DCT Computation Using Real-Valued DFT Algorithms

Ryszard Stasiński
Dept. of Electronics and Telecommunications
Poznań University of Technology
Poznań, Poland

ABSTRACT

In the paper it has been shown that when data vector size is odd, then the discrete cosine transform (DCT) can be computed by a real-valued DFT algorithm for appropriately permuted data samples. The same is true for the DST. Moreover, for composite transform sizes prime factor DCT algorithms can be constructed consisting of odd size real-valued DFT algorithms and DCT/DST algorithms which sizes are equal to a power of number 2. Similarly as for the DFT, the algorithms can be nested. It has been proven that the new DCT/DST algorithms require the smallest known numbers of arithmetical operations.

1. Introduction

The discrete cosine transform (DCT) is one of the most important tools of digital signal processing. It is widely used in image processing, especially in image coding, speech analysis and coding, implementation of filter banks etc. Since its formulation it has been known that the DCT can be computed efficiently through a DFT, and not all 'custom' DCT algorithms have been better in this respect. The first algorithm for DCT sizes being a power of number 2 requiring the smallest number of arithmetical operations has been described in [1]. Optimality of the algorithm can be deduced from the provided in the paper construction showing how basing on this algorithm the split-radix FFT can be derived. The new applications of the DCT prompted interest in DCT algorithms for any data vector sizes. Probably the most complete set of fast DCT algorithms can be found in [2], however, algorithms presented there have complex descriptions, hence, they are difficult to evaluate and implement.

The derived in this paper DCT and DST algorithms are based on the prime factor FFT algorithm (PFA FFT), which greatly simplifies their description and evaluation. The idea of derivation has been suggested in [3]. In section 2 it is shown that DCTs and DSTs for data vector sizes being odd numbers can be computed by DFT algorithms for real-valued data of the same size. Then, in section 3 the result is generalized to the case when data vector sizes are $q2^s$, q is odd, then DCTs can be computed by prime factor algorithms consisting of q-point real-valued FFTs and 2^s-point DCT/DST algorithms. In section 4 it is outlined how nesting (i.e. WFTA-like) and common factor DCT/DST algorithms can be constructed. Finally, in section 5 it is proven that the new algorithms require the smallest known numbers of arithmetical operations.

2. Derivation

Let us start with definitions of the transforms considered in the paper. The not-normalized version of the N-point DCT (DCT-II in [4]) for data samples $x(n)$ can be defined as follows:

$$X_C(k) = c(k)\cdot X(k) = c(k)\sum_{n=0}^{N-1} x(n)\cdot\cos\left[(n+\tfrac{1}{2})k\tfrac{\pi}{N}\right] \quad (1)$$

$$c(k) = \begin{cases} 1/\sqrt{2} & \text{for} \quad k=0, \\ 1 & \text{for} \quad k = 0,1,\ldots,N-1, \end{cases}$$

where $X(k)$ samples are computed by algorithms presented in the paper. The DST (DST-II in [4]) is defined in a similar way, except cosine function is replaced by the sine one, $k=1,2,\ldots,N$, and $c(k)$ is not equal to 1 for $k=N$. The DFT is defined as usual:

$$X_F(k) = \sum_{n=0}^{N-1} x(n)\cdot W_N{}^{kn}, \quad W_N = \exp\left(\frac{2\pi}{N}\right),$$

$k=0,1,\ldots N-1$. Because of Hermitian symmetry real-valued DFT algorithms compute samples for $0 \le k \le N/2$ only, which gives twofold reduction of memory requirements and even more than twofold reduction of computational burden [5].

The N-point DCT for data $x(n)$ can be computed by a DFT algorithm of size $4N$ for input vector formed as follows:

$$\begin{aligned} &x_F(2N+2n) = x_F(2n) = 0, \quad n = 0,1,\ldots,N-1; \\ &x_F(2n+1) = x(n), \\ &x_F(4N-2n-1) = x(n). \end{aligned} \quad (2)$$

Namely, DFT for symmetric data transforms into DCT-I [4], and by setting to zero even input DFT samples we get appropriate subsampling of the transform kernel:

$$X_F(k) = \sum_{n=0}^{2N-2} x_F(2n+1)\cos\left[(2n+1)k\frac{2\pi}{4N}\right] =$$
$$= 2\sum_{n=0}^{N-1} x(n)\cos\left[(n+\tfrac{1}{2})k\frac{\pi}{N}\right] = 2\cdot X(k), \quad k = 0,1,\ldots,N-1.$$

Then, for obtaining correct DCT samples it suffices to normalize samples $X_F(k)$ (1), [4].

