4.5 Discrete Orthogonal Transforms

There are three main reasons to apply signal transformations. One is that in the transform domain the information content of the signal to be processed has, or may have a better interpretation. Typically this is the case with the Fourier-type transformations where the concept of frequency provides a direct physical background for the transformation. A second reason can be simply computational efficiency: certain transform domain calculations give better overall performance. Finally the third main reason is that certain transformations can help in data reduction problems.

In this Section the most important one-dimensional linear signal transforms are characterized and investigated from the viewpoints of the possible implementations. These transforms can be considered as sequence to vector (S-V) and vector to sequence (V-S) transformers, respectively. Here the vectors give the transformed domain representation of the discrete time sequence. The S-V transformer is often used for signal analysis while the V-S direction for synthesis.

As a basic introduction Subsection 2.4.1 gives an overview of signal representation techniques based on linear transformations. From this overview it turns out that from mathematical point of view a discrete transformation is simply a matrix-vector product where the vector of size N consists of the subsequent input samples while the matrix of size $N * N$ contains the weighting factors determined for the actual transformation in hand. The inherent parallelism of such an operation offers very efficient ways of implementation and in some cases certain symmetry properties of the transformation matrices may provide further improvements. Among these possibly the advent of the fast transformation algorithms could cause the largest breakthrough. Subsection 2.4.2 is devoted to fast transformation algorithms and their implementation.

The interpretation of transformations as a matrix-vector products supposes that the whole signal "manipulation" is block oriented and is performed as a "single" operation. In practical application it may happen that the input data to be processed is a continuous flow, and it would be of interest to follow at least a certain portion (e.g. a block of N subsequent samples) of this sequence, and to calculate its transform domain equivalent in every step. Obviously it is always possible to consider every block in every step separately, however, recursion techniques can provide more efficient solutions. In Subsection 2.4.3 this problem is investigated and the so-called sliding window technique is introduced and applied for the evaluation of transforms.

Block-oriented fast algorithms and time-recursive transformers can be combined in an efficient way. These techniques are summarized in Subsection 2.4.4.

4.5.1 Overview of typical discrete orthogonal transforms

This Subsection provides a basic introduction to discrete signal transformations. The discrete signals are represented as simple sequences (typically) of limited length, while transformations are performed as simple multiplications by invertible matrices. The transformed version of a discrete sequence is an alternative representation within the vector space of discrete sequences. This alternative representation may better emphasize some of the features of the signal in hand, therefore some of the further signal processing operations might have lower complexity. Discrete signal transformations are (typically) block-oriented operations, i.e. the complete input sequence will have an alternative representation of the same size. This feature, however, may be problematic for larger amount of input data, or similarly in the case of continuous input data flow. In such situations a good compromise can be the decomposition of the input data into (possibly overlapping) blocks of smaller size, and the application of a series of transformations on these blocks. This block-orientation limits the "resolution" of the transform domain representation on one hand, but on the other it is better suited to the transform domain characterization of nonstationary signals.

Signal representations in vector spaces

The characterization or modeling of a signal can be performed with the help of known signals. This kind of description is typically an approximation, and usually called as signal representation. Due to the techniques applied during approximation there are several common aspects of these different representations. A very interesting interpretation can be given if this representation is made in so-called signal spaces. The signal spaces can be considered as generalizations of the Euclidean space, where every point within the space represents a signal. The concept of distance remains valid, and this distance can be used to characterize the relation of the different signals. The concepts of metric, algebraic operations, linearity, norm, and scalar, or inner product play important role in typical signal spaces. With this latter the orthogonality of two signals can also be defined. All these concepts can be extended to stochastic signals, too. In the case of discrete signals the signal space is simply a vector space with the elementary vector algebraic operations.

In the following development vector spaces of dimension N will be considered. This means that all the vectors consisting of real or complex numbers are of this dimension, and all the transformations within this space can be given as invertible $N * N$ matrices. Every point within this space represents a discrete signal, and can be described by its coordinates. If a transformation is applied to a certain element of the vector space the resulting new vector will express the very same information within another coordinate system. This second coordinate system is defined by N linearly independent vectors (called basis system) of the very same vector space, and the new coordinates are given by the transformation. It is obvious that the change of the coordinate system may support further processing like feature extraction, etc. depending on the basis applied. The representation of the signal $\underline{y} = [y_0, y_1, \dots, y_{N-1}]^T$ within the coordinate system defined by the linearly independent vectors $\underline{c}_m = [c_{0m}, c_{1m}, \dots, c_{N-1,m}]^T$, $m = 0, 1, \dots, N-1$, is given by

$$\underline{y} = \sum_{m=0}^{N-1} x_m \underline{c}_m \tag{4.1}$$

where the $\{x_m\}$ values, $m = 0, 1, \dots, N-1$, are the coordinates of $\underline{y}$. The equivalent form in matrix formulation is

$$\underline{y} = \underline{\underline{C}}\, \underline{x} \tag{4.2}$$

where $\underline{x} = [x_0, x_1, \dots, x_{N-1}]^T$, and $\underline{\underline{C}} = [\underline{c}_0, \underline{c}_1, \dots, \underline{c}_{N-1}]$, i.e. the basis vectors are the columns of matrix $\underline{\underline{C}}$. The representation of $\underline{y}$ in the system of $\underline{\underline{C}}$ is $\underline{x}$:

$$\underline{x} = \underline{\underline{C}}^{-1} \underline{y} = \underline{\underline{T}}\, \underline{y} \tag{4.3}$$

where $\underline{\underline{T}}$ denotes the transformation matrix. The columns of $\underline{\underline{T}}$ form another basis system in which the representation of $\underline{x}$ is given by the coordinate vector $\underline{y}$. The rows of $\underline{\underline{T}}$, however, form the so-called reciprocal basis system ($\underline{\underline{T}} = [\underline{t}_0, \underline{t}_1, \dots, \underline{t}_{N-1}]^T$, where $\underline{t}_m = [t_{m0}, t_{m1}, \dots, t_{m,N-1}]^T$). The vectors of the basis and reciprocal basis systems are related by:

$$\underline{t}_m^T \underline{c}_n = \delta_{mn} \ , \qquad m, n = 0, 1, \dots, N-1 \tag{4.4}$$

where $\delta_{mn} = 1$, if $m = n$, otherwise 0. In the above development n is used as index of the "source", while m as "destination" (transform) domain index. From (4.2) and (4.3) it is trivial that $\underline{\underline{C}}$ is the matrix of the so-called inverse transformation.

