[Journal Name], [Volumn Number], 1–12 ([Volumn Year])
© [Volumn Year] Kluwer Academic Publishers, Boston. Manufactured in The Netherlands.

A New Fast Discrete Fourier Transform*

FENG ZHOU
Dept. of Information and Electronic Engineering, Zhejiang University, Hangzhou, P.R.China

PETER KORNERUP
Dept. of Mathematics and Computer Science, Odense University, Odense, Denmark

;

Abstract. This paper presents a new fast Discrete Fourier Transform (DFT) algorithm. By rewriting the DFT, a new algorithm is obtained that uses $2^{n-2}(3n-13)+4n-2$ real multiplications and $2^{n-2}(7n-29)+6n+2$ real additions for a real data $N=2^n$ point DFT, comparable to the number of operations in the Split-Radix method, but with slightly fewer multiply and add operations in total. Because of the organization of multiplications as plane rotations in this DFT algorithm, it is possible to apply a pipelined CORDIC algorithm in a hardware implementation of a long-point DFT, e.g. at a 100MHz input rate, a 1024-point transform can be realized with a 200MHz clocking of a single CORDIC pipeline.

1. Introduction

The Discrete Fourier Transform (DFT) is a very useful algorithm, playing an important role in various digital signal processing applications from telecommunication, image processing, radar, sonar to vibrational analysis and material analysis and etc. According to the definition of DFT/IDFT, $4N^2$ real multiplications are needed for a N-point DFT/IDFT. In order to accelerate the DFT/IDFT processing, various algorithms for reducing the number of multiplications and additions have been reported during the past decades [1]-[8]. The traditional Fast Fourier Transform (FFT) is the most famous algorithm, using $2^{n+1}(n+1)$ real multiplications for $N=2^n$, and the FFCT [6] and Split-Radix [8] FFT algorithms, using only $2^{n-1}(n-3)+2$ multiplications and $2^{n-1}(3n-5)+4$ additions. In addition to software realizations of FFT, various hardware implementation [9]-[16] of FFT have been designed to meet the high speed or real-time requirements of DFT applications. In the new DFT algorithm presented in this paper, the number of real multiplications and real additions is respectively reduced to $2^{n-2}(3n-13)+4n-2$ and $2^{n-2}(7n-29)+6n+2$ for real data DFT. Compared to the Split-Radix [8] FFT algorithms, our new DFT algorithm needs slightly more real multiplications but fewer real additions, so that the total number of floating point operations is smaller. However, it has been the purpose to rewrite the DFT to express the computations in terms of plane rotations, to be able to apply the CORDIC algorithm. The number of such rotations needed is $2^{n-2}(n-4)+2(n-1)$, and the number of additions and multiplications quoted above corresponds to the implementation of each rotation as a complex multiplication.

The derivation of the method is quite involved and tedious, it is based on exploiting symmetries in the trigonometric factors of the expressions $A(k)=\sum_{n=0}^{N-1}a(n)cos\frac{2\pi}{N}kn$ and $B(k)=\sum_{n=0}^{N-1}b(n)sin\frac{2\pi}{N}kn$, allowing the sums to be split in halves and recombined recursively. The result-

*Supported by the Danish Research Councils, grant number 5.21.08.02

ing recursive equations are fairly simple, employing plane rotations applied to terms referenced by indices obtained by reversing certain bit patterns.

Because of the way multiplications and additions are organized as plane rotations in our DFT algorithm, it is possible efficiently to apply a pipelined CORDIC algorithm in a hardware implementation of the long-point DFT. In Section 2 we rewrite the DFT into the new algorithm. Section 3 discusses possible hardware implementations, and an appendix contains further details of the rewriting of the DFT.

2. A New Expression for DFT/IDFT

The $N = 2^n$ real data DFT/IDFT transformations are defined as follows:

$$\begin{aligned} F(k) &= \frac{1}{N}\sum_{n=0}^{N-1} f(n)e^{-j\frac{2\pi}{N}kn} \\ &= \frac{1}{N}\sum_{n=0}^{N-1} f(n)cos\frac{2\pi}{N}kn - j\frac{1}{N}\sum_{n=0}^{N-1} f(n)sin\frac{2\pi}{N}kn, \\ f(n) &= \sum_{k=0}^{N-1} F(k)e^{j\frac{2\pi}{N}kn} \\ &= \sum_{k=0}^{N-1} F_r(k)cos\frac{2\pi}{N}kn - \sum_{k=0}^{N-1} F_i(k)sin\frac{2\pi}{N}kn, \end{aligned}$$

where all $f(n)$ are real data and $F(k) = F_r(k) + jF_i(k), k, n = 0, 1, 2, \cdots, N-1$. For the computation of DFT/IDFT we shall only discuss the following two expressions

$$\begin{aligned} A(k) &= \sum_{n=0}^{N-1} a(n)cos\frac{2\pi}{N}kn, \\ B(k) &= \sum_{n=0}^{N-1} b(n)sin\frac{2\pi}{N}kn, \end{aligned}$$

for $k, n = 0, 1, 2, 3, \cdots, N-1$. Since $A(N-k) = A(k)$ and $B(N-k) = -B(k)$, k can be limited to the interval 0 to $N/2$. First we define four functions:

$$f_0^{\pm}(n) = a(n) \pm a(N-n), \quad g_0^{\pm}(n) = b(n) \pm b(N-n)$$

for $n = 1, 2, 3, \cdots, \frac{N}{2} - 1$, and using these we can split $A(k)$ and $B(k)$ $(k = 0, 1, 2, \cdots, N/2-1)$ twice into two parts,

$$\begin{aligned} A(k &= \sum_{n=1}^{\frac{N}{2}-1} [a(n) + a(N-n)]cos\frac{2\pi}{N}kn \\ &\quad + a(\frac{N}{2})cos\pi k + a(0) \\ &= \sum_{n=1}^{\frac{N}{4}-1} \left[f_0^+(n) + (-1)^k f_0^+(\frac{N}{2} - n)\right] cos\frac{2\pi}{N}kn \\ &\quad + f_0^+(\frac{N}{4})cos\frac{\pi k}{2} + a(\frac{N}{2})cos\pi k + a(0) \\ B(k) &= \sum_{n=1}^{\frac{N}{2}-1} [b(n) - b(N-n)]sin\frac{2\pi}{N}kn \\ &= \sum_{n=1}^{\frac{N}{4}-1} \left[g_0^-(n) - (-1)^k g_0^-(\frac{N}{2} - n)\right] sin\frac{2\pi}{N}kn \\ &\quad + g_0^-(\frac{N}{4})sin\frac{\pi k}{2}. \end{aligned}$$

To be able to continue the splitting we shall also introduce

$$f_L^{\pm}(0) = f_{\lfloor\frac{L}{2}\rfloor}^+(0) \pm f_{\lfloor\frac{L}{2}\rfloor}^+(\frac{N}{4L}), \quad f_0^{\pm}(0) = a(0) \pm a(\frac{N}{2})$$

for $L = 1, 2, 4, \cdots, \frac{N}{4}$ and also

$$\begin{aligned} f_L^{\pm}(n) &= f_{\lfloor\frac{L}{2}\rfloor}^+(n) \pm f_{\lfloor\frac{L}{2}\rfloor}^+(\frac{N}{2L} - n), \\ g_L^{\pm}(n) &= g_{\lfloor\frac{L}{2}\rfloor}^-(n) \pm g_{\lfloor\frac{L}{2}\rfloor}^-(\frac{N}{2L} - n) \end{aligned}$$

for $L = 1, 2, 4, \cdots, \frac{N}{8}, n = 1, 2, 3, \cdots, \frac{N}{4L} - 1$.

Finally, for $cos\frac{2k\pi}{N}$ and $sin\frac{2k\pi}{N}$, according to some trigonometric identities, we have

$$\begin{aligned} &cos\frac{2k\pi}{N}(\frac{N}{4L} - n) \\ &= \begin{cases} (-1)^{\frac{m}{2}} cos\frac{2\pi}{N}kn, k = mL, m = 0, 2, 4, 6, \cdots \\ (-1)^{\frac{m-1}{2}} sin\frac{2\pi}{N}kn, k = mL, m = 1, 3, 5, 7, \cdots \end{cases} \end{aligned}$$

and

$$\begin{aligned} &sin\frac{2k\pi}{N}(\frac{N}{4L} - n) \\ &= \begin{cases} -(-1)^{\frac{m}{2}} sin\frac{2\pi}{N}kn, k = mL, m = 0, 2, 4, 6, \cdots \\ (-1)^{\frac{m-1}{2}} cos\frac{2\pi}{N}kn, k = mL, m = 1, 3, 5, 7, \cdots \end{cases} \end{aligned}$$

where $L = 1, 2, 4, 8, \cdots$.