Let us assume that N is odd, and that the $4N$-point DFT algorithm is the $4\times N$ PFA FFT [6]. The most widely used input data permutation pattern for this algorithm is:

$$x_F(n'') \leftarrow x_F(n'\cdot N + n\cdot 4), \qquad (3)$$

n'=0,1,2,3, n''=0,1,...,4N-1, index computations are done modulo N. The structure of the algorithm is shown in Fig.1. Value $n'\cdot N + n\cdot 4$ is odd only for n' odd, hence, N-point DFTs for n' even in Fig.1 have zero inputs and are obsolete (2). Input sample indices to N-point DFTs for $n'=1,3$ are: *N, N+4, N+8,...* and *3N, 3N+4, 3N+8,...* respectively. If we revert order of input samples for $n'=3$, then we have *3N, 3N-4, 3N-8, ...* Notice that

$$x_F(3N - n\cdot 4) = x_F[4N - (N + n\cdot 4)] = x_F(N + n\cdot 4),$$

indices are taken modulo $4N$. This means that the input to N-point transform for $n'=3$ is identical but reverted with respect to that for $n'=1$, hence, N-point DFT samples for $n'=3$ form complex conjugate pairs with those for $n'=1$. Concluding, computation of the N-point DFT for $n'=3$ is obsolete, too.

It is shown in Fig.2 what are simplifications in the stage of 4-point DFTs resulting from these symmetries, $X_N(k)$, $X_4(k')$ are samples of the N-point DFT for $n'=1$, and the 4-point DFT for some k, respectively, *k'=0,1,2,3, k''=0,1,...N-1.* As can be seen, indeed, computation of N-point DCT can be done by N-point DFT algorithm for appropriately permuted data samples, moreover, real-valued version of DFT algorithm suffices. The permutation pattern is (2), (3):

$$x(n) \leftarrow x[2n + (N-1)/2], \quad 0 \le n \le (N-1)/4,$$
$$x(n) \leftarrow x[(3N+1)/2 - 1 - 2n], \quad (N-1)/4 < n \le (3N-1)/4,$$
$$x(n) \leftarrow x[2n - (3N+1)/2], \quad (3N-1)/4 < n < N.$$

We need know now the relation between the N-point DFT samples $X_N(k)$ and the DCT. In general, index for the $4\times N$ PFA FFT sample $X_F(k'')$, $k''=0,1,...,4N-1$, can be computed from indices of 4- and N-point DFTs by the Chinese Remainder Theorem [6], which means that they are related as follows:

$$k'' \bmod N = k, \quad k'' \bmod 4 = k',$$

see Fig.1. For k'' indices up to N pairs of (k, k') indices are: (0,0), (1,1), (2,2), (3,3), (4,0), (5,1), ..., in general (*k, k mod 4*). Fig.2 reveals that $X(k)$ samples (1) can be obtained from the N-point real-valued DFT, and form the following pattern:

$$..\mathrm{Re}\{X(4i)\}, \mathrm{Im}\{X(4i+1)\}, -\mathrm{Re}\{X(4i+2)\}, -\mathrm{Im}\{X(4i+3)\}..$$

where $0 \le i < N/4$, if *4i+k'>N/2*, then take *Re{X(N-4i-k')}= Re{X(4i+k')}*, or *Im{X(N-4i-k')}=-Im{X(4i+k')}*.

Derivation of the DST algorithms for odd N is analogous, the main difference is that we are forming an anti-symmetric input vector to the $4N$-point DFT, i.e. (2):

$$x_F(4N - 2n - 1) = -x(n). \qquad (4)$$

Notice that the imaginary part of the DFT kernel has negative sign: $\mathrm{Im}\{W_M\} = -\sin 2\pi/M$, and that the range of DST indices is $k=1,2,...,N$. A DST algorithm can be also obtained from a DCT one by appropriate data sign changes, see [1].

3. Generel case

The above derivation forms a good starting point for constructing DCT algorithms of any size. The size N can be factorized: $N = 2^s \cdot q$, where q is an odd number, hence the vector $x_F(n'')$ (2) is processed by the $4N = 2^{s+2}\cdot q = Kq$-point DFT. We choose the $K\times q$ PFA FFT having input data permutation pattern:

$$x_F(n'') \leftarrow x_F(n'\cdot q + n\cdot K), \quad n' = 0,1,...,K-1; \qquad (5)$$

index computations are done modulo $4N$ [6]. Algorithm structure is as in Fig.1, only the N-point algorithms are replaced by the q-point ones, size of output DFTs is now K, and *n',k'=0,1,...,K-1.* We can verify that q-point DFTs for even n' are obsolete (have all zeros at their inputs), similarly as those for *n',k'>K/2*, as their outputs are complex conjugate to samples of DFTs number *K-k'*.

