k(Z, Z_2) = C^k(Z, Z_2)$ and it can be shown that $c_n^k(\nu)$ satisfies the following two recursive equations:

$$c_n^k(\nu-1) = \tfrac{1}{2}[c_n^k(\nu) + c_n^{k+2^{\nu-1}}(\nu)] \quad (22a)$$

and

$$c_{n+d_2/2^\nu}^k(\nu-1) = \tfrac{1}{2}\text{shift-}L(k2^{m_1-\nu+1}) \times ([c_n^k(\nu) - c_n^{k+2^{\nu-1}}(\nu)]) \quad (22b)$$

where the function shift-$L(x)(y)$ stands for x-word left shifts of the polynomial y followed by a sign inversion of the overflowed words. By an iteration on $r \leqslant \nu \leqslant 1$, c_n $(0 \leqslant n \leqslant d_2 - 1)$ can be obtained from eqns. 22 and the flow structure is similar to that of the decimation-in-frequency FFT. The flowchart of the proposed algorithm is shown in Fig. 2. The advantage of this new approach, which is even better than the algorithm given in Section 2, is that in the new algorithm the multiplications required for complex mapping and the FFT-like modulus reduction–CRT reconstruction computations are replaced by cyclic word shifts. This fact, especially, meets the requirements of VLSI implementation.

5 FPT algorithm with modularised structure for computing the 2-D DFT

In this Section, the FPT algorithm for computing the 2-D DFT is modularised into identical modules. Without loss of generality, let us consider the 2-D DFT of the $d_1 \times d_2$ complex sequence x_{t_1,t_2} defined by

$$\bar{X}_{k_1,k_2} = \sum_{t_1=0}^{d_1-1}\sum_{t_2=0}^{d_2-1} x_{t_1,t_2} W_{d_1}^{t_1k_1} W_{d_2}^{t_2k_2} \quad (23)$$

where $d_i = 2^{m_i}$ and $W_{d_i} = e^{-j2\pi/d_i}$, for $i = 1, 2$ and $m_1 \leqslant m_2$.

To compute $\bar{X}_{k_1,k_2}$ by using the FPT, eqn. 23 is represented by the following three polynomial equations:

$$\bar{X}_{k_1}(Z) = \sum_{t_1=0}^{d_1-1} X_{t_1}(Z) W_{d_1}^{t_1k_1} \bmod (Z^{d_2}-1) \quad (24a)$$

$$X_{t_1}(Z) = \sum_{t_2=0}^{d_2-1} x_{t_1,t_2} Z^{t_2} \quad (24b)$$

$$\bar{X}_{k_1,k_2} = \bar{X}_{k_1}(Z) \bmod (Z - W_{d_2}^{k_2}) \quad (24c)$$

Based on eqn. 1, the index k_2 can be partitioned into 2^r parts as

$$k_2 = 2^r d + n \quad (25)$$

where $0 \leqslant 2^r - 1$ and $0 \leqslant d \leqslant d_2/2^r - 1$. It follows that

$$Z - W_{d_2}^{2^r d+n} \mid Z^{d_2/2^r} - \alpha^{2^r-n} \quad (26)$$

where '$a \mid b$' means a is divided by b. Now let

$$X_{t_1}^n(Z) = \sum_{t_2=0}^{d_2/2^r-1} x_{t_1,t_2}^n \equiv X_{t_1}(Z) \bmod (Z^{d_2/2^r} - \alpha^{2^r-n}) \quad (27)$$

Derivations and discussions of the FPT algorithm for 2-D DFTs are given in Reference 2 and are not repeated here. From Reference 2, eqns. 1 and 24–27 it follows that, for k_2 odd (i.e. $k_2 = 2^r d + n$ with n odd), eqn. 24 can be reformulated as

$$\bar{X}_{k_2k_1}^n(Z) = \sum_{t_1=0}^{d_1-1} X_{t_1}^n(Z)(Z^{d_2/d_1})^{t_1k_1} \bmod (Z^{d_2/2^r} - \alpha^{2^r-n}) \quad (28a)$$

$$\bar{X}_{k_2k_1,k_2} = \bar{X}_{k_2k_1}^n(Z) \bmod (Z - W_{d_2}^{k_2}) \quad (28b)$$