For the well-known case of the DFT the values of $\underline{\underline{T}}$ and $\underline{\underline{C}}$ are as follows:

$$t_{mn} = \frac{1}{N}\, e^{-j\,\frac{2\pi}{N}\,mn} \ , \qquad c_{nm} = e^{j\,\frac{2\pi}{N}\,mn} \ , \qquad n, m = 0, 1, \dots, N-1 \ . \tag{4.5}$$

Here n and m are the discrete time and discrete frequency indices, respectively.

In practice transformations with special properties are applied. An important family is the set of unitary transforms (See e.g. [AR75]). For these the calculation of the reciprocal basis vectors is rather simple, because

$$\underline{\underline{T}} = \underline{\underline{C}}^{-1} = \alpha \underline{\underline{C}}^* \ . \tag{4.6}$$

Here α is scalar multiplier. (As an example see e.g. (4.5), where $\alpha = 1/N$). This scalar multiplier is simply a normalizing factor, and can be easily avoided if it is "distributed" between the basis and reciprocal bases. (For the Fourier case $\sqrt{1/N}$ should be applied). The members of the family of unitary transforms have orthogonal basis system, i.e.:

$$\alpha \underline{c}_m^* \underline{c}_n = \delta_{mn} \ , \qquad m, n = 0, 1, \ \dots \ , N-1 \tag{4.7}$$

and from (4.6) it turns out that the same is true for the reciprocal basis system. The systems with $\alpha = 1$ are called orthonormal bases. With such transformations we can decompose a discrete signal $\underline{y}$ into orthogonal components. This decomposition is in correspondence with the series expansion of signals. The coefficients in such series expansions are the coordinate values within the coordinate system of the basis vectors applied, and finally the weighted basis vectors represent the components of the signal decomposition.

There are very many alternative notational conventions for discrete transformations in the literature. The general framework introduced above is well suited to the needs of the different approaches, however, is not widely used yet. For the convenience of the reader in the subsequent subsections the notations try to follow the typical forms. In the case of transformations the input sequence will be denoted by $\underline{x}$, and its transform by $\underline{X}$. Their time-dependency can be emphasized by $\underline{x}(n)$ and $\underline{X}(n)$. Here $\underline{X}$ and $\underline{X}(n)$ consist of the coefficients of the series expansions described above. For filter-banks performing sliding transformations $\underline{x}$, $\underline{x}(n)$ and $\underline{Y}$, $\underline{Y}(n)$ denote the input, and the output vectors, respectively. Here the outputs represent the actual samples of the signal components which coincide with the transform of the last N input samples. A further alternative notation will be $\underline{g}_m = \underline{t}_m$, $m = 0, 1, \ \dots \ , N-1$, for the reciprocal basis system to emphasize the relation of some signal transformer structures to a family of observers also by this notation.

The most important discrete orthogonal transforms

DFT: Discrete Fourier Transform

$$t_{mn} = \frac{1}{N} e^{-j\frac{2\pi}{N}mn} \; ; \;\; c_{mn} = e^{j\frac{2\pi}{N}mn} \, , \qquad m,n = 0,1,\ \dots ,N-1 \qquad (4.8)$$

Note: The normalizing factor $\frac{1}{N}$ can be used also with the inverse transform or distributed as $\sqrt{1/N}$.

c-DFT: Complementary Discrete Fourier Transform

$$t_{mn} = \frac{1}{N} e^{-j\frac{(2m+1)\pi}{N}n} \; ; \;\; c_{mn} = e^{j\frac{(2m+1)\pi}{N}n} \, , \qquad m,n = 0,1,\ \dots ,N-1 \quad (4.9)$$

DHT: Discrete Hartley Transform

$$\begin{aligned} t_{mn} &= \tfrac{1}{N}\mathrm{cas}(\tfrac{2\pi}{N}mn) = \tfrac{1}{N}\left[\cos(\tfrac{2\pi}{N}mn) + \sin(\tfrac{2\pi}{N}mn)\right] \; ; \\ c_{mn} &= \tfrac{1}{N}\mathrm{cas}(\tfrac{2\pi}{N}mn) \, , \qquad\qquad m,n = 0,1,\ \dots ,N-1 \end{aligned} \qquad (4.10)$$

DCT-I: Discrete Cosine Transform-I

$$t_{mn} = c_{mn} = \left(\begin{array}{ll} \sqrt{\frac{1}{N}} \, , & m,n = 0 \\ \sqrt{\frac{2}{N}} \cos(\frac{\pi}{N}mn) \, , & m,n = 1,2,\ \dots ,N-1 \end{array} \right. \qquad (4.11)$$

DST-I: Discrete Sine Transform-I

$$t_{mn} = c_{mn} = \sqrt{\frac{2}{N}} \sin\left(\frac{\pi}{N}mn\right) \, , \qquad m,n = 1,2,\ \dots ,N-1 \qquad (4.12)$$

DCT-II: Discrete Cosine Transform-II

$$t_{mn} = c_{mn} = \begin{cases} \sqrt{\frac{1}{N}}\,, & m = 0 \\ \sqrt{\frac{2}{N}} \cos\left[\frac{\pi}{N} m(n+0.5)\right]\,, & n = 0;\ m,n = 1,2,\ \dots,N-1 \end{cases} \tag{4.13}$$