These symmetries imply how the K-point DFTs can be pruned. From Fig.2 we can see that input K-point DFT samples for even indices are zero, while the others exhibit Hermitian symmetry: $x_K(K-n') = x_K^*(n')$, star denotes complex conjugate. This pattern is described in equations (2), (4), hence, instead of K-point DFT computation real part of vector can be processed by the $K/4$-point DCT, and imaginary part by the $K/4$-point DST.

The details are as follows: $K/4$-point DCT computes K-point DFT samples of a symmetrical vector formed as in (2) for indices *k'=0,1,...,K/4-1*, hence, real parts of DFT samples, $K/4$-point DST computes DFT samples of an anti-symmetrical vector (4) for *k'=1,2,...,K/4*, hence, the imaginary DFT part[1]. For $k=0$ samples of the q-point DFTs are real-valued, hence, there is no $K/4$-point DST for them. The samples of the K-point DFT for *k'=K/4+1, K/4+2,..., 3K/4* can be obtained by reverting signs of appropriately chosen remaining ones. Their indices can be deduced from the derivation of M-point radix-2 decimation-in-time FFT: at its very end DFT components for even and odd data samples are combined [6]:

$$X(m) = X_{even}(m) + X_{odd}(m), \quad m = 0,1,...,M/2-1,$$
$$X(m + M/2) = X_{even}(m) - X_{odd}(m),$$

here $X_{even}(m) = 0, M = K$. Finally, we can reduce computation burden by using real-valued q-point DFT algorithms.

The input permutation pattern to the obtained in this manner DCT for any N is (2), (5):

$$x(n) \leftarrow x[n\cdot K/2 + (n'\cdot q - 1)/2], \quad (n\cdot K + n'\cdot q) \bmod 4N < 2N,$$
$$x(n) \leftarrow x[2N - 1 - n\cdot K/2 - (n'\cdot q - 1)/2] \quad \text{otherwise},$$

index computations are done modulo N, $n' = 1,3,...,K/2-1$. The formula for output $4N$-point DFT permutations comes out from the Chinese Remainder Theorem [6], hence, the relations between $4N$-, q-, and K-point DFT indices are:

$$k'' \bmod q = k, \quad k'' \bmod K = k',$$

This means that N-point DCT samples ($X(k)$ in (1)), being equal to first N samples of the $4N$-pont DFT are obtained from q-point DFTs and K-point DFTs indexed by pairs (*k'' mod q, k'' mod K*), which is analogous to the result from the previous section.

[1] Discrepancy between kernel signs of DST, and imaginary part of DFT induces careful data sign considerations here.

4. Other possibilities

What we have just derived is the DCT algorithm analogous to the PFA FFT. Similarly as for the DFT, we can use it as a basis for construction of WFTA-like DCT algorithm. We start with splitting computations inside q-point DFTs into those for real and imaginary part of the transform. This can be done by forming input data symmetric and anti-symmetric vectors, processed then by DCT I, and DST I algorithms [1, 4]. If derived carefully, this computation scheme requires the same amount of arithmetical operations as real-valued DFT algorithm [5]. Then two two-dimensional-like structures emerge: DCT I algorithms computing real part of q-point DFTs, followed by $K/4$-point DCTs, and DST I algorithms for imaginary part of these DFTs followed by $K/2$-point DSTs. We can now nest these structures [6].

Another possibility is to come back from prime-factor type algorithm to a common factor one. In [5] it has been shown that PFA FFT can be obtained from CFA FFT by shifting parts of input (or output) data in such a way that rotation factors become equal to 1. We can do a step back here, and shift, or in general permute input data to q-point DFTs. If chosen carefully, the operation may simplify input and/or output permutations of the algorithm, the price is $0(N)$ more operations linked with reappearance of rotation factors.

5. Optimality of the algorithms

In [1] it has been shown how to construct one of the best known DFT algorithm of size $4N$ for N being a power of number 2 from an N-point DCT one. This has meant that the derived in that paper DCT algorithm has been the best known one, too. Namely, if it has existed a DCT algorithm requiring less arithmetical operations, then by the same construction we would be able to derive a DFT algorithm better than the split-radix FFT from [1].