Note that, since k_2 and d_1 are coprime to each other, $k_2k_1 \bmod d_1$ is only a permutation of k_1. Now with $k_2 = 2^r d + n$ and letting

$$\bar{X}_{(2^rd+n)k_1}^n(Z) = \sum_{s=0}^{d_2/2^r-1} y_{k_1,s}^n Z^s \quad (29a)$$

then

$$\bar{X}_{(2^rd+n)k_1,(2^rd+n)} = \sum_{s=0}^{d_2/2^r-1} y_{k_1,s}^n W_{d_2}^{(2^rd+n)s} = \sum_{s=0}^{d_2/2^r-1} (y_{k_1,s}^n W_{d_2}^{sn}) W_{d_2/2^r}^{sd} \quad (29b)$$

Thus, for fixed k_1, eqn. 28b becomes $2^{r-1}d_2/2^r$-point 1-D DFTs as described in eqns. 29. Further, since k_2 is odd here, eqn. 29 can be computed very efficiently by using the generalised FFT algorithms. The calculation of remainders mod $(Z^{d_2/2^r} - \alpha^{2^r-n})$ required in eqns. 27 and 28a can be accomplished by a FFT-like structure as described in Reference 9.

Now, let us change our attention to the case of k_2 even, i.e. $k_2 = 2^r d + n$ with n even. Since

$$Z - W_{d_2}^{k_2+1} \mid Z^{d_2/2^r} - \alpha^{2^r-n-1} \quad (30)$$

then, in the same way as eqn. 28 was derived, eqn. 24 can also be reformulated for k_2 even as

$$\bar{X}_{(k_2+1)k_1}^n(Z) = \sum_{t_1=0}^{d_1-1} [X_{t_1}^{n'}(Z)](Z^{d_2/d_1})^{t_1k_1} \times \bmod (Z^{d_2/2^r} - \alpha^{2^r-n-1}) \quad (31a)$$

$$\bar{X}_{t_1}^{n'}(Z) = \sum_{t_2=0}^{d_2/2^r-1} [X_{t_1,t_2}^n W_{d_2}^{-t_2}] Z^{t_2} \times \bmod (Z^{d_2/2^r} - \alpha^{2^r-n-1}) \quad (31b)$$

$$\bar{X}_{(k_2+1)k_1,k_2} = \bar{X}_{(k_2+1)k_1}^n(Z) \bmod (Z - W_{d_2}^{k_2+1}) \quad (31c)$$