DST-II: Discrete Sine Transform-II

$$t_{mn} = c_{mn} = \begin{cases} \sqrt{\frac{2}{N}} \sin\left[\frac{\pi}{N} m(n+0.5)\right]\,, & n = 0;\ m,n = 1,2,\ \dots,N-1 \\ (-1)^n \sqrt{\frac{1}{N}} & m = N \end{cases} \tag{4.14}$$

DCT-III: Discrete Cosine Transform-III

$$t_{mn} = c_{mn} = \begin{cases} \sqrt{\frac{1}{N}}\,, & n = 0 \\ \sqrt{\frac{2}{N}} \cos\left[\frac{\pi}{N} n(m+0.5)\right]\,, & m = 0; \\ & m,n = 1,2,\ \dots,N-1 \end{cases} \tag{4.15}$$

DST-III: Discrete Sine Transform-III

$$t_{mn} = c_{mn} = \begin{cases} \sqrt{\frac{2}{N}} \sin\left[\frac{\pi}{N} n(m+0.5)\right]\,, & m = 0;\ m,n = 1,2,\ \dots,N-1 \\ (-1)^m \sqrt{\frac{1}{N}}\,, & n = N \end{cases} \tag{4.16}$$

DCT-IV: Discrete Cosine Transform-IV

$$t_{mn} = c_{mn} = \sqrt{\frac{2}{N}} \cos\left[(n+0.5)(m+0.5)\frac{\pi}{N}\right] , \qquad m,n = 1,2,\ \dots ,N-1 \tag{4.17}$$

DST-IV: Discrete Sine Transform-IV

$$t_{mn} = c_{mn} = \sqrt{\frac{2}{N}} \sin\left[(m+0.5)(n+0.5)\frac{\pi}{N}\right] , \qquad m,n = 0,1,2,\ \dots ,N-1 \tag{4.18}$$

WHT: Walsh-Hadamard Transform

$$t_{mn} = \frac{1}{N}\mathrm{wal}(m,n) \ ; \ \ c_{mn} = \mathrm{wal}(m,n) \tag{4.19}$$

where

$$\mathrm{wal}(m,n) = (-1)^{<m,n>} \ \text{ with } \ <m,n> = \sum_{k=0}^{M-1} m_k n_k \ , \ \ M = \log_2 N \ , \tag{4.20}$$

in which the terms m_k and n_k are the coefficients of the binary representation of m and n, respectively.

Note: In (4.20) the so-called natural or Hadamard ordering is applied [AR75].

From the above list of important transformations for signal representation the DFT is the most important tool because of the physical meaning of the Fourier domain. Some other transforms may be useful if the efficiency of the calculations is of interest. The Walsh-Hadamard transform seems to be the more efficient in this respect since except the normalizing factor of $1/N$ no multiplications are required. If only a few transform domain coefficients are to be calculated then an indirect calculation scheme presented on Fig. 4.1 can have practical importance (see e.g. [TH78]).

KLT: Karhunen-Loève Transform

An important application of the orthogonal transforms is data compression. The key to securing data compression is signal representation, which

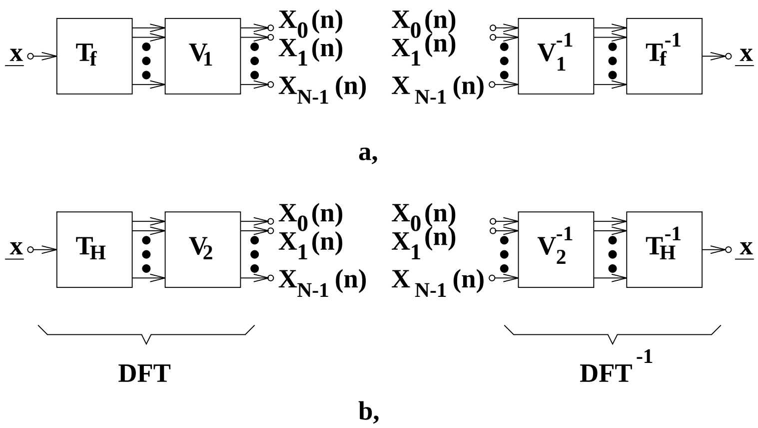


Figure 4.1: Indirect implementation of transformations: a, Implementation of transformations via fast transformations, b, implementation of the DFT via the Walsh-Hadamard transformation

concerns the representation of a given class (or classes) of signals in an efficient manner. For more efficient representations, one obtains an orthogonal transform of $\underline{X}$ which results in $\underline{Y} = \underline{\underline{T}}\,\underline{X}$. The objective is to select a subset of M components of $\underline{Y}$, where M is substantially less than N. The remaining $(N-M)$ components can then be discarded without introducing objectionable error, when the signal is reconstructed using the retained M components of $\underline{Y}$. the orthogonal transforms must therefore be compared with respect to some error criterion. One such often used criterion is the mean-square error criterion.

We seek the orthogonal transform that enables signal representation and yet is optimum with respect to the mean-square error criterion.

Let $\underline{\underline{T}}$ be an orthogonal transform given by

$$\underline{\underline{T}}^T = [\underline{t}_0, \underline{t}_1, \;\cdots, \underline{t}_{N-1}] \tag{4.21}$$

where the $\underline{t}_m$ are N-vectors. For convenience, the basis vectors $\{\underline{t}_m\}$ are assumed to be real-valued and orthonormal – i.e.,

$$\underline{t}_m^T \underline{t}_n = \begin{cases} 1, & m = n \\ 0, & m \neq n \end{cases} \tag{4.22}$$

For each vector $\underline{X}$ belonging to a given class of data vectors, we obtain

$$\underline{Y} = \underline{\underline{T}}\,\underline{X} \tag{4.23}$$

where $\underline{X}^T = [x_0, x_1, \ \dots \ , x_{N-1}]$ and $\underline{Y}^T = [y_0, y_1, \ \dots \ , y_{N-1}]$.