Now we can continue to split $A(k)$ and $B(k)$ using the above formulas. For $L = 1, 2, 4, \cdots, \frac{N}{16}, \frac{N}{8}$,

and the even multiples of L, $k = 2mL$, $m = 0, 1, 2, 3, \cdots, \frac{N}{4L}$, we find

$$A(k) = \quad (1)$$
$$\sum_{n=1}^{\frac{N}{8L}-1} \left[f_L^+(n) + (-1)^m f_L^+(\frac{N}{4L} - n) \right] cos(\frac{2\pi}{N} kn)$$
$$+(-1)^m f_{\lfloor \frac{L}{2} \rfloor}^+(\frac{N}{4L}) + f_L^+(\frac{N}{8L}) cos(\frac{k\pi}{4L}) + f_{\lfloor \frac{L}{2} \rfloor}^+(0)$$
$$B(k) = \quad (2)$$
$$\sum_{n=1}^{\frac{N}{8L}-1} \left[g_L^-(n) - (-1)^m g_L^-(\frac{N}{4L} - n) \right] sin(\frac{2\pi}{N} kn)$$
$$+g_L^-(\frac{N}{8L}) sin(\frac{k\pi}{4L}).$$

For the odd multiples of L, $k = (2m+1)L$, $m = 0, 1, 2, 3 \cdots, \frac{N}{4L} - 1$, we have

$$A(k) =$$
$$\sum_{n=1}^{\frac{N}{8L}-1} \left[f_L^-(n) cos(\frac{2\pi}{N} kn) + (-1)^m f_L^-(\frac{N}{4L} - n) sin(\frac{2\pi}{N} kn) \right]$$
$$+ f_L^-(\frac{N}{8L}) cos(\frac{k\pi}{4L}) + f_{\lfloor \frac{L}{2} \rfloor}^-(0),$$
$$B(k) =$$
$$\sum_{n=1}^{\frac{N}{8L}-1} \left[g_L^+(n) sin(\frac{2\pi}{N} kn) + (-1)^m g_L^+(\frac{N}{4L} - n) cos(\frac{2\pi}{N} kn) \right]$$
$$+g_L^+(\frac{N}{8L}) sin(\frac{k\pi}{4L}) + g_{\lfloor \frac{L}{2} \rfloor}^-(\frac{N}{4L}) sin(\frac{k\pi}{2L}).$$

For simplicity we let $S_k = (-1)^{\frac{k-L}{2L}} = (-1)^m$. For $k = (2m+1)L, m = 0, 1, 2, 3, \cdots, \frac{N}{4L} - 1$, $A(k), B(k)$ can be further split, but now we are using the following formulas to combine two parts of the split results:

$$X cos\frac{2k\pi}{N}(\frac{N}{Li} - n) + Y sin\frac{2k\pi}{N}(\frac{N}{Li} - n)$$
$$= \begin{bmatrix} cos\frac{2\pi}{N}kn & sin\frac{2\pi}{N}kn \end{bmatrix} \begin{bmatrix} cos\frac{2k\pi}{Li} & sin\frac{2k\pi}{Li} \\ sin\frac{2k\pi}{Li} & -cos\frac{2k\pi}{Li} \end{bmatrix} \begin{bmatrix} X \\ Y \end{bmatrix}$$
$$X sin\frac{2k\pi}{N}(\frac{N}{Li} - n) + Y cos\frac{2k\pi}{N}(\frac{N}{Li} - n)$$
$$= \begin{bmatrix} sin\frac{2\pi}{N}kn & cos\frac{2\pi}{N}kn \end{bmatrix} \begin{bmatrix} -cos\frac{2k\pi}{Li} & sin\frac{2k\pi}{Li} \\ sin\frac{2k\pi}{Li} & cos\frac{2k\pi}{Li} \end{bmatrix} \begin{bmatrix} X \\ Y \end{bmatrix}$$

Through some rewritings, details of which are shown in the appendix, we arrive at

$$A(k) =$$
$$\sum_{i=1,2,4,8,\cdots}^{N/16L} \left[P_L^i(\frac{N}{16Li}, k) cos(\frac{\pi k}{8Li}) + S_k P_L^i(\frac{3N}{16Li}, k) sin(\frac{\pi k}{8Li}) \right]$$
$$+ f_L^-(\frac{N}{8L}) cos(\frac{\pi k}{4L}) + f_{\lceil \frac{L}{2} \rceil}^-(0)$$
$$= P_L^{\frac{N}{8L}}(1, k),$$
$$B(k) =$$
$$\sum_{i=1,2,4,8,\cdots}^{N/16L} \left[Q_L^i(\frac{N}{16Li}, k) sin(\frac{\pi k}{8Li}) + S_k Q_L^i(\frac{3N}{16Li}, k) cos(\frac{\pi k}{8Li}) \right]$$
$$+ g_L^+(\frac{N}{8L}) sin(\frac{k\pi}{4L}) + g_{\lceil \frac{L}{2} \rceil}^-(\frac{N}{4L}) sin(\frac{k\pi}{2L})$$
$$= Q_L^{\frac{N}{8L}}(1, k),$$

where $P_L^i(n,k)$ and $Q_L^i(n,k)$ are recursively defined below in terms of $f_0^\pm(n)$ and $g_0^\pm(n)$. For notational convenience in these recursive definitions we replace $P_L^i(n,k)$ and $Q_L^i(n,k)$ by $p_L^i(n_k)$ and $q_L^i(n_k)$ for suitable indices n_k, defined using the binary representation of $k = \sum_{j=0}^{M-1} a_i 2^j L$ or $m = \sum_{j=0}^{M-2} b_j 2^j$

$$N_k = 2^{M-2}(1 - S_k) + S_k \sum_{j=1}^{M-1} a_j 2^{M-1-j}$$
$$= a_1 2^{M-2} + (-1)^{a_1} \sum_{j=1}^{M-1} a_j 2^{M-1-j} \quad (3)$$
$$n_k = N_k + S_k n, \text{ and } n_k' = N_k + S_k(\frac{N}{8Li} - n) \quad (4)$$

where $M = log_2 \frac{N}{2L}$ and $S_k = (-1)^m$ is also equal to $(-1)^{a_1} = (-1)^{b_0}$. Note that the binary representation of N_k contains a reversed bit pattern of that of k.

Therefore we can summarize this DFT algorithm as follows: For each value of $L = 1, 2, 4, ..., \frac{N}{8}$, we have

$$\begin{bmatrix} p_L^1(0) \\ p_L^1(\frac{N}{8L}) \end{bmatrix} = \begin{bmatrix} cos\frac{\pi}{4} & sin\frac{\pi}{4} \\ sin\frac{\pi}{4} & -cos\frac{\pi}{4} \end{bmatrix} \begin{bmatrix} 0 \\ f_L^-(\frac{N}{8L}) \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \end{bmatrix} f_{\lceil \frac{L}{2} \rceil}^-(0)$$
$$\begin{bmatrix} q_L^1(\frac{N}{4L}) \\ q_L^1(\frac{N}{8L}) \end{bmatrix} = \begin{bmatrix} -cos\frac{\pi}{4} & sin\frac{\pi}{4} \\ sin\frac{\pi}{4} & cos\frac{\pi}{4} \end{bmatrix} \begin{bmatrix} 0 \\ g_L^+(\frac{N}{8L}) \end{bmatrix} + \begin{bmatrix} 1 \\ -1 \end{bmatrix} g_{\lceil \frac{L}{2} \rceil}^-(\frac{N}{4L})$$

and for $L \leq \frac{N}{16}$ when $i = 1, 2, 4, 8, \cdots, \frac{N}{16L}$ and $k = (2m+1)L, m = 0, 1, 2, 3, \cdots, i-1$, for $n = 1, 2, 3, \cdots, \frac{N}{16Li} - 1$ the recursive equations are:

$$\begin{bmatrix} p_L^1(n) \\ p_L^1(\frac{N}{4L} - n) \end{bmatrix} = \begin{bmatrix} f_L^-(n) \\ f_L^-(\frac{N}{4L} - n) \end{bmatrix}$$
$$\begin{bmatrix} q_L^1(n) \\ q_L^1(\frac{N}{4L} - n) \end{bmatrix} = \begin{bmatrix} g_L^+(n) \\ g_L^+(\frac{N}{4L} - n) \end{bmatrix}$$