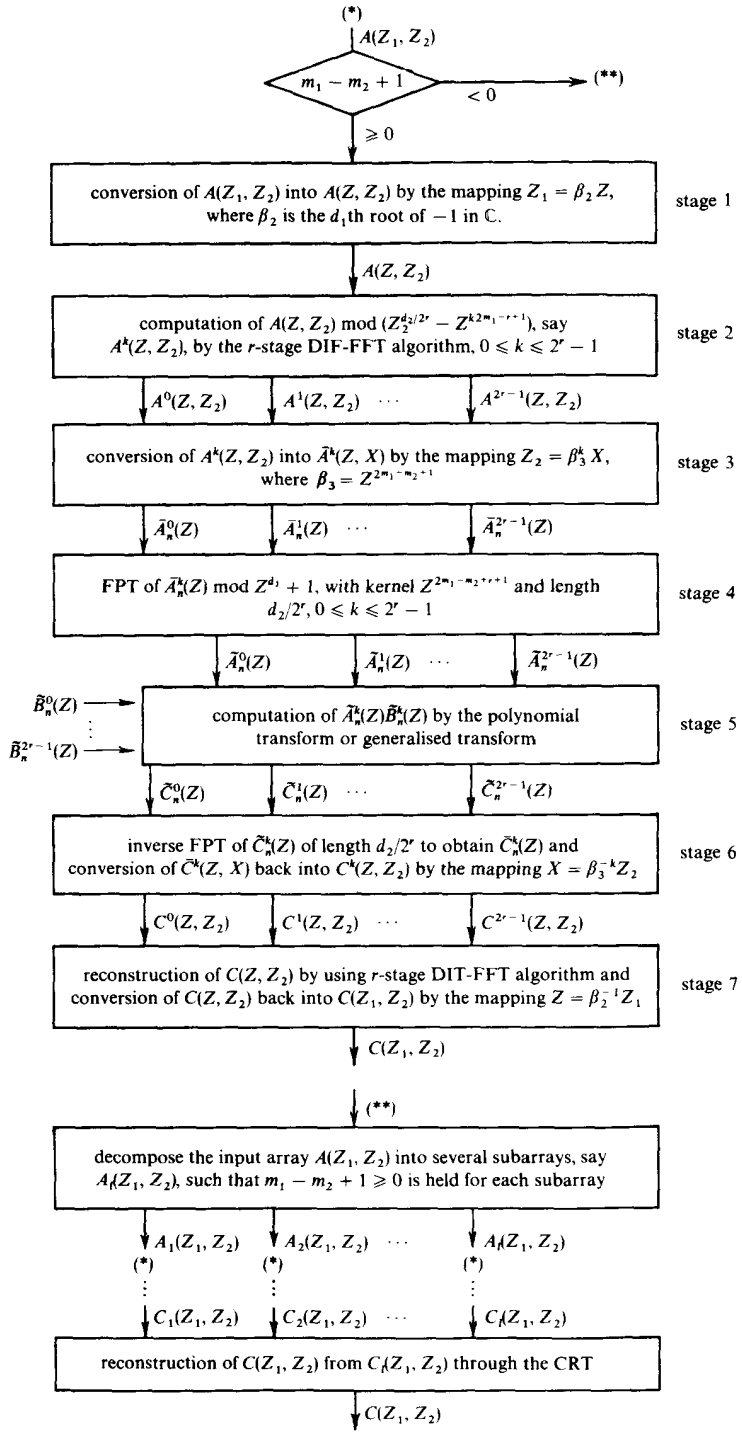

Fig. 2 *Computation of a 2-D cyclic convolution of dimension $d_1 \times d_2$, where $d_i = 2^{m_i}$, $i = 1, 2$, and $1 \leqslant m_1 < m_2$, $m_2 \in Z^+$*

In this case, it follows that

$$k_2 + 1 = 2^r d + n + 1 = 2^r d + n'$$

where $1 \leqslant n'$: odd $\leqslant 2^r - 1$ and

$$Z - W_{d_2}^{2^r d + n'} \mid Z^{d_2/2^r} - \alpha^{2^r - n'}$$

Hence by comparing eqn. 28 with eqn. 31, it is easy to conclude that, for k_2 even, the FPT algorithm for computing 2-D DFTs can also be modularised following the same procedures as for the case of k_2 odd. The flowchart of this new algorithm is shown in Fig. 3. It is obvious, from Fig. 3, that for the computation of a $d_1 \times d_2$ 2-D DFT, only FPTs of length d_1 and FFTs of length $d_2/2^r$ are required.

novel modularised FPT algorithm for 2-D digital convolution with even better performance than the original one. Finally, a modularised FPT algorithm for computing 2-D DFTs has also been presented. The common features of the proposed algorithms is that only FPTs and FFTs of the same length are required throughout the whole process. In other words, it is believed that the proposed algorithms possess the properties of regularity and modularity.