From equations (4.21) and (4.22) it follows that $\underline{\underline{T}}^T \underline{\underline{T}} = \underline{\underline{I}}$, and hence

$$\underline{X} = \underline{\underline{T}}^T \ \underline{Y} = [\underline{t}_0, \underline{t}_1, \ \dots \ , \underline{t}_{N-1}]\underline{Y} \tag{4.24}$$

which yields

$$\underline{X} = y_0\underline{t}_0 + y_1\underline{t}_1 + \ \dots \ + Y_{N-1}\underline{t}_{N-1} = \sum_{i=0}^{N-1} y_i\underline{t}_i \ . \tag{4.25}$$

We wish to retain a subset $\{y_0, y_1, \ \dots \ , y_M\}$ of the components of $\underline{Y}$ and yet estimate $\underline{X}$. This can be done by replacing the remaining $N - M$ components of $\underline{Y}$ by preselected constants b_i to obtain

$$\tilde{\underline{X}}(M) = \sum_{i=0}^{M-1} y_i\underline{t}_i + \sum_{i=M}^{N-1} b_i\underline{t}_i \tag{4.26}$$

where $\tilde{\underline{X}}(M)$ denotes the estimate of $\underline{X}$.

The error introduced by neglecting the $N - M$ terms can be represented as

$$\Delta\underline{X} = \underline{X} - \tilde{\underline{X}}(M) \tag{4.27}$$

where $\Delta\underline{X}$ is the error vector. That is

$$\Delta\underline{X} = \underline{X} - \sum_{i=0}^{M-1} y_i\underline{t}_i - \sum_{i=M}^{N-1} b_i\underline{t}_i \tag{4.28}$$

From equations (4.25) and (4.28) it follows that

$$\Delta\underline{X} = \sum_{i=M}^{N-1} (y_i - b_i)\underline{t}_i \tag{4.29}$$

Thus the mean-square error $\varepsilon(M)$ is given by

$$\varepsilon(M) = \mathrm{E}\left\{x \left|\left|\Delta\underline{X}^2\right|\right|\right\} = \mathrm{E}\left\{(\Delta\underline{X})^T(\Delta\underline{X})\right\} \tag{4.30}$$

Substitution of (4.29) in (4.30) leads to

$$\varepsilon(M) = \mathrm{E}\left\{\sum_{i=M}^{N-1}\sum_{j=M}^{N-1}(y_i - b_i)(y_j - b_j)\underline{t}_i^T\underline{t}_j\right\} \tag{4.31}$$

which simplifies to yield [see eq. (4.22)]

$$\varepsilon(M) = \sum_{i=M}^{N-1} \mathrm{E}\left\{(y_i - b_i)^2\right\} \tag{4.32}$$

From equation (4.32) it follows that for every choice of $\underline{t}_i$ and b_i, we obtain a value for $\varepsilon(M)$. We seek the choice which minimize $\varepsilon(M)$. The process of choosing the optimum b_i and $\underline{t}_i$ respectively is carried out in two steps.

Step 1. The optimum b_i are obtained as follows:

$$\frac{\partial}{\partial b_i}\mathrm{E}\left\{(y_i - b_i)^2\right\} = -2\left[\mathrm{E}\left\{y_i\right\} - b_i\right] = 0 \tag{4.33}$$

which yields

$$b_i = \mathrm{E}\left\{y_i\right\} \tag{4.34}$$

Now, from equations (4.22) and (4.25) one obtains

$$y_i = \underline{t}_i \underline{X} \tag{4.35}$$

Thus we have

$$b_i = \underline{t}_i^T \mathrm{E}\left\{\underline{X}\right\} = \underline{t}_i^T \underline{\bar{X}} \tag{4.36}$$

where $\underline{\bar{X}} = \mathrm{E}\left\{\underline{X}\right\}$.

Again, since the quantity $(y_i - b_i)$ in equation (4.32) is a scalar, we can write $\varepsilon(M)$ as

$$\varepsilon(M) = \sum_{i=M}^{N-1} \mathrm{E}\left\{(y_i - b_i)^2\right\} \tag{4.37}$$

Substitution of $y_i = \underline{t}_i^T \underline{X}$ and $b_i = \underline{t}_i \underline{\bar{X}}$ in (4.37) results in

$$\varepsilon(M) = \sum_{i=M}^{N-1} \underline{t}_i^T \mathrm{E}\left\{(\underline{X} - \underline{\bar{X}})(\underline{X} - \underline{\bar{X}})^T\right\} \underline{t}_i\ . \tag{4.38}$$

Since $\underline{\underline{\Sigma}}_x = \mathrm{E}\left\{(\underline{X} - \underline{\bar{X}})(\underline{X} - \underline{\bar{X}})^T\right\}$ is the covariance matrix of $\underline{X}$, we have

$$\varepsilon(M) = \sum_{i=M}^{N-1} \underline{t}_i^T \underline{\underline{\Sigma}}_x \underline{t}_i\ . \tag{4.39}$$

Step 2. To obtain the optimum $\underline{t}_i$ we must not only minimize $\varepsilon(M)$ with respect to $\underline{t}_i$ but also satisfy the constraint $\underline{t}_i \underline{t}_i^T = 1$. Thus we use the method of Lagrange multipliers and minimize

$$\widehat{\varepsilon}(M) = \varepsilon(M) - \sum_{i=M}^{N-1} \beta_i[\underline{t}_i^T \underline{t}_i - 1] = \sum_{i=M}^{N-1} \left\{\underline{t}_i^T \underline{\underline{\Sigma}}_x \underline{t}_i - \beta_i[\underline{t}_i^T \underline{t}_i - 1]\right\} \tag{4.40}$$

with respect to $\underline{t}_i$ where β_i denotes the Lagrange multiplier.