for $n = 1, 2, 3, \cdots, \frac{N}{8L} - 1$ and otherwise

$$\begin{bmatrix} p_L^{2i}(n_k) \\ p_L^{2i}(n'_k) \\ p_L^{2i}(\frac{N}{4L}\text{-}n'_k) \\ p_L^{2i}(\frac{N}{4L}\text{-}n_k) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & \text{-}1 & 0 & 0 \\ 0 & 0 & \text{-}1 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos\theta_k & sin\theta_k & 0 \\ 0 & sin\theta_k & -cos\theta_k & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_L^i(n_k) \\ p_L^i(n'_k) \\ p_L^i(\frac{N}{4L}\text{-}n'_k) \\ p_L^i(\frac{N}{4L}\text{-}n_k) \end{bmatrix}$$

$$\begin{bmatrix} q_L^{2i}(n_k) \\ q_L^{2i}(n'_k) \\ q_L^{2i}(\frac{N}{4L}\text{-}n'_k) \\ q_L^{2i}(\frac{N}{4L}\text{-}n_k) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & \text{-}1 & 0 & 0 \\ 0 & 0 & \text{-}1 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -cos\theta_k & sin\theta_k & 0 \\ 0 & sin\theta_k & cos\theta_k & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} q_L^i(n_k) \\ q_L^i(n'_k) \\ q_L^i(\frac{N}{4L}\text{-}n'_k) \\ q_L^i(\frac{N}{4L}\text{-}n_k) \end{bmatrix}$$

where n_k, n'_k are defined by (4) and $\theta_k = S_k \frac{k\pi}{4Li}$.

For the case $n{=}\frac{N}{16Li}$, let $n_k{=}N_k + S_k \frac{N}{16Li}$ with $N_k = \sum_{j=1}^{M-1} a_j 2^{M-1-j}$ and $n'_k{=}\frac{N}{4L} - n_k$, we have

$$\begin{bmatrix} p_L^{2i}(n'_k - \frac{N}{16Li}) \\ p_L^{2i}(n_k) \\ p_L^{2i}(n'_k) \\ p_L^{2i}(n_k - \frac{N}{16Li}) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & \text{-}1 & 0 & 0 \\ 0 & 0 & \text{-}1 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & sin\theta'_k & \text{-}cos\theta'_k & 0 \\ 0 & cos\theta'_k & sin\theta'_k & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_L^i(n'_k - \frac{N}{16Li}) \\ p_L^i(n_k) \\ p_L^i(n'_k) \\ p_L^i(n_k - \frac{N}{16Li}) \end{bmatrix}$$

$$\begin{bmatrix} q_L^{2i}(n'_k + \frac{N}{16Li}) \\ q_L^{2i}(n_k) \\ q_L^{2i}(n'_k) \\ q_L^{2i}(n_k + \frac{N}{16Li}) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ \text{-}1 & 1 & 0 & 0 \\ 0 & 0 & 1 & \text{-}1 \\ 0 & 0 & 1 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & sin\theta'_k & cos\theta'_k & 0 \\ 0 & cos\theta'_k & \text{-}sin\theta'_k & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} q_L^i(n'_k + \frac{N}{16Li}) \\ q_L^i(n_k) \\ q_L^i(n'_k) \\ q_L^i(n_k + \frac{N}{16Li}) \end{bmatrix}$$

where $\theta'_k = \frac{\pi}{4}(1 - S_k) + \frac{\theta_k}{2}$.

Finally for each L, we obtain the corresponding expressions for $A(k)$ and $B(k)$ for $k = L, 3L, 5L, \cdots, \frac{N}{2} - L$ as

$$A(k) = p_L^{\frac{N}{8L}}(N_k), \;\; B(k) = q_L^{\frac{N}{8L}}(\tfrac{N}{4L} - N_k). \qquad (5)$$

The above computations are iterated until $L = \frac{N}{8}$. For $L = \frac{N}{8}$, the remaining $A(k)$ and $B(k)$, $k = 0, 2L, 4L$, are determined from (1) and (2) as

$$\begin{bmatrix} A(0) \\ A(N/2) \\ A(N/4) \\ B(0) \\ B(N/2) \\ B(N/4) \end{bmatrix} = \begin{bmatrix} f^+_{\frac{N}{4}}(0) \\ f^-_{\frac{N}{4}}(0) \\ f^-_{\frac{N}{8}}(0) \\ 0 \\ 0 \\ g^-_{\frac{N}{16}}(1) \end{bmatrix}$$

Observe that these expressions containing trigonometric factors now are in the form of plane rotations, which can be realized by the CORDIC algorithm.

For the DFT $a(n) = b(n) = f(n)$ and $F(k)$ becomes

$$F(k) = A(k) - jB(k), \;\; F(N-k) = A(k) + jB(k)$$

and for the IDFT $a(k) = F_r(k), b(k) = F_i(k)$ and the real $f(n)$ becomes

$$f(n) = A(n) - B(n), \;\; f(N-n) = A(n) + B(n).$$

A diagram of the computations for $N = 64$ DFT using the above expressions is shown in Figure 1.

Now we can calculate how many multiplications and additions are needed for an $N = 2^n$ DFT/IDFT. For any value of L, the number of CORDIC operations for computing $A(k)$ is $1 + \frac{N}{16L} log_2 \frac{N}{8L}$. When L runs through the values $1, 2, 4, \cdots, N/8$, the total number of all CORDIC operations is then

$$\sum_L (1 + \frac{N}{16L} log_2 \frac{N}{16L}) = \frac{N}{8} log_2 \frac{N}{16} + log_2 \frac{N}{2}.$$

Note that the above number of CORDIC operations includes calculations for $f^-_L(\frac{N}{8L}) cos\frac{\pi}{4}$. The total number of CORDIC operations for calculating all $A(n)$ and $B(n)$ values can be split to find an equivalent number of additions and multiplications:

$$\begin{aligned} 2(\frac{N}{8} log_2 \frac{N}{16} + log_2 \frac{N}{2}) &= 2^{n-2}(n-4) + 2(n-1) \\ &= \left[2^{n-2}(n-5) + 2(n-1)\right] \\ &+ \left[2^{n-2} - 2(n-2)\right] + 2(n-2), \end{aligned}$$

in which each of $2(n-2)$ CORDIC operations computing $p_L^1(0)$ and $p_L^1(\frac{N}{8L})$ need 1 multiplication and 2 additions, each of $2^{n-2} + 2(2-n)$ CORDIC operations whose angle is just equal to $\frac{\pi}{4}$ need only