A major challenge in modern digital system design is the discovery of algorithms that are inherently capable of matching the advantages offered by VLSI technology and concurrent processing. Among the most important and highly desirable properties of a VLSI/concurrent algo-

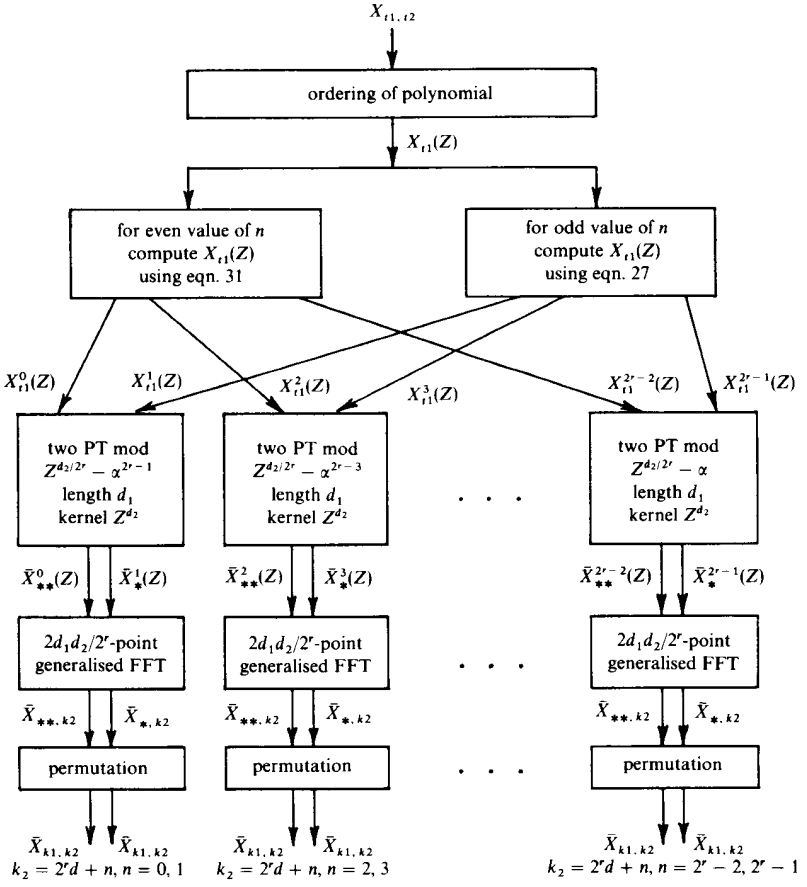

Fig. 3 *Computation of a 2-D DFT of dimension $d_1 \times d_2$, where $d_1 = 2^{m_1}$ and $d_2 = 2^{m_2}$, for $1 \leqslant m_1 \leqslant m_2$*
$* = k_1k_2$; $** = (k_2 + 1)k_1$

6 Conclusions and discussions

In this paper, modularised FPT algorithms for 2-D digital convolution and 2-D DFT are investigated in detail. Based on the ideas of Reference 9, finite-field arithmetic has been proposed for use with the modularised FPT. In this approach, the required operations are word shifts, bit shifts and real multiplications in the NTT domain only. We have then substituted the concept of equal-length decomposition into the multivariable FPT algorithm proposed in Reference 5 and we obtained a rithm are those of modularity and regularity. The modularised FPT algorithms proposed in this paper meet these requirements. All the proposed FPT algorithms described in Figs. 1–3 were written in C-programming language on an IBM PC-AT. These programs have been tested by several examples and verified to be correct.

7 Acknowledgment

The authors wish to thank the anonymous referees for their assistance in correcting this paper.