Now, it can be shown that

$$\underline{\underline{\Sigma}}_x \underline{t}_i = \beta_i \underline{t}_i \ . \tag{4.41}$$

By definition equation (4.41) implies that $\underline{t}_i$ is the eigenvector of the covariance matrix $\underline{\underline{\Sigma}}_x$ and β_i is the i-th eigenvalue. Denoting β_i by λ_i and substituting (4.41) in (4.39), the minimum mean-square error is given by

$$\varepsilon_{min}(M) = \sum_{i=M}^{N-1} \lambda_i \ . \tag{4.42}$$

The expansion defined in (4.25) is thus in terms of the eigenvectors of the covariance matrix. This expansion is called the Karhunen-Loève expansion. The $\underline{t}_i$ which comprise $\underline{\underline{T}}$ in (4.21) are eigenvectors of $\underline{\underline{\Sigma}}_x$. Hence the transformation $\underline{Y} = \underline{\underline{T}}\ \underline{X}$ is called the Karhunen-Loève Transform (KLT).

Note: The KLT is the optimum transform for signal representation with respect to the mean-square error criterion as it follows from the above analysis.

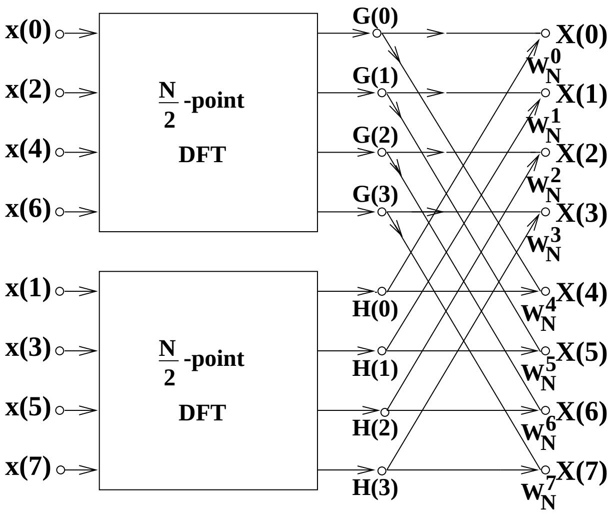


Figure 4.2: Flow graph of the decimation-in-time decomposition of an N-point DFT computation into two $N/2$-point DFT computations ($N = 8$)

From the above development it is clear that in the case of proper a priori knowledge there exists an orthogonal transformation optimal for data reduction in the "mean-square error" sense. However, this a priori knowledge is not always available. The discrete cosine transformations presented above have the important property that their basis vectors closely approximate the eigenvectors of a class of matrices called Toeplitz matrices. This class of matrices is defined as

$$\underline{\underline{\Phi}} = \begin{bmatrix} 1 & \rho & \rho^2 & \dots & \rho^{N-1} \\ \rho & 1 & \rho & \dots & \rho^{N-2} \\ \rho^{N-1} & \rho^{N-2} & \rho^{N-3} & \dots & 1 \end{bmatrix} \tag{4.43}$$

with $0 < \rho < 1$. Such matrices have important role in different signal characterizations. As a consequence of this property the DCTs are effectively used in the area of image processing typically for data reduction.

4.5.2 Fast algorithms and their implementations

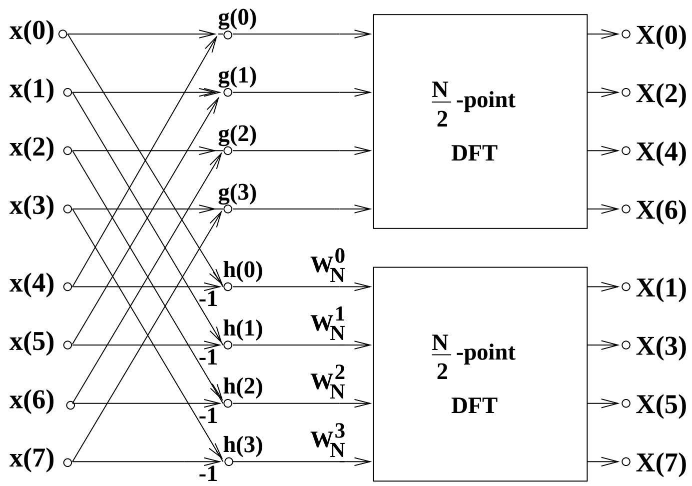


Figure 4.3: Flow graph of the decimation-in-frequency decomposition of an N-point DFT computation into two $N/2$-point DFT computations ($N = 8$)

Fast algorithms for efficient computation of the different orthogonal transforms, especially that of the DFT has accelerated the application of transformed domain techniques in digital signal processing in a number of diverse areas. In this subsection the most important fast versions are summarized with special emphasis on the fast Fourier transformation (FFT).

The key idea of the fast transformation algorithms can be understood with the help of Figs. 4.2 and 4.3. If we observe the symmetry and regularity of the discrete Fourier "kernel", i.e. that of the complex exponential, it is easy to understand that an N-point DFT can be evaluated also by a "decimation in time" strategy where we apply smaller size DFTs each of which receives a decimated input sequence. The partial results can be combined at the output. Similarly a combination of symmetrically located input data pairs followed by a smaller size DFT proves to be efficient especially if N is an integer power of two.

4.5.3 Time-recursive discrete orthogonal transforms

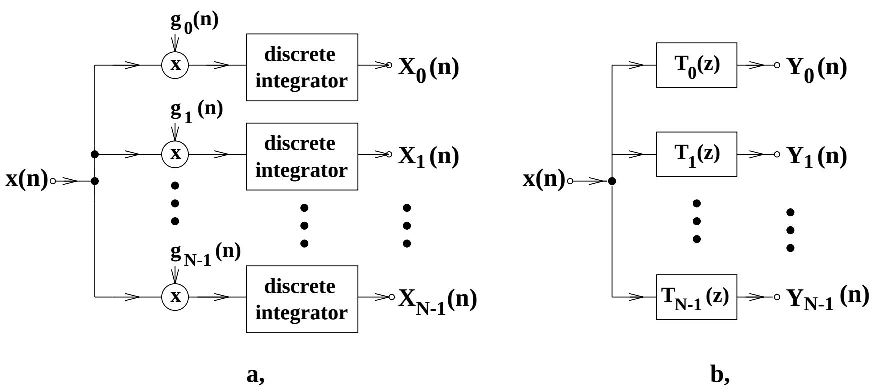


Figure 4.4: Periodic waveform signal analysis: a, By reciprocal basis system, b, by filter-banks