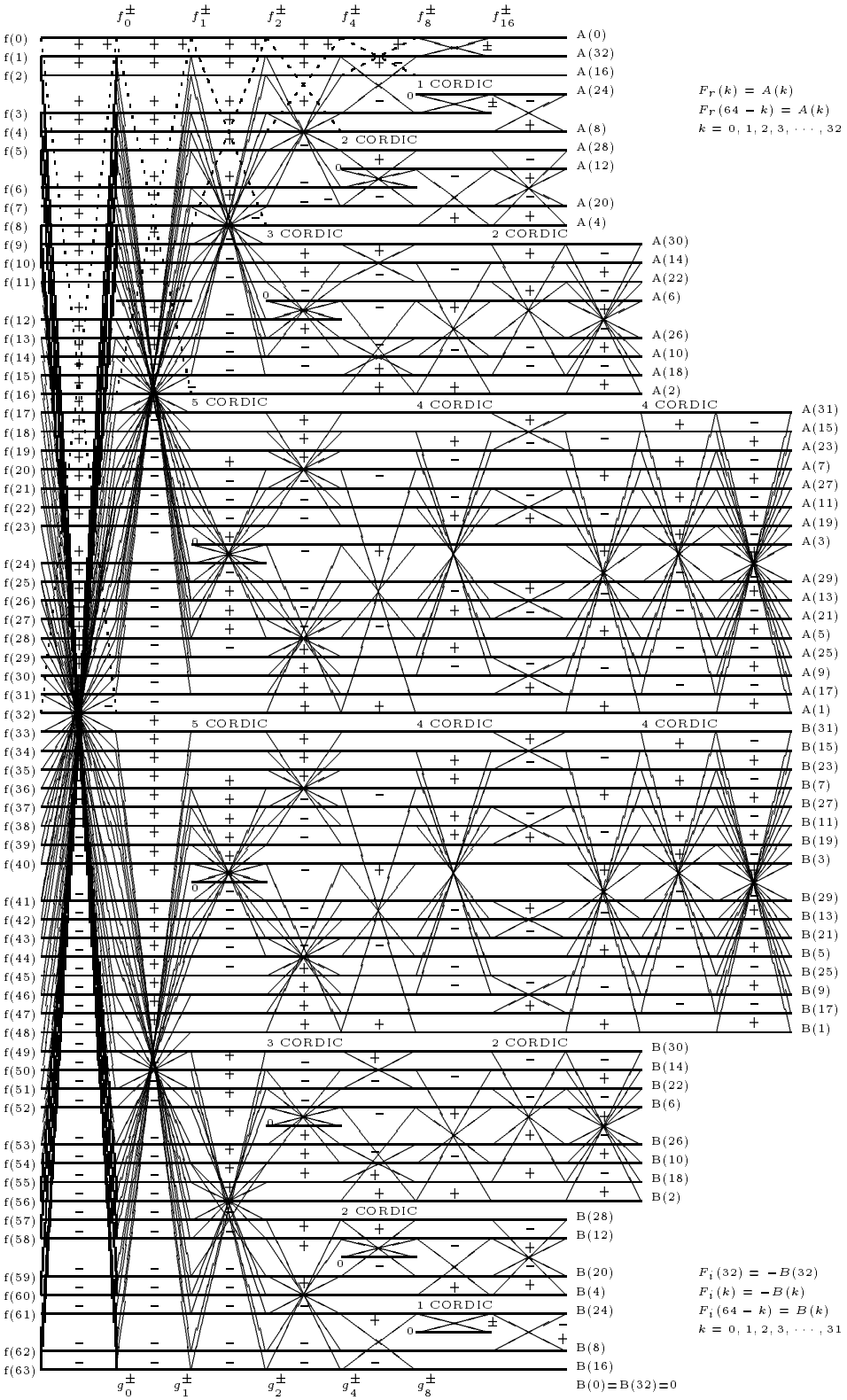

Fig. 1. A block diagram of 64-point DFT computations using the new DFT algorithm

2 multiplications and 2+4 additions, and each of $2^{n-2}(n-5)+2(n-1)$ CORDIC operations can be realized by using 3 multiplications and 3+4 additions. Therefore the total number of real multiplications required is

$$2^{n-2}(3n-13)+4n-2,$$

and the number of real additions required (including calculations for all $f^{\pm}(n), g^{\pm}(n)$) is

$$2^{n-2}(7n-29)+6n+2.$$

The traditional Fast Fourier Transform (FFT) uses $2n2^n$ real multiplications and $3n2^n$ additions. The FFCT [6] and Split-Radix [8] FFT algorithms of identical complexity (in which a complex multiplication is realized by 3 real multiplication and 3 real additions) need only $2^{n-1}(n-3)+2$ multiplications and $2^{n-1}(3n-5)+4$ additions. Table 1 shows a comparison between these and our method for different values of N.

Table 1. Number of real multiplications and additions in various DFT algorithms (S-R is the Split-Radix algorithm and FFT is the traditional FFT algorithm).

	Multiplications			*Additions*		
N	FFT	S-R	OUR	FFT	S-R	OUR
2^4	128	10	10	192	60	22
2^5	320	34	38	480	164	80
2^6	768	98	102	1152	420	234
2^7	1792	258	282	2688	1028	684
2^8	4096	642	734	6144	2436	1778
2^9	9216	1538	1826	13824	5636	4408
2^{10}	20480	3586	4390	30720	12804	10558

From the table we see that for small values of N, compared with the FFCT [6] and Split-Radix [8] methods, the arithmetic complexity in our method is almost the same. For large values of N our method use a few more multiplications, but fewer additions. But the total number of operations is lower, which is important if addition and multiplication take the same time, as in many DSP processors.

In the above we have just discussed the DFT/IDFT for real data. For complex data $f(n)=f_r(n)+jf_i(n)$, we have

$$\begin{aligned} F(k) &= \frac{1}{N}\sum_{n=0}^{N-1} f(n)e^{-j\frac{2\pi}{N}kn} \\ &= \frac{1}{N}\sum_{n=0}^{N-1}[f_r(n)+jf_i(n)]cos\frac{2\pi}{N}kn \\ &+ \frac{1}{N}\sum_{n=0}^{N-1}[f_i(n)-jf_r(n)]sin\frac{2\pi}{N}kn, \end{aligned}$$

$$\begin{aligned} f(n) &= \sum_{k=0}^{N-1} F(k)e^{j\frac{2\pi}{N}kn} \\ &+ \sum_{k=0}^{N-1}[F_r(k)+jF_i(k)]cos\frac{2\pi}{N}kn \\ &- \sum_{k=0}^{N-1}[F_i(k)-jF_r(k)]sin\frac{2\pi}{N}kn. \end{aligned}$$

So for complex data, using the new algorithm, the number of real multiplications and real additions is twice the number of operations for real data.

Compared to the traditional FFT/IFFT algorithm, the new algorithm requires more data-address computations, as shown in (3), (4) and (5). However, it is easy to realize these data-address computations in hardware or by table look-up, noting that $\sum_{j=1}^{M-1} a_j 2^{M-1-j}$ in binary is just a part of the reversed bit pattern of the binary representation for $k=\sum_{j=0}^{M-1} a_j 2^j$.

3. Hardware for the new DFT algorithm

Chen and Zhu [16] presented a column FFT suitable for VLSI. The basic building block is a 64-point column FFT and uses 16 butterfly processors, which include a total of 192 bit-serial real multipliers, 352 bit-serial adders and 32 31-bit shift registers. A crossbar switch matrix network (two 64×128 crossbar switch matrices for the basic building block) provides dynamically reconfigurable interconnections among the butterfly processors. For our new expression for real data DFT/IDFT, using a structure similar to their basic building block with a 64-point column FFT, we need 16 butterfly processors which consist of only 70 bit-serial real multipliers, 128 bit-serial adders, 26 31-bit shift registers and one 64×64 crossbar switch matrix network. Compared to their work, however, a more complex control over the crossbar is needed.

We may iteratively use 16 butterfly processors (BP) of the type shown in Figure 2, and a 64×64 crossbar switch for a 64-point column FFT. The whole computation can be divided into four steps: $f_1^{\pm}, g_1^{\pm}$ are produced at the beginning. $f_2^{\pm}, g_2^{\pm}$ and P_1^2, Q_1^2 are produced in the second step. f_4^-, $f_8^{\pm}$, g_4^+, $g_8^{\pm}$ and $P_1^4, Q_1^4, P_2^2, Q_2^2$ are produced at the third step. Finally all $F(k)$ are calculated (Note

actually only 32 values of $F(k) = A(k) - jB(k)$ are calculated, since $F(N-k) = A(k) + jB(k)$, the other 32 values are easily obtained). Considering all types of BP computations, it is possible to design a butterfly processor based on standard multipliers as shown in Figure 2.

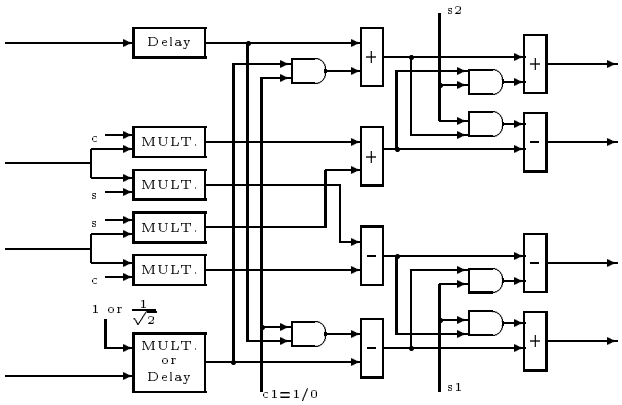

Fig. 2. The structure of a butterfly processor

where c1, s1, s2 are used to control the computations. Of the 16BPs, 6 needs five multipliers and the other 10 need four multipliers. For IDFT, $F_r(k) = F_r(N-k)$ and $F_i(k) = -F_i(N-n)$, $f_0^+(k) = 2F_r(k), g_0^-(k) = 2F_i(k)$, hence the computations of $f_0^+(k), g_0^-(k)$ are unnecessary. Because $f(n) = A(n) - B(n)$ and $f(N-n) = A(n) + B(n)$ for IDFT, an extra step is needed at the end for those computations. The remaining computations are the same as for the DFT.

In order to conveniently construct a long point column DFT/IDFT, it is better to design a basic building block for a 128-point column DFT/IDFT. The steps needed for a 128-point DFT/IDFT are only one more than the steps needed for a 64-point column DFT/IDFT. Because there are fewer multiplications in our new method, in almost the same size chip as Chen and Zhu's design for a 64 point DFT, it is possible to realize our basic building block for a 128-point column DFT. We can use several such basic building blocks (BBB) and some extra switches to realize a long-point column DFT/IDFT. Thus 2 BBBs are required for the 256-point DFT/IDFT, 4 BBBs for the 512-point DFT/IDFT and 8 BBBs for the 1024-point DFT/IDFT.