8 References

1 AGARWAL, R.C., and BURRUS, C.S.: 'Number theoretic transform to implement fast digital convolution', *Proc. IEEE*, 1975, **63**, pp. 550–560

2 NUSSBAUMER, H.J.: 'Digital filtering using polynomial transforms', *Electron. Lett.*, June 1977, **13**, pp. 386–387

3 ARAMBEPOLA, B., and RAYNER, P.J.W.: 'Efficient transforms for multidimensional convolutions', *Electron. Lett.*, 1979, **15**, (7), pp. 189–190

4 TRUONG, T.K., REED, I.S., LIPES, R.G., and WU, C.: 'On the application of a fast polynomial transform and the Chinese remainder theorem to compute a two-dimensional convolution', *IEEE Trans.*, February 1981, **ASSP-29**, (1), pp. 91–99

5 PITAS, I., and STRINTZIS, M.G.: 'Multidimensional cyclic convolution algorithms with minimal multiplicative complexity', *IEEE Trans.*, March 1987, **ASSP-35**, (3), pp. 384–390

6 NUSSBAUMER, H.J., and QUANDALLE, P.: 'Fast computation of discrete Fourier transform using polynomial transforms', *IEEE Trans.*, April 1979, **ASSP-27**, pp. 169–181

7 NUSSBAUMER, H.J.: 'New polynomial transform algorithms for multidimensional DFTs and convolutions', *IEEE Trans.*, February 1981, **ASSP-29**, (1), pp. 74–84

8 WU, JA-LING., and PEI, S.C.: 'Fast biased polynomial transforms for 2-D prime length DFTs', *Electron. Lett.*, October 1984, **20**, (22), pp. 933–934

9 TRUONG, T.K., *et al.*, 'An FPT algorithm with a modularized structure for computing 2-D cyclic convolutions', *IEEE Trans.*, September 1988, **ASSP-36**, pp. 1540–1542

10 CORSINI, P., and FROSINI, G.: 'Properties of the multidimensional Generalized discrete Fourier transform', *IEEE Trans.*, 1979, **C-28**, (11), pp. 819–830

11 REED, I.S., TRUONG, T.K., YEH, C.-S., and SHAO, H.M.: 'An improved FPT algorithm for computing two-dimensional cyclic convolutions', *IEEE Trans.*, August 1983 **ASSP-31**, pp. 1048–1050

12 MARTINELLI, G.: 'Long convolutions using number theoretic and polynomial transforms', *IEEE Trans.*, October 1984, **ASSP-32**, pp. 1090–1092

13 PEI, S.C., and WU, JA-LING: 'Improved long convolutions using generalized number theoretic and polynomial transforms', *IEEE Trans.*, December 1985, **ASSP-33**, pp. 1626–1627

14 GOLOMB, S.W., REED, I.S., and TRUONG, T.K.: 'Integer convolutions over the finite field GF($3 \cdot 2^n + 1$)', *SIAM J. Appl. Math.*, March 1977, **32**, pp. 357–365

15 GILBERT, J., and GILBERT, L.: 'Elements of modern algebra' (PWS-KENT Publishing, 1988)

16 MEAD, C., and CONWAY, L.: 'Introduction to VLSI systems' (Addison-Wesley, Reading, MA, 1980)

17 NUSSBAUMER, H.J.: 'Fast Fourier transform and convolution algorithms' (Springer-Verlag, 1981)

9 Appendix

9.1 Details of the proposed algorithm

The complexities of the proposed algorithms will be analysed in detail in this Appendix and a comparison made to other related works.

For simplicity and without loss of generality, the following assumptions have been made:

- $d_1 = d_2 = N$
- the cost of forward FPT is equivalent to that of the inverse one
- the cost of an m-point polynomial addition is equivalent to that of m arithmetic additions
- the speed of an i-word shift of an m-point polynomial is equivalent to that of im-word shifts. (The validity of this assumption implies that the polynomial shifts are implemented by a 'barrel shifter' [16].)

For comparison purposes, the complexities of Nussbaumer's FPT [17, Fig. 6.6, p. 175] and the modularized FPT algorithm given in Reference 9 are also considered.