There are very many applications where on one hand a continuous input-data flow is to be dealt with, while on the other hand a transform-domain representation would be desirable (S-V transformer). In such cases the input vector to be transformed is composed from the last N input samples and this "window" is being shifted by one sample every time a new sample arrives. From (4.3) the transform domain representation of the vector $\underline{x}(n)$

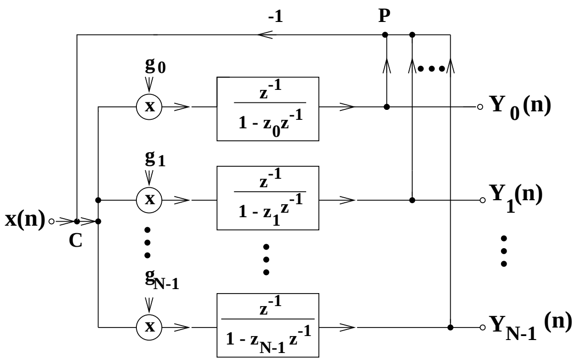


Figure 4.5: Periodic waveform signal analysis: Resonator-based DFT analyzer

composed from the last N samples can be given as

$$\underline{Y}(n) = \underline{\underline{T}}\ \underline{x}(n) \tag{4.44}$$

where $\underline{Y}(n)$ is the corresponding "sliding" transform or alternatively the output of an FIR filter-bank, where the transfer function of the mth filter has the form of

$$T_m(z) = t_{m,N-1} + t_{m,N-2}z^{-1} + \ \ldots\ + t_{m,0}z^{-(N-1)}\,, \qquad m = 0, 1,\ \ldots\ , N-1 \tag{4.45}$$

i.e., its weights are related to the mth row of $\underline{\underline{T}}$. Fig. 4.4 shows the difference between the block-oriented and the sliding-window transformations. For simplicity let's suppose periodic signals with periods of N. On Fig. 4.4.a the calculations related to the subsequent input samples can be followed. Each channel contains a multiplier and a discrete integrator. The multiplier coefficients are the corresponding components of the reciprocal basis vectors. The discrete integrator sums up the subsequent products. This system operates for $0 \le n \le N-1$, and gives complete result only for $n = N-1$. On Fig. 4.4.b the structure of an FIR analysis filter-bank is demonstrated. Each filter within the filter-bank is of length N. Here the outputs generate the signal components which correspond to the transformation of the last N input samples. This version can be operated for arbitrary $n \ge 0$.

For the case of the most widely used DFT, using (4.5) the following

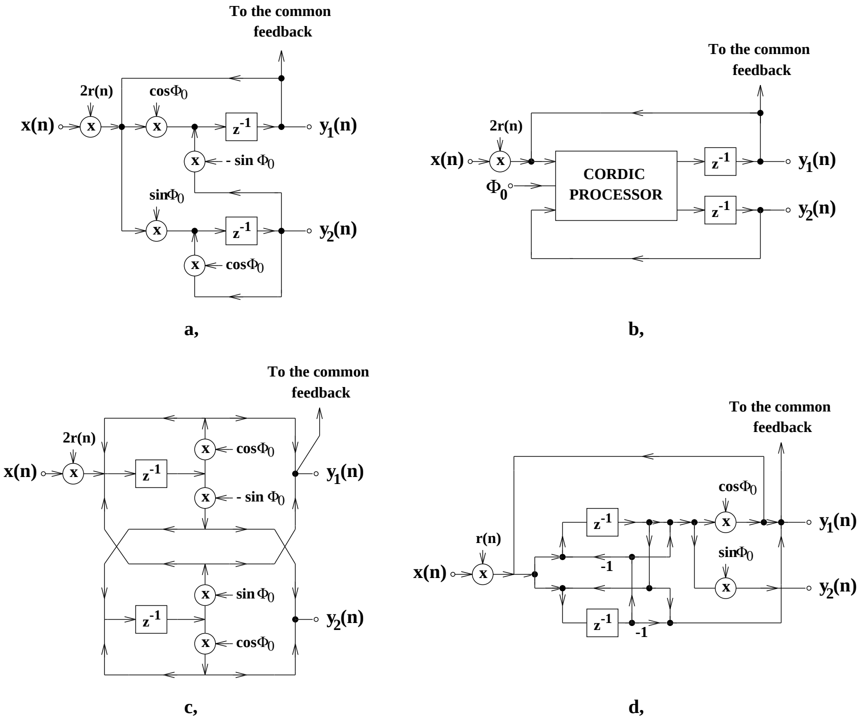


Figure 4.6: Periodic waveform signal analysis: Second-order resonator-based structures

formulation is possible

$$
\begin{aligned}
T_m(z) &= \tfrac{1}{N}\left[z_m^{-(N-1)} + z_m^{-(N-2)}z^{-1} + \ \dots\ + z^{-(N-1)}\right] = \\
&= \tfrac{z_m}{N}\ \left[1 + z_m z^{-1} + \ \dots\ + z_m^{N-1}z^{-(N-1)}\right] = \qquad (4.46)\\
&= \tfrac{z_m}{N}\ \tfrac{1-z^{-N}}{1-z_m z^{-1}}
\end{aligned}
$$

where $z_m = e^{j(2\pi m/N)}$, $m = 0, 1,\ \dots\ , N-1$. The filter-bank described with the last element of (4.3) is a special case of the Lagrange structure, well known from the early literature of digital signal processing [RG75]. The implementation of the numerator $[1 - z^{-N}]$ is extremely simple, and is common to all channels, but the first-order IIR transfer functions having poles on the unit circle may lead to numerical problems under finite precision conditions.