An alternative way to implement the DFT/IDFT is to use pipelined CORDIC processors to realize the equivalent of the four multiplication, the following adders and the computation of the trigonometric coefficients. A CORDIC computation performs a vector rotation:

$$\begin{bmatrix} X' \\ Y' \end{bmatrix} = \begin{bmatrix} sin\alpha & cos\alpha \\ cos\alpha & -sin\alpha \end{bmatrix} \begin{bmatrix} X \\ Y \end{bmatrix},$$

by a number of micro rotations using a predetermined set of angles ϕ_i derived from α.

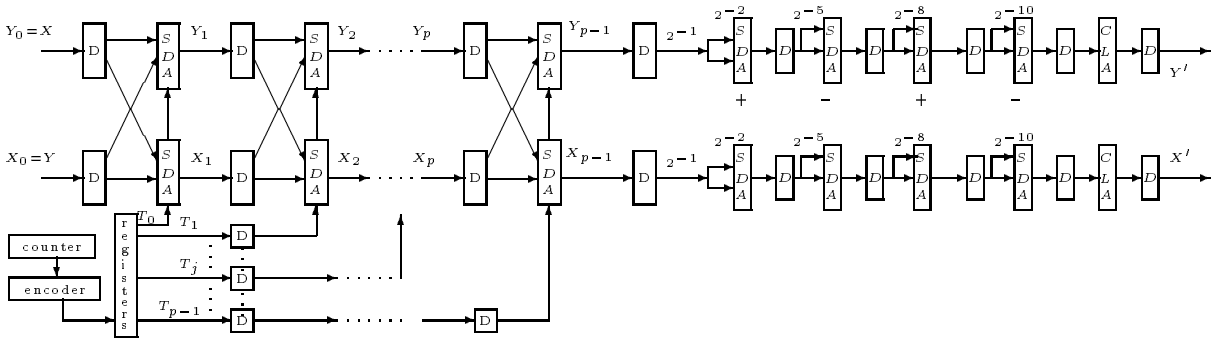

Fig. 3. A pipelined CORDIC structure

The pipelined CORDIC structure is shown in Figure 3, in which the leftmost part is controlled by $T_0, T_1, \cdots, T_{p-1}$ (as encodings of the angles ϕ_i) performs the equivalent of four multiplications, one addition and one subtraction [13]. But using the CORDIC algorithm, the results must be multiplied by a constant $\frac{1}{K_c}$, $K_c \approx 1.6467603$, this multiplication can be realized by a simple structure as shown in the rightmost part of Figure 3, because $\frac{1}{K_c} \approx \frac{1}{2}(1+\frac{1}{4})(1-\frac{1}{32})(1+\frac{1}{256})(1-\frac{1}{1024})$ turns out to be a good approximation.

We can see from the above analysis that the DFT/IDFT can be calculated using CORDIC computations and additions/subtractions. This was actually our main motivation for the derivation of the new method. It is quite efficient for a high speed long-point DFT/IDFT. For ex-

ample for 100MHz input data rate, a 1024-point DFT/IDFT can be realized using a single pipelined CORDIC structure, if the delay of each step is below $6.6ns$. It is possible to realize this hardware using redundant SDAs and CLAs (signed-digit and carry lookahead adders), which only need a 2-5ns delay. Of course the pipelined CORDIC structure can also be applied to the traditional FFT, but it would require a step delay of less than $2.5ns$ for a DFT/IDFT beyond 256 points at 100MHz, so at least two pipelined CORDIC processors would be needed to achieve that input rates.

4. Conclusions

By exploiting some trigonometric identities in the DFT computation, the number of multiplications needed for DFT/IDFT has been reduced compared to the traditional FFT, so in almost the same area as needed for the traditional 64-point FFT, we can implement a 128-point real data DFT. One way to realize the new algorithm in hardware, which is more efficient for a long-point DFT, is to use a pipelined CORDIC algorithm, which simultaneously completes four multiplications. In the pipelined CORDIC structure, the data rate of the DFT/IDFT depends only on one carry look-ahead adder as the critical component in the pipeline. At present integration technology, it is possible to perform each pipeline step in about 2-5ns. Thus a single CORDIC pipeline is sufficient to realize a high speed DFT/IDFT, e.g. at a 100MHz input rate, a 1024-point transform can be realized with a 200MHz clocking of a single CORDIC pipeline.

Appendix

When $L = 1, 2, 4, 8, \cdots, N/8$, for the odd multiples of L, $k = L, 3L, 5L, \cdots, \frac{N}{2}-L$, we can rewrite $A(k)$ as

$$
\begin{aligned}
A(k) \;=\; & \sum_{n=1}^{\frac{N}{8L}-1}\left[f_L^-(n)cos\frac{2\pi}{N}kn + S_k f_L^-(\frac{N}{4L}-n)sin\frac{2\pi}{N}kn\right] \\
& + f_L^-(\frac{N}{8L})cos\frac{k\pi}{4L} + f^+_{\lfloor\frac{L}{2}\rfloor}(0) \\
= \; & \sum_{n=1}^{\frac{N}{16L}-1}\left[f_L^-(n)cos\frac{2\pi}{N}kn + S_k f_L^-(\frac{N}{4L}-n)sin\frac{2\pi}{N}kn\right] \\
& + \sum_{n=\frac{N}{16L}+1}^{\frac{N}{8L}-1}\left[f_L^-(n)cos\frac{2\pi}{N}kn + S_k f_L^-(\frac{N}{4L}-n)sin\frac{2\pi}{N}kn\right] \\
& + \left[f_L^-(\frac{N}{16L})cos\frac{k\pi}{8L} + S_k f_L^-(\frac{N}{4L}-\frac{N}{16L})sin\frac{k\pi}{8L}\right] \\
& + f_L^-(\frac{N}{8L})cos\frac{k\pi}{4L} + f^+_{\lfloor\frac{L}{2}\rfloor}(0) \\
= \; & \sum_{n=1}^{\frac{N}{16L}-1}\Big\{f_L^-(n)cos\frac{2\pi}{N}kn + S_k f_L^-(\frac{N}{4L}-n)sin\frac{2\pi}{N}kn \\
& + f_L^-(\frac{N}{8L}-n)cos\frac{2k\pi}{N}(\frac{N}{8L}-n) \\
& + S_k f_L^-(\frac{N}{8L}+n)sin\frac{2k\pi}{N}(\frac{N}{8L}-n)\Big\} \\
& + \left[f_L^-(\frac{N}{16L})cos\frac{k\pi}{8L} + S_k f_L^-(\frac{3N}{16L})sin\frac{k\pi}{8L}\right] \\
& + f_L^-(\frac{N}{8L})cos\frac{k\pi}{4L} + f^+_{\lfloor\frac{L}{2}\rfloor}(0) \\
= \; & \sum_{n=1}^{\frac{N}{16L}-1}\Bigg\{\Big[f_L^-(n) + f_L^-(\frac{N}{8L}-n)cos\frac{k\pi}{4L} \\
& + S_k f_L^-(\frac{N}{8L}+n)sin\frac{k\pi}{4L}\Big]cos\frac{2\pi}{N}kn \\
& + S_k\Big[S_k f_L^-(\frac{N}{8L}-n)sin\frac{k\pi}{4L} - f_L^-(\frac{N}{8L}+n)cos\frac{k\pi}{4L} \\
& + f_L^-(\frac{N}{4L}-n)\Big]sin\frac{2\pi}{N}kn\Bigg\} \\
& + \left[f_L^-(\frac{N}{16L})cos\frac{k\pi}{8L} + S_k f_L^-(\frac{3N}{16L})sin\frac{k\pi}{8L}\right] \\
& + f_L^-(\frac{N}{8L})cos\frac{k\pi}{4L} + f^+_{\lfloor\frac{L}{2}\rfloor}(0).
\end{aligned}
$$

Now for $n = 1, 2, 3, \cdots, \frac{N}{8L}-1$ we define

$$P_L^1(n,k) = f_L^-(n), \quad P_L^1(\frac{N}{4L}-n,k) = f_L^-(\frac{N}{4L}-n)(1)$$

and for $i = 1, 2, 4, 8, \cdots, \frac{N}{16L}$, $n = 1, 2, 3, \cdots, \frac{N}{16Li}-1$

$$
\begin{aligned}
P_L^{2i}(n,k) \;=\; & P_L^i(n,k) \qquad (2) \\
& + \left[P_L^i(\frac{N}{8Li}-n,k)cos\frac{k\pi}{4Li} + S_k P_L^i(\frac{N}{8Li}+n,k)sin\frac{k\pi}{4Li}\right] \\
P_L^{2i}(\frac{N}{8Li}-n,k) \;=\; & \qquad (3) \\
& \left[S_k P_L^i(\frac{N}{8Li}-n,k)sin\frac{k\pi}{4Li} - P_L^i(\frac{N}{8Li}+n,k)cos\frac{k\pi}{4Li}\right] \\
& + P_L^i(\frac{N}{4Li}-n,k)
\end{aligned}
$$

together with

$$R_L^{2i}(k) = P_L^i(\frac{N}{16Li}, k)cos\frac{k\pi}{8Li} + S_k P_L^i(\frac{3N}{16Li}, k)sin\frac{k\pi}{8Li}.$$