Since the algorithm given in Fig. 2 seems to be the most complicated, we shall start our analysis with the two-variable modularised FPT algorithm given in Section 4. As indicated in Fig. 2, there are seven stages in this algorithm and the complexities (including multiplications, additions and shifts) for each stage are:

(a) stage 1: $2N^2$ multiplications (forward mapping)

(b) stage 2: (polynomial reduction in eqn. 20)

$2(1/2)(2^r)\log_2(2^r)$ $N^2/2^r$ word shifts

$2(2^r)\log_2(2^r)$ $N^2/2^r$-point polynomial additions

(c) stage 3: (the mapping in eqn. 14)

$2(2^r)$ $N^2/2^r$ word shifts

(d) stage 4: (forward FPT)

$2(2^r)$ FPTs of size $(N * N/2^r)$

(e) stage 5: (skew-circular convolution)

$2^r N[3\{1/2(N/2^r)\log_2(N/2^r)\} + N/2^r]$ multiplications

$2^r N[3(N/2^r)\log_2(N/2^r)]$ additions

(f) stage 6: (inverse FPT and the inverse mapping of stage 3)

2^r FPTs of size $(N * N/2^r)$

2^r $N^2/2^r$ word shifts

(g) stage 7: (CRT reconstruction and the inverse mapping of stage 1)

$(1/2)(2^r)\log_2(2^r)$ $N^2/2^r$ word shifts

$(2^r)\log_2(2^r)$ $N^2/2^r$-point polynomial additions

N^2 multiplications

Thus, the complexities of this algorithm become:

$$\left.\begin{array}{ll} \text{number of multiplications:} & 4N^2 + (3/2)N^2 \times [\log_2 N - r] \\ \text{number of additions:} & 3N^2 \log_2 N \\ \text{number of shifts:} & (3/2)rN^2 + 3N^2 \\ 3 \times 2^r \text{ FPTs of size } N * N/2^r & \end{array}\right\} \quad (32)$$

Following the same approach, the complexities of the finite-field modularised FPT algorithm given in Fig. 1 become:

$$\left.\begin{array}{ll} \text{number of multiplications:} & N^2 \\ \text{number of additions:} & 3N^2 \log_2 N \\ \text{number of shifts:} & 3N^2 + (3/2)N^2 \times \log_2 N \\ 3 \times 2^r \text{ FPTs of size } N * N/2^r & \end{array}\right\} \quad (33)$$

The complexities of Nussbaumer's FPT become:

$$\left.\begin{array}{ll} \text{number of multiplications:} & 4N^2 + (3/2)N^2 \times \log_2 N \\ \text{number of additions:} & 3N^2 \log_2 N \\ 3 \text{ FPTs of size } N * N & \end{array}\right\} \quad (34)$$

The complexities of the modularised FPT given in Reference 9 become:

$$\left.\begin{array}{ll} \text{number of multiplications:} & 4N^2 + (3/2)N^2 \times \log_2 N \\ \text{number of additions:} & 3N^2 \log_2 N \\ 3 \times 2^r \text{ FPTs of size } N * N/2^r & \end{array}\right\} \quad (35)$$

From exprs. 32 and 34 it follows that (in the case of $r = 1$) the algorithm given in Fig. 2 is more efficient than Nussbaumer's FPT, even in the sequential implementation environment, if

(i) the cost of $(3/2)N^2$ multiplications is higher than that of $3N^2 + (3/2)N^2$ shifts

(ii) the cost of a FPT of size $N * N$ is higher than that of two FPTs of size $(N * N/2)$.

The validity of (i) depends on the implementation technology and is always true in general purpose machines. Condition (ii) can be verified with the aid of Table 1 in Reference 11.

From exprs. 32 and 35, it follows that the algorithm given in Fig. 2 is more efficient than that in Reference 9 if condition (i) is valid.

From exprs. 33 and 35, it is clear that the finite-field modularised FPT is always more efficient than its complex field counterpart because the cost of the shift is always less than that of the multiplication.