An interesting development based on the observer theory [Hos80] has led to an alternative structure [Pec86], [PMP96] presented on Fig. 4.5. This structure can be characterized as a set of parallel, complex coefficient, first-order resonators within a common feedback loop. The resonator poles are located on the unit circle correspondingly to the Nth roots of unity. The weighting coefficients can be calculated by

$$g_m = \frac{1}{N} z_m \,, \qquad m = 0, 1, \, \dots \,, N-1 \tag{4.47}$$

and the transfer function by

$$\begin{aligned} T_m(z) &= \frac{1}{N} \frac{\frac{z_m z^{-1}}{1 - z_m z^{-1}}}{1 + \frac{1}{N} \sum_{n=0}^{N-1} \frac{z_n z^{-1}}{1 - z_n z^{-1}}} = \frac{z_m z^{-1}}{N} \frac{1 - z^{-N}}{1 - z_m z^{-1}} = \\ &= \frac{1}{N} \left(z_m z^{-1} + z_m^2 z^{-2} + \, \dots \, + z_m^N z^{-N} \right) \end{aligned} \tag{4.48}$$

which gives the mth DFT coefficient of the last N samples.

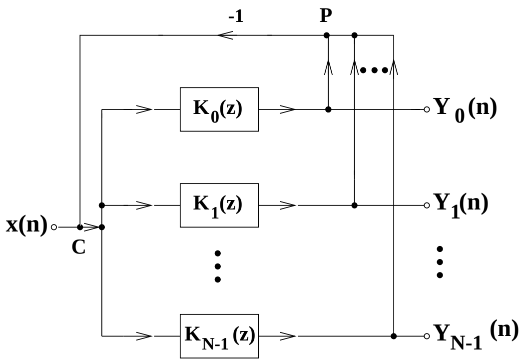


Figure 4.7: Periodic waveform signal analysis: Block-scheme of a resonator-based analyzer

The resonators on Fig. 4.5 are given in their first-order, complex coefficient form. For practical implementations these are replaced by real coefficient, second-order versions which produce the real and imaginary parts of two transformed domain coefficients being in complex conjugate relation. Starting from the state variable description of the resonators very many alternatives can be derived. Concerning secondary effects the behavior of the various versions can be different. Fig. 4.6 shows those versions which have relatively good performance. Using a systematic quantization strategy these versions have zero input limit cycle immunity and the low roundoff noise [Pec89].

Starting from the DFT analyzer of Fig. 4.5, and forming simply a linear combination, signal decompositions according to another basis system is also possible [PF90]. The block diagram of such a system is given in Fig. 4.7, where $K_m(z)$, $m = 0, 1, \ldots, N-1$, is a linear combination of resonator transfer functions having the form of $g_n z^{-1}/(1 - z_n z^{-1})$, $n = 0, 1, \ldots, N-1$. Similarly to (4.48) the transfer functions of the channels will implement FIR filters with impulse response defined by the mth row of the transformation matrix.

In the case of arbitrary transformations the direct utilization of the basis-reciprocal basis systems can also be considered as it is done on Fig. 4.4.a. Such a system is given on Fig. 4.8. In this system both the series

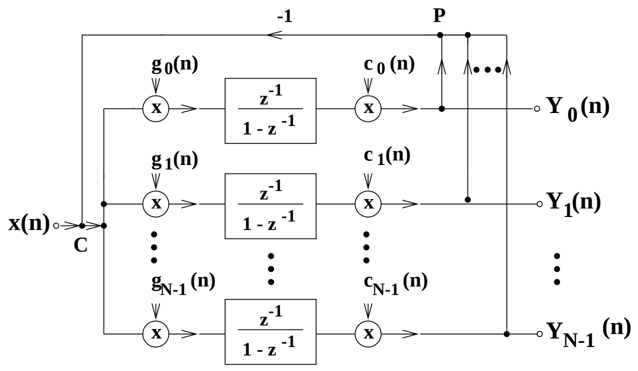


Figure 4.8: Periodic waveform signal analysis: Basis-reciprocal basis synthesizer

expansion coefficients and the signal components are calculated. This latter one can be found at the channel outputs while the others at the outputs of the discrete integrators.

The structures of Figs. 4.5, 4.7, and 4.8 can be derived in different ways, but it is common that the signal is decomposed into components and these components are summed up at point P, i.e. it is reconstructed and subtracted from the input at point C. For all the above structures the transfer value from the input to point P equals

$$H_P(z) = z^{-N} \ , \tag{4.49}$$

i.e. the delay of a complete block. If the input is periodic with this period, then after N samples the difference at point C will be zero and the resonators will continue to generate the sinusoidal components without any "disturbance". If there is a difference at point C, then the error signal will modify the resonator states to reduce this difference. The behavior of the complete control loop at and around the resonator frequencies is very promising due to the very high loop gain.

The V-S transformer structures perform the so called inverse transformations. They start from the transform-domain coefficients available as a parallel vector and produce a serial sequence. A periodic repetition of this operation results in a periodic waveform. As the first step let's consider the generation of a single sinusoidal waveform. This can always be performed using a ROM or RAM filled with the samples of the signal to be generated. This memory is addressed in a systematic way to access periodically all the signal samples which are needed. In this subsection, however, an alternative form is emphasized simply because it will serve as an internal building block of larger signal generating filter-banks. Single sinusoidal waveforms can be generated by complex coefficient, first-order systems as it is given on Fig. 4.9.a. The state variable description of this system has the form of

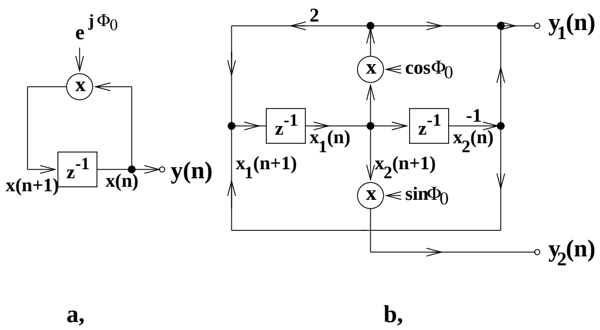


Figure 4.9: Direct form digital oscillator: a, First-order complex, b, second-order real versions

$$\begin{aligned} x(n+1) &= e^{j\Phi_0}\, x(n) \\ y(n) &= x(n) \ . \end{aligned} \tag{4.50}$$