Now $A(k)$ can then be further rewritten as

$$\begin{aligned}
A(k) \;=\; & \\
& \sum_{n=1}^{\frac{N}{16L}-1}\Big[P_L^2(n,k)cos\frac{2\pi}{N}kn + S_k P_L^2(\frac{N}{8L}-n,k)sin\frac{2\pi}{N}kn\Big] \\
& \quad + R_L^1(k) + f_L^-(\frac{N}{8L})cos\frac{k\pi}{4L} + f^+_{\lfloor\frac{L}{2}\rfloor}(0) \\
& \sum_{n=1}^{\frac{N}{32L}-1}\Big[P_L^4(n,k)cos\frac{2\pi}{N}kn + S_k P_L^4(\frac{N}{16L}-n,k)sin\frac{2\pi}{N}kn\Big] \\
& \quad + R_L^2(k) + R_L^1(k) + f_L^-(\frac{N}{8L})cos\frac{k\pi}{4L} + f^+_{\lfloor\frac{L}{2}\rfloor}(0) \\
& \sum_{n=1}^{\frac{N}{16Li}-1}\Big[P_L^{2i}(n,k)cos\frac{2\pi}{N}kn + S_k P_L^{2i}(\frac{N}{16L}-n,k)sin\frac{2\pi}{N}kn\Big] \\
& \quad + R_L^i(k) + \cdots + R_L^2(k) + R_L^1(k) \\
& \quad + f_L^-(\frac{N}{8L})cos\frac{k\pi}{4L} + f^+_{\lfloor\frac{L}{2}\rfloor}(0) \\
& = \sum_{i=1,2,4,8,\cdots}^{\frac{N}{16L}} R_L^i(k) + f_L^-(\frac{N}{8L})cos\frac{k\pi}{4L} + f^+_{\lfloor\frac{L}{2}\rfloor}(0).
\end{aligned}$$

For $i = 1, 2, 4, \cdots, \frac{N}{16L}$ and $n = \frac{N}{16Li}$, we define $P_L^{2i}(n,k)$ by

$$\begin{aligned}
P_L^1(\frac{N}{8L}, k) \;&=\; f_L^-(\frac{N}{8L})cos\frac{k\pi}{4L} + f^+_{\lfloor\frac{L}{2}\rfloor}(0) \\
P_L^{2i}(\frac{N}{16Li}, k) \;&=\; R_L^i(k) + P_L^i(\frac{N}{8Li}, k).
\end{aligned}$$

So finally we arrive at the following expression for $A(k)$

$$\begin{aligned}
A(k) \;&=\; \sum_{i=1,2,4\cdots}^{\frac{N}{16L}} R_L^{2i}(k) + P_L^1(\frac{N}{8L}, k) \\
&= \sum_{i=j,\cdots}^{\frac{N}{16L}} R_L^{2i}(k) + P_L^j(\frac{N}{8Lj}, k) = P_L^{\frac{N}{8L}}(1,k).
\end{aligned}$$

Now we will discuss how to calculate $P^{2i}(n,k)$. We divide the calculation of $P^{2i}(n,k)$ into two cases, one is for $i = 1, 2, 4, \cdots, \frac{N}{32L}$ and $n = 1, 2, 4, \cdots, \frac{N}{16Li} - 1$, another is for $n = \frac{N}{16Li}$ and $i = 1, 2, 4, \cdots, \frac{N}{16L}$. We begin with the first case. In the above equation the range of k is $k = L, 3L, 5L, \cdots, \frac{N}{2L} - 1$. However, since $P_L^1(n,k)$ is independent of k, according to some trigonometric identities, we have for $k = L, 3L, 5L, \cdots, (2i-1)L$

$$\begin{aligned}
P_L^{2i}(n, 4Li-k) \;&=\; P_L^i(n,k) \qquad (4)\\
&- \Big[P_L^i(\frac{N}{8Li}-n,k)cos\frac{k\pi}{4Li} + S_k P_L^i(\frac{N}{8Li}+n,k)sin\frac{k\pi}{4Li}\Big] \\
P_L^{2i}(\frac{N}{4Li}-n, 2Li-k) \;&=\; P_L^i(\frac{N}{4Li}-n,k) \qquad (5)\\
&- \Big[S_k P_L^i(\frac{N}{8Li}-n,k)sin\frac{k\pi}{4Li} - P_L^i(\frac{N}{8Li}+n,k)cos\frac{k\pi}{4Li}\Big]
\end{aligned}$$

and for $k = L, 3L, 5L, \cdots, (4i-1)L$

$$P_L^{2i}(n, 8Lij \pm k) = P_L^{2i}(n,k), \quad j = 1, 2, 3, \cdots, \frac{N}{16Li}.$$

Combining equations (3), (4) and (5), we have for $k = L, 3L, 5L, \cdots (2i-1)L$

$$\begin{aligned}
\begin{bmatrix} P_L^{2i}(n,k) \\ P_L^{2i}(n,4Li-k) \\ P_L^{2i}(\frac{N}{8Li}-n, 4Li-k) \\ P_L^{2i}(\frac{N}{8Li}-n,k) \end{bmatrix} &= \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix} \qquad (6)\\
&\times \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos\theta_k & sin\theta_k & 0 \\ 0 & sin\theta_k & -cos\theta_k & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} P_L^i(n,k) \\ P_L^i(\frac{N}{8Li}-n,k) \\ P_L^i(\frac{N}{8Li}+n,k) \\ P_L^i(\frac{N}{4Li}-n,k) \end{bmatrix}
\end{aligned}$$

with $\theta_k = S_k \frac{k\pi}{4Li}$.

In order to convenience the recursive computation we replace $P_L^i(n,k)$ with $p_L^i(n_k)$, that is, for $k = L, 3L, 5L, \cdots, (2i-1)L$, we define

$$p_L^1(n) = P_L^1(n,k), \text{ for } n = 1, 2, 3, \cdots, N/8L - 1$$

where

$$\begin{aligned}
\begin{bmatrix} p_L^i(n_k) \\ p_L^i(n'_k) \\ p_L^i(\frac{N}{4L}-n'_k) \\ p_L^i(\frac{N}{4L}-n_k) \end{bmatrix} \;&=\; \begin{bmatrix} P_L^i(n,k) \\ P_L^i(\frac{N}{8Li}-n,k) \\ P_L^i(\frac{N}{8Li}+n,k) \\ P_L^i(\frac{N}{4Li}-n,k) \end{bmatrix}, \qquad (7)\\
\begin{bmatrix} p_L^{2i}(n_k) \\ p_L^{2i}(n'_k) \\ p_L^{2i}(\frac{N}{4L}-n'_k) \\ p_L^{2i}(\frac{N}{4L}-n_k) \end{bmatrix} \;&=\; \begin{bmatrix} P_L^{2i}(n,k) \\ P_L^{2i}(n,4Li-k) \\ P_L^{2i}(\frac{N}{8Li}-n,4Li-k) \\ P_L^{2i}(\frac{N}{8Li}-n,k) \end{bmatrix}
\end{aligned}$$

so equations (3) and (6) can be rewritten as the following

$$\begin{bmatrix} p_L^1(n) \\ p_L^1(\frac{N}{4L}-n) \end{bmatrix} \;=\; \begin{bmatrix} f_L^-(n) \\ f_L^-(\frac{N}{4L}-n) \end{bmatrix}$$

for $n = 1, 2, 3, \cdots, N/8L - 1$ and

$$\begin{bmatrix} p_L^{2i}(n_k) \\ p_L^{2i}(n'_k) \\ p_L^{2i}(\frac{N}{4L}-n'_k) \\ p_L^{2i}(\frac{N}{4L}-n_k) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

$$\times \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos\theta_k & sin\theta_k & 0 \\ 0 & sin\theta_k & -cos\theta_k & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_L^i(n_k) \\ p_L^i(n'_k) \\ p_L^i(\frac{N}{4L} - n'_k) \\ p_L^i(\frac{N}{4L} - n_k) \end{bmatrix}$$

where n_k, n'_k depend on n and k. Recursively calculating equation (7), if k is represented as $k = \sum_{j=0}^{M-1} a_j 2^j L$ with $M = log_2 \frac{N}{2L}$, we find the following relation:

$$\begin{aligned} n_k &= N_k + S_k n \\ n'_k &= N_k + S_k(\frac{N}{8Li} - n) \qquad (8) \\ N_k &= a_1 2^{M-2} + (-1)^{a_1} \sum_{j=2}^{M-1} a_j 2^{M-1-j}. \qquad (9) \end{aligned}$$

Therefore, for $i = 1, 2, 4, \cdots, \frac{N}{32L}$ and $n = 1, 2, 3, \cdots, \frac{N}{16Li} - 1$, all $P_L^{2i}(n,k) = p_L^{2i}(n_k)$ values can be obtained using the above equations.