Now let us consider the modularised FPT algorithm for computing DFTs. For comparison purposes, the complexities of the Nussbaumer's approach [17, Fig. 75, p. 200] are also considered. Without loss of generality, let us consider the algorithm given in Fig. 3 for $r = 2$. In this case, no multiplications are required in the polynomial reduction and polynomial transform stages. The number of multiplications required for complex mapping is the same for the two algorithms. Thus the only different between the two, in terms of multiplications, is that the algorithm in Fig. 3 requires $4N$ $N/4$-point generalised FFTs whereas Nussbaumer's algorithm requires $2N$ $N/2$-point odd-frequency DFTs [17]. It is easy to verify that the proposed algorithm needs $N^2/2$ multiplications fewer than that of Nussbaumer's approach.

9.2 Example of 2 * 4 cyclic convolution

A concrete example of a 2 * 4 cyclic convolution computed by the combined algorithms of Section 3 and 4 is given in this Appendix to clarify the operation of the algorithms. Consider the following 2-D cyclic convolution.

$$\begin{bmatrix} 2 & 1 & 5 & 2 \\ 3 & 4 & 6 & 7 \end{bmatrix} (*2) \begin{bmatrix} 2 & 4 & 2 & 3 \\ 1 & 3 & 2 & 5 \end{bmatrix} \tag{36}$$

where '*2' denotes the operation of 2-D cyclic convolution.

Form eqns. 6 and 10, eqn. 36 can be represented in finite-field two-variable polynomial form as

$$\langle C(Z_1, Z_2)\rangle_F = \langle A(Z_1, Z_2)B(Z_1, Z_2)\rangle_F \times \bmod (Z_1^2 - 1), (Z_2^4 - 1) \tag{37}$$

where

$$\langle A(Z_1, Z_2)\rangle_F = (2 + 3Z_1) + (1 + 4Z_1)Z_2 + (5 + 6Z_1)Z_2^2 + (2 + 7Z_1)Z_2^3$$

$$\langle B(Z_1, Z_2)\rangle_F = (2 + Z_1) + (4 + 3Z_1)Z_2 + (2 + 2Z_1)Z_2^2 + (3 + 5Z_1)Z_2^3$$

and the field F is chosen to be integer modulo P with $P = 65537 > 2(7)(22) + 1$.

Let $Z_1 = \alpha Z$ and $\alpha = 2^8 = \sqrt{(-1)} \pmod{65537}$, then eqn. 36 can be rewritten as

$$\langle C(Z_1, Z_2)\rangle_F = \langle A(Z, Z_2)B(Z_1, Z_2)\rangle_F \times \bmod (Z^2 + 1), (Z_2^4 - 1) \tag{38}$$

Without loss of generality, let us take $r = 2$ and factorise $z_2^4 - 1$ into

$$Z_2^4 - 1 = \prod_{k=0}^{3} (Z_2 - \alpha^k) \tag{39}$$

From the decomposition algorithm given in eqns. 20, $A^k(Z, Z_2)$ and $B^k(Z, Z_2)$ can be found from the butterfly-like flow structures, shown in Figs. 4 and 5 respectively. The results are:

$$A^0(Z, Z_2) = \langle 10 + 20(2^8)Z\rangle_F$$

$$A^1(Z, Z_2) = \langle [-3 + 3(2^8)] + [-1 - 3(2^8)]Z\rangle_F$$

$$A^2(Z, Z_2) = \langle 4 - 2(2^8)Z\rangle_F$$

$$A^3(Z, Z_2) = \langle [-3 - 3(2^8)] + [1 - 3(2^8)]Z\rangle_F$$

$$B^0(Z, Z_2) = \langle 11 + 11(2^8)Z\rangle_F$$

$$B^1(Z, Z_2) = \langle 2(2^8) + [1 - (2^8)]Z\rangle_F$$

$$B^2(Z, Z_2) = \langle -3 - 5(2^8)Z\rangle_F$$

$$B^3(Z, Z_2) = \langle -2(2^8) + [-1 - (2^8)]Z\rangle_F$$

After completing the calculation of the polynomial products $A^k(Z, Z_2)B^k(Z, Z_2)$ modulo $(Z^2 + 1)$ by the use of polynomial transforms or the generalised transforms for skew-circular convolution [10–13], one obtains