We will use the construction from [1] for proving optimality of the DCT algorithms derived in this paper. Firstly, a DCT algorithm can be transformed into a DST one, and vice versa, by some data sign changes, hence, we need not a DST algorithm when we have a DCT one. When constructing the $4N$-point DFT algorithm from an N-point DCT one, a pair of DCT and DST algorithms process symmetric and anti-symmetric data vectors of size N formed from data samples for odd indices, their generation is done by $2N$ additions/subtractions. Data for even indices are transformed by a $2N$-point DFT, and then results combined with those from DCT/DST by using $4N$ additions/subtractions. For odd N the $2N$-point DFT can be computed by the $2\times N$ PFA FFT, in which the 2-point DFTs require $2N$ additions/subtractions. Summarizing, for odd N we need arithmetical operations of 4 DFT algorithms of size N (two of them do DCT/DST computation[1]), plus $8N$ auxiliary additions/subtractions. On the other hand, $4N$-point DFT for odd N can be computed by $4\times N$ PFA FFT containing 4 DFT algorithms of size N, and N DFT algorithms of size 4, the latter requiring in total $8N$ additions/subtractions, too. If it exists a better DCT algorithm than that described in the paper, then we are obtaining a better DFT algorithm than the $4\times N$ PFA FFT, which is unlikely.

In general, inside the $4N$-point DFT algorithm constructed from the N-point DCT one we have $4N/q$ DFTs of size q, $2q=8N/K$ DCT/DST transforms of size $K/4$ (inside two N-point DCT algorithms), and $4N/K$ DFTs of size $K/2$, if the $2N$-point DFT for even data samples is computed as the $K/2\times q$ PFA FFT, plus $6N$ auxiliary additions/subtractions. If we compute the $4N$-point DFT as the $K\times q$ PFA FFT, then we have $4N/q$ DFTs of size q, and q DFTs of size K, where each of the latter transforms, if computed in accordance with [1], contains 2 DCTs of size $K/4$, one DFT of size $K/2$, and $3K/2$ auxiliary additions/subtractions. We can verify that total counts of transforms/operations are the same for both algorithms. It can be shown that optimized DFT modules for $K=8,16$ [6] are constructed in accordance with [1], hence, the conclusion is true also for $4N$-point WFTA and DFT algorithm constructed from the N-point WFTA-like DCT algorithm.

6. Conclusion

It has been shown in the paper that the discrete cosine and sine transforms (DCT and DST) can be computed by slightly modified DFT algorithms for real-valued data of the same size. When transform size is odd, the modification consist only in changed input/output data permutations. In general, when the transform size is $q2^s$, q being odd, in addition to it 2^s-point DFT algorithms in the PFA FFT structure should be replaced by those for DCTs and DSTs. The PFA FFT-like DCT/DST algorithm can be then transformed into the WFTA-like, or CFA FFT-like one, hence, it appears that for any important DFT algorithm its DCT/DST counterpart can be derived. Finally, it has been shown that the new DCT/DST algorithms require the smallest known numbers of arithmetical operations.

References

[1] Vetterli M., and Nussbaumer. H.J. "Simple FFT and DCT algorithms with reduced number of operations". *Signal Processing,* Vol. 6, 1984, pp. 267-278.

[2] Bi G., and Yu L.W. "DCT algorithms for composite sequence length". *IEEE Trans. Signal Processing*, Vol. 46, 1998, pp. 554-562.

[3] Stasinski R. "Optimal DCT algorithms for any data block size". *Nordic Signal Processing Conference (NORSIG-96)*, Tampere, Finland, 1996.

[4] Wang Z. "Fast algorithms for the discrete W transform and for the discrete Fourier transform". *IEEE Trans. Acoust., Speech, Signal Proces.*, Vol. 32, 1984, pp. 803-816.

[5] Stasinski R. "The techniques of the generalized fast Fourier transform algorithm". *IEEE Trans.Signal Processing*, Vol. 39, 1991, pp. 1058-1069.

[6] Blahut R.E. *Fast algorithms for digital signal processing*. Addison-Wesley, Reading MA, 1985.

[1] For complex data symmetric and anti-symmetric DFT parts are added/subtracted, but this is compensated by operation savings in real-valued FFTs.