For the second case, that is for $i = 1, 2, 4, \cdots, \frac{N}{16Li} - 1, n = \frac{N}{16Li}$, again using some trigonometric identities, we have

$$\begin{bmatrix} R_L^i(k) \\ R_L^i(8Li-k) \\ R_L^i(4Li+k) \\ R_L^i(4Li-k) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ -1 & 0 \\ 0 & -1 \\ 0 & 1 \end{bmatrix} \times \begin{bmatrix} cos\frac{k\pi}{8Li} & S_k sin\frac{k\pi}{8Li} \\ sin\frac{k\pi}{8Li} & -S_k cos\frac{k\pi}{8Li} \end{bmatrix} \begin{bmatrix} P_L^i(\frac{N}{16Li}, k) \\ P_L^i(\frac{3N}{16Li}, k) \end{bmatrix}$$

for $k = L, 3L, 5L, \cdots, (2i-1)L$ together with

$$R_L^i(16Lij \pm k) = R_L^i(k),$$

for $j = 1, 2, 3, \cdots, \frac{N}{32Li}$ and the odd multiples $k = L, 3L, 5L, \cdots, (8i-1)L$.

From equation (3), we can obtain

$$\begin{aligned} P_L^1(\frac{N}{8L}, 8Lj \pm L) &= P_L^1(\frac{N}{8L}, L) \\ &= f^+_{\lfloor \frac{L}{2} \rfloor}(0) + f_L^-(\frac{N}{8L}) cos\frac{\pi}{4} \\ P_L^1(\frac{N}{8L}, 8Lj \pm 3L) &= P_L^1(\frac{N}{8L}, 3L) \\ &= f^+_{\lfloor \frac{L}{2} \rfloor}(0) - f_L^-(\frac{N}{8L}) cos\frac{\pi}{4} \end{aligned}$$

for $j = 1, 2, 3, \cdots, \frac{N}{16L}$ and

$$P_L^i(\frac{N}{8Li}, 8Lij \pm k) = P_L^i(\frac{N}{8Li}, k)$$

for $k = L, 3L, 5L, \cdots, (4i-1)L,\ j = 2, 3, \cdots, \frac{N}{16Li}$. Therefore we have

$$\begin{bmatrix} P_L^{2i}(\frac{N}{16Li}, k) \\ P_L^{2i}(\frac{N}{16Li}, 8Li-k) \\ P_L^{2i}(\frac{N}{16Li}, 4Li+k) \\ P_L^{2i}(\frac{N}{16Li}, 4Li-k) \end{bmatrix}$$

$$\begin{aligned} &= \begin{bmatrix} R_L^i(k) \\ R_L^i(8Li-k) \\ R_L^i(4Li+k) \\ R_L^i(4Li-k) \end{bmatrix} + \begin{bmatrix} P_L^i(\frac{N}{8Li}, k) \\ P_L^i(\frac{N}{8Li}, 8Li-k) \\ P_L^i(\frac{N}{8Li}, 4Li+k) \\ P_L^i(\frac{N}{8Li}, 4Li-k) \end{bmatrix} \\ &= \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos\frac{k\pi}{8Li} & S_k sin\frac{k\pi}{8Li} & 0 \\ 0 & sin\frac{k\pi}{8Li} & -S_k cos\frac{k\pi}{8Li} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\ &\quad \times \begin{bmatrix} P_L^i(\frac{N}{8Li}, k) \\ P_L^i(\frac{N}{16Li}, k) \\ P_L^i(\frac{3N}{16Li}, k) \\ P_L^i(\frac{N}{8Li}, 4Li-k) \end{bmatrix}. \qquad (10) \end{aligned}$$

Similarly to the first case, in order to convenience the recursive computation we introduce $p_L^i(n_k)$ to replace $P_L^i(n,k)$, that is we define

$$\begin{aligned} \begin{bmatrix} p_L^1(0) \\ p_L^1(\frac{N}{16L}) \\ p_L^1(\frac{3N}{16L}) \\ p_L^1(\frac{N}{8L}) \end{bmatrix} &= \begin{bmatrix} P_L^1(\frac{N}{8L}, L) \\ P_L^1(\frac{N}{16L}, k) \\ P_L^1(\frac{3N}{16L}, k) \\ P_L^1(\frac{N}{8L}, 3L) \end{bmatrix} \\ &= \begin{bmatrix} f^+_{\lfloor \frac{L}{2} \rfloor}(0) + f_L^-(\frac{N}{8L}) cos\frac{\pi}{4} \\ f_L^-(\frac{N}{16L}) \\ f_L^-(\frac{3N}{16L}) \\ f^+_{\lfloor \frac{L}{2} \rfloor}(0) - f_L^-(\frac{N}{8L}) cos\frac{\pi}{4} \end{bmatrix} \end{aligned}$$

and for the general case when $S_k = 1$,

$$\begin{aligned} \begin{bmatrix} p_L^i(n_k - \frac{N}{16Li}) \\ p_L^i(n_k) \\ p_L^i(n'_k) \\ p_L^i(n'_k - \frac{N}{16Li}) \end{bmatrix} &= \begin{bmatrix} P_L^i(\frac{N}{8Li}, k) \\ P_L^i(\frac{N}{16Li}, k) \\ P_L^i(\frac{3N}{16Li}, k) \\ P_L^i(\frac{N}{8Li}, 4Li-k) \end{bmatrix}, \\ \begin{bmatrix} p_L^{2i}(n_k - \frac{N}{16Li}) \\ p_L^{2i}(n'_k) \\ p_L^{2i}(n_k) \\ p_L^{2i}(n'_k - \frac{N}{16Li}) \end{bmatrix} &= \begin{bmatrix} P_L^{2i}(\frac{N}{16Li}, k) \\ P_L^{2i}(\frac{N}{16Li}, 8Li-k) \\ P_L^{2i}(\frac{N}{16Li}, 4Li+k) \\ P_L^{2i}(\frac{N}{16Li}, 4Li-k) \end{bmatrix} \end{aligned} \qquad (11)$$

and when $S_k = -1$,

$$\begin{aligned} \begin{bmatrix} p_L^i(n'_k - \frac{N}{16Li}) \\ p_L^i(n_k) \\ p_L^i(n'_k) \\ p_L^i(n_k - \frac{N}{16Li}) \end{bmatrix} &= \begin{bmatrix} P_L^i(\frac{N}{8Li}, k) \\ P_L^i(\frac{N}{16Li}, k) \\ P_L^i(\frac{3N}{16Li}, k) \\ P_L^i(\frac{N}{8Li}, 4Li-k) \end{bmatrix}, \\ \begin{bmatrix} p_L^{2i}(n'_k - \frac{N}{16Li}) \\ p_L^{2i}(n_k) \\ p_L^{2i}(n'_k) \\ p_L^{2i}(n_k - \frac{N}{16Li}) \end{bmatrix} &= \begin{bmatrix} P_L^{2i}(\frac{N}{16Li}, k) \\ P_L^{2i}(\frac{N}{16Li}, 8Li-k) \\ P_L^{2i}(\frac{N}{16Li}, 4Li+k) \\ P_L^{2i}(\frac{N}{16Li}, 4Li-k) \end{bmatrix} \end{aligned} \qquad (12)$$

where $n_k = N_k + S_k \frac{N}{16Li}, \quad n'_k = \frac{N}{4L} - n_k. \qquad (13)$

and with $M = log_2 \frac{N}{2L}$

$$k = \sum_{j=0}^{M-1} a_j 2^j L, \quad N_k = \sum_{j=1}^{M-1} a_j 2^{M-1-j}. \qquad (14)$$

Combining the above two equations (11) and (12), equation (10) is rewritten as

$$\begin{bmatrix} p_L^{2i}(n'_k - \frac{N}{16Li}) \\ p_L^{2i}(n_k) \\ p_L^{2i}(n'_k) \\ p_L^{2i}(n_k - \frac{N}{16Li}) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & sin\theta'_k & -cos\theta'_k & 0 \\ 0 & cos\theta'_k & sin\theta'_k & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_L^{i}(n'_k - \frac{N}{16Li}) \\ p_L^{i}(n_k) \\ p_L^{i}(n'_k) \\ p_L^{i}(n_k - \frac{N}{16Li}) \end{bmatrix}$$

with

$$\theta'_k = \frac{\pi}{4}(1 - S_k) + S_k \frac{k\pi}{8Li} \qquad (15)$$

for $k = L, 3L, 5L, \cdots, (2i-1)L$.