$$C^0(Z, Z_2) = \langle 330 + 330(2^8)Z\rangle_F$$

$$C^1(Z, Z_2) = \langle [-2 - 4(2^8)] + [6 + 4(2^8)]Z\rangle_F$$

$$C^2(Z, Z_2) = \langle -2 - 14(2^8)Z\rangle_F$$

$$C^3(Z, Z_2) = \langle -2 + 4(2^8)] + [-6 + 4(2^8)]Z\rangle_F$$

From the reconstruction algorithm of eqns. 22, $C(Z, Z_2)$ can also be obtained from $C^k(Z, Z_2)$ through a butterfly-

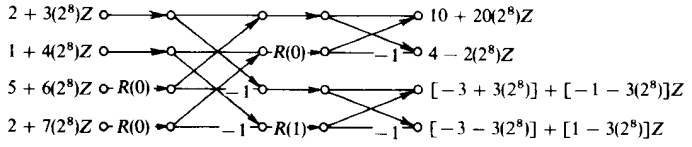

Fig. 4 *Computation of $A(Z, Z_2)$ mod $(Z^2 + 1)(Z_2 - Z^k)$, say $A^k(Z, Z_2)$*

R(i) stands for *i*-word right-rotation of polynomials $(a + bz, a, b \in C)$, followed by a sign inversion of the overflow words

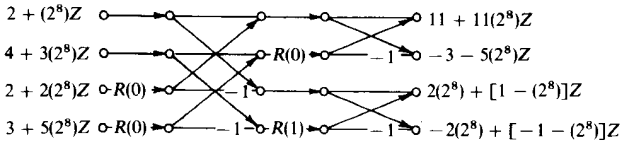

Fig. 5 *Computation of $B(Z, Z_2)$ mod $(Z^2 + 1)(Z_2 - Z^k)$, say $B^k(Z, Z_2)$*

R(i) stands for *i*-word right-rotation of polynomials $(a + bZ, a, b \in C)$, followed by a sign inversion of the overflow words

like flow structure, as shown in Fig. 6. The result is:

$$C(Z, Z_2) = \langle (81 + 81(2^8)z) + (86 + 88(2^8)Z)Z_2 + (83 + 77(2^8)Z)Z_2^2 + (80 + 84(2^8)Z)Z_2^3 \rangle_F$$

Performing the inverse mapping $Z = \alpha^{-1}Z_1 = 2^{-8}Z_1$, one obtains:

$$C(Z_1, Z_2) = \langle [81 + 81(2^8)Z_1] + [86 + 88(2^8)Z_1]Z_2 + [83 + 77(2^8)Z_1]Z_2^2 + [80 + 84(2^8)Z_1]Z_2^3 \rangle_F$$

That is

$$\begin{bmatrix} 2 & 1 & 5 & 2 \\ 3 & 4 & 6 & 7 \end{bmatrix} (*2) \begin{bmatrix} 2 & 4 & 2 & 3 \\ 1 & 3 & 2 & 5 \end{bmatrix} = \begin{bmatrix} 81 & 86 & 83 & 80 \\ 81 & 88 & 77 & 84 \end{bmatrix} \quad (40)$$

As shown in this example, there are no multiplications throughout the computation except for the skew-circular convolution and only FPTs, FFTs or NTTs, all of length two, are required.

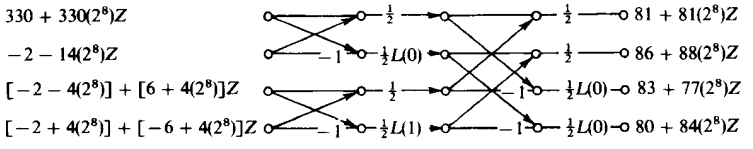

Fig. 6 *Reconstruction of $C(Z, Z_2)$ from $C^k(Z, Z_2)$*
L(i) stands for *i*-word left-rotation of polynomials ($a + bZ$, $a, b \in C$), followed by a sign inversion of the overflow words