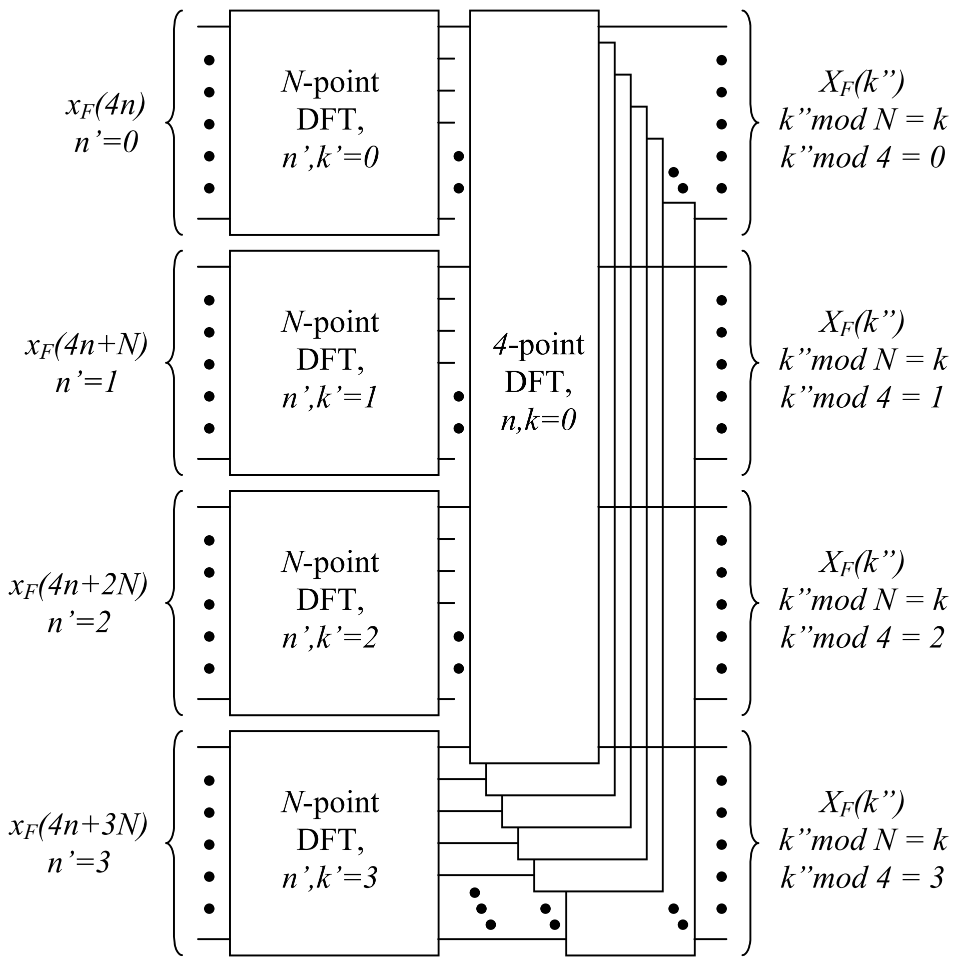

Fig. 1.: Structure of the *4×N* PFA FFT, *N* is odd, for explanations see text.

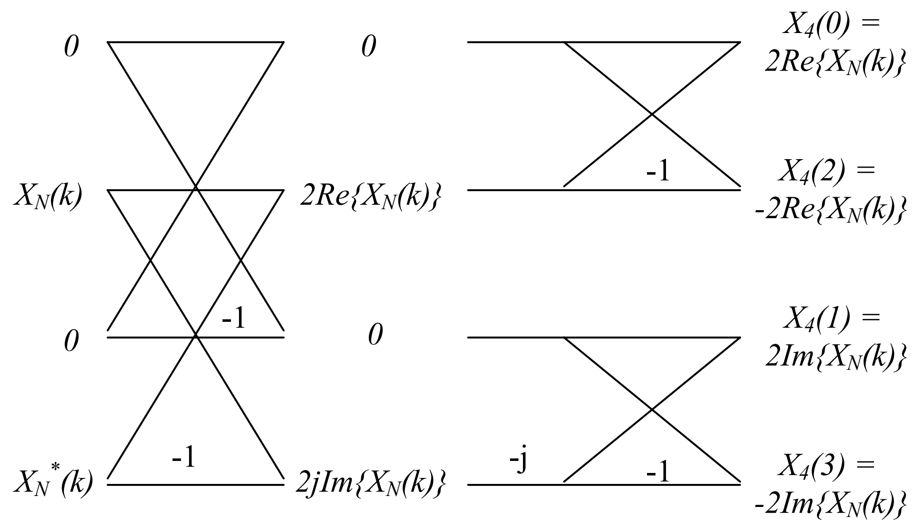

Fig.2: Example of output 4-point DFT algorithm of the *4×N* PFA FFT, and how it processes data when *N*-point DCT is computed.