Finally we get $A(k)$ for $k = L, 3L, 5L, \cdots, \frac{N}{2} - L$

$$A(k) = p_L^{\frac{N}{8L}}(N_k).$$

Similarly when $L = 1, 2, 4, 8, \cdots, \frac{N}{8}$, for $B(k)$, $k = L, 3L, 5L \cdots, \frac{N}{2} - L$, we have

$$\begin{bmatrix} q_L^1(\frac{N}{4L}) \\ q_L^1(\frac{N}{8L}) \end{bmatrix} = \begin{bmatrix} g^-_{\lfloor \frac{L}{2} \rfloor}(\frac{N}{4L}) - g_L^+(\frac{N}{8L})cos\frac{\pi}{4} \\ -g^-_{\lfloor \frac{L}{2} \rfloor}(\frac{N}{4L}) - g_L^+(\frac{N}{8L})cos\frac{\pi}{4} \end{bmatrix}$$

$$\begin{bmatrix} q_L^1(n) \\ q_L^1(\frac{N}{4L} - n) \end{bmatrix} = \begin{bmatrix} g_L^+(n) \\ g_L^+(\frac{N}{4L} - n) \end{bmatrix}$$

for $n = 1, 2, 3, \cdots, \frac{N}{8L} - 1$, and

$$\begin{bmatrix} q_L^{2i}(n_k) \\ q_L^{2i}(n'_k) \\ q_L^{2i}(\frac{N}{4L} - n'_k) \\ q_L^{2i}(\frac{N}{4L} - n_k) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -cos\theta_k & sin\theta_k & 0 \\ 0 & sin\theta_k & cos\theta_k & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} q_L^{i}(n_k) \\ q_L^{i}(n'_k) \\ q_L^{i}(\frac{N}{4L} - n'_k) \\ q_L^{i}(\frac{N}{4L} - n_k) \end{bmatrix}$$

for $i = 1, 2, 4, \cdots, \frac{N}{32L}$, and where n_k, n'_k and θ_k are respectively defined by (9) and (6), and

$$\begin{bmatrix} q_L^{2i}(n'_k + \frac{N}{16Li}) \\ q_L^{2i}(n_k) \\ q_L^{2i}(n'_k) \\ q_L^{2i}(n_k + \frac{N}{16Li}) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & 1 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & sin\theta'_k & cos\theta'_k & 0 \\ 0 & cos\theta'_k & -sin\theta'_k & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} q_L^{i}(n'_k + \frac{N}{16Li}) \\ q_L^{i}(n_k) \\ q_L^{i}(n'_k) \\ q_L^{i}(n_k + \frac{N}{16Li}) \end{bmatrix},$$

for $i = 1, 2, 4, \cdots, \frac{N}{16L}$, where n_k, n'_k and θ_k are respectively defined by (13) and (15). Finally we have

$$B(k) = q_L^{\frac{N}{8L}}(\frac{N}{4L} - N_k),$$

with N_k defined by (14).

References

1. James W. Cooley and John W. Tukey,"An Algorithm for the Machine Calculation of Complex Fourier Series", *Math. Comp.*, Vol.19(90), pp.297-301, Apr. 1965.
2. W. T. Cochran, *et al*, " What is the Fast Fourier Transform?", *Proc. IEEE*, Vol.55, pp.1664-1674, Oct. 1967.
3. R. C. Singleton, "On Computing the Fast Fourier Transform", *CACM*, Vol.10, pp.153-161, Feb. 1971.
4. G. D. Bergland, "A Guided Tour of the Fast Fourier Transform", *IEEE Spectrum*, Vol.6, No.7, pp.41-52, July 1969.
5. J. B. Martens, "Recursive cyclotomic factorization - A New Algorithm for Calculating the Discrete Fourier Transform", *IEEE Trans. on ASSP*, Vol.ASSP-32, pp.750-761, Apr. 1984.
6. M. Vetterli and H. J. Nussbaumer,"Simple FFT and DCT Algorithm with Reduced Number of Operation", *Signal Processing*, Vol.6, pp.267-278, July 1984
7. Z. Wang,"Fast Algorithms for the Discrete W Transform and the Discrete Fourier Transform", *IEEE Trans. on ASSP*, Vol.ASSP-32, pp.803-816, Aug. 1984.
8. P. Duhamel, "Implementation of Split-Radix FFT Algorithms for Complex, Real, and Real-Symmetric Data", *IEEE Trans. on ASSP*, Vol.ASSP-34, No.2, pp.285-295, Apr. 1986.
9. C. D. Thompson, "Fourier Transforms in VLSI", *IEEE Trans. Comp.*, Vol.10, pp.1047-1057, Nov. 1983.
10. A. Norton and A. J. Siberger,"Parallelization and Performance Analysis of the Cooley-Tukey FFT Algorithm for Shared-memory Architecture", *IEEE Trans. Comp.*, Vol.C-36, No.5, pp.581-591, May 1987
11. K. Yamashita, *et al*,"A Wafer-Scale 170000-Gate FFT Processor with Built-In Test Circuits", *IEEE Journal of Solid-State Circuits*, Vol.23, pp.336-342, 1988
12. J. O'Brien, *et al*,"A 200 MIPS Signal-Chip 1K FFT Processor", *1989 Proc. IEEE Int. Solid-State Circuits Conf.*, pp.166-167, Feb. 1989.
13. J. You and S. S. Wong, "A High Performance Single Chip FFT Array Processor for Wafer Scale Integration", in *Proc. of the Int. Conf. on Wafer Scale Integration*, pp.60-67, 1990.

14. J. E. Whelchel, *et al*, " The Systolic Phase Rotation FFT - A New Algorithm and Parallel Processor architecture", In *Proc. 1990. Int. Conf. Acoust., Speech, Signal Processing*, Vol.2, pp.1021-1024, 1990.

15. Earl E. Swartzlander, Jr., *et al*, "A Radix-8 Scale FFT Processor", *Journal of VLSI Signal Processing*, Vol.4, No.2/3, pp.165-176, May 1992.

16. Tom Chen and Li Zhu, "An Expandable Column FFT architecture Using Circuit Switching Networks", *Journal of VLSI Signal Processing*, Vol.6, No.3, pp.243-257, Dec. 1993.

17. F. Zhou and P. Kornerup, "A High Speed DCT/IDCT Using a Pipelined CORDIC Algorithm," *Proc. 12th IEEE Symposium on Computer Arithmetic*, IEEE Press, pp.180-187, July 1995.

Feng Zhou (no picture available) was born in Hangzhou, Zhejiang, PRC, in 1961, and received the PhD degree in 1991 from Zheijiang University, Hangzhou. While serving with the Dept. of Information and Electronic Engineering at the same university, he spent a leave in 1994 and 1995 as a post.doc. with Odense University, Denmark. On return from this leave he was designated Director of the Signal Processing Lab. His research interests are in image compression, coding, signal and image processing, computer vision and information theory.

Peter Kornerup was born in Aarhus, Denmark, 1939. He received the mag.scient. degree in mathematics from Aarhus University, Denmark, in 1967. After a period with the University Computing Center, from 1969 involved in establishing the computer science curriculum at Aarhus University, he helped found the Computer Science Department in 1971, and through most of the 70's and 80's he served as chairman of the department. He spent a leave during 1975/76 with the University of Southwestern Louisiana, Lafayette, LA, and another in 1979 with Southern Methodist University, Dallas, TX. Since 1988 he has been Professor of Computer Science at Odense University, Odense, Denmark, where he has also served as the chairman of its Department of Mathematics and Computer Science. His research interests include compiler construction, computer networks and computer architecture, but in particular computer arithmetic and number representations. Prof. Kornerup has served on the program committees for a number of IEEE and ACM sponsored meetings, in particular he has been on the Program Committee of the 4th through the 13th IEEE Symposium on Computer Arithmetic, and served as Program Chairman for these symposia in 1983, 1991 and for the 14th in 1999. He was an associate editor of the IEEE Transactions on Computers during 1991–95, and is a member of the ACM and the IEEE.