If $x(0) = 1$ then Re $y(n) = \cos(\Phi_0 n)$, Im $y(n) = \sin(\Phi_0 n)$, $n = 0, 1, \ldots$. Fig. 4.9.b shows the equivalent real coefficient, second-order system. Its state variable description is given by

$$\begin{aligned} x_1(n+1) &= 2\cos\Phi_0\ x_1(n) - x_2(n) \\ x_2(n+1) &= x_1(n) \\ y_1(n) &= \cos\Phi_0\ x_1(n) - x_2(n) \\ y_2(n) &= \sin\Phi_0\ x_1(n)\ . \end{aligned} \tag{4.51}$$

If $x_1(0) = 0$, and $x_2(0) = -1$ then

$$y_1(n) = \cos(\Phi_0\ n), \qquad y_2(n) = \sin(\Phi_0\ n)\ . \tag{4.52}$$

The structures of Fig. 4.9 can be considered as digital oscillators but also as two dimensional inverse transformers, as well. The initial conditions correspond to the transform-domain representation which specifies a sinusoidal waveform.

For more general periodic waveforms larger scale systems are needed. Fig. 4.10 shows two possible alternatives. The first one on Fig. 4.10.a is

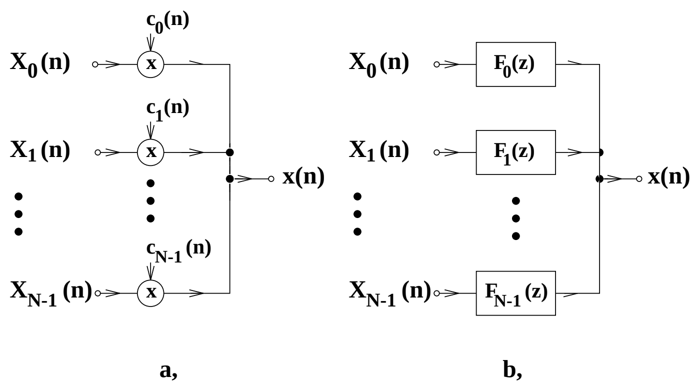


Figure 4.10: Periodic waveform signal synthesis: Signal generation a, by basis functions, b, by base-generating filter-banks

based on the basis system and forms the linear combination of the bases weighted by the transformed-domain coefficients. This operation can be described by the following notations and equation

$$\underline{c}(n) \quad = \quad [c_0(n),\ c_1(n),\ \ldots\ ,\ c_{N-1}(n)]^T,$$

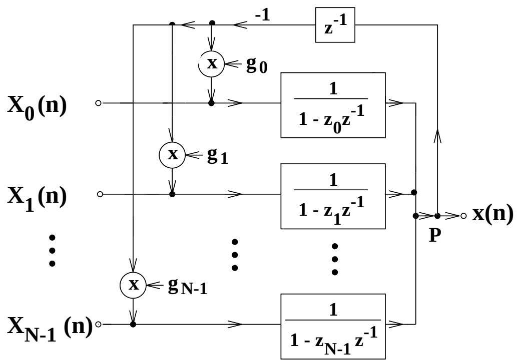


Figure 4.11: Periodic waveform signal synthesis: Resonator-based synthesizer

$$
\begin{aligned}
\underline{X}(n) &= [X_0(n),\ X_1(n),\ \dots,\ X_{N-1}(n)]^T, \qquad\qquad n = 0, 1, \dots \\
x(n) &= \underline{c}^T(n)\underline{X}(n)\ . \qquad\qquad (4.53)
\end{aligned}
$$

After the first N samples the operation can be continued simply by periodically repeating the samples of the bases.

On Fig. 4.10.b the signal generation is performed by a filter-bank consisting of N filters. If the impulse responses of these filters consist of the corresponding bases, i.e. the impulse response of $F_m(z)$ ($m = 0, 1, \dots, N-1$) consists of the subsequent basis vector elements $c_m(n)$, $n = 0, 1, \dots, N-1$, and the $X_m(n)$ values appear periodically in every Nth step at the input of each channel, otherwise the input is zero, then the filters will generate the weighted bases, and finally the sum gives the required waveform. These filters are called basis-generating filters.

For a continuous operation the basis generating filters require the periodic repetition of the transform domain representation of the signal. This can be performed by a system given by

$$
X_m(z) = X_m(0)\,\frac{1}{1 - z^{-N}}\ , \qquad m = 0, 1, \dots, N-1\ . \qquad (4.54)
$$

This transfer function has an interesting relation to the basis generating filters. A relatively condensed filter-bank form can be created due to the fact that the zeros of the basis generating filters coincide with the poles of equation (4.54) except one which corresponds to the actual sinusoidal component generation. The condensed form is the so-called resonator-based synthesizer (See Fig. 4.11). In this structure parallel, complex coefficient, first-order resonators operate within a common feedback loop. If for the g_m coefficients equation (4.47) is applied then for the filter channels we have

$$\begin{aligned} F_m(z) &= \frac{\frac{1}{1-z_m z^{-1}}}{1+\frac{1}{N}\sum_{n=0}^{N-1}\frac{z_n z^{-1}}{1-z_n z^{-1}}} = \frac{1-z^{-N}}{1-z_m z^{-1}} = \\ &= 1 + z_m z^{-1} + \ ... \ + z_m^{N-1} z^{-(N-1)} \end{aligned} \tag{4.55}$$

where the relation $\prod_{n=0}^{N-1}(1-z_n z^{-1}) = 1 - z^{-N}$ has been utilized. It is interesting to note that the structure on Fig. 4.11 is simply the transposed version of that on Fig. 4.5.

In the case of arbitrary basis systems the scheme of Fig. 4.10.a can also be followed. The feedback system producing the basis generating filter-bank for this particular case is given on Fig. 4.12. This structure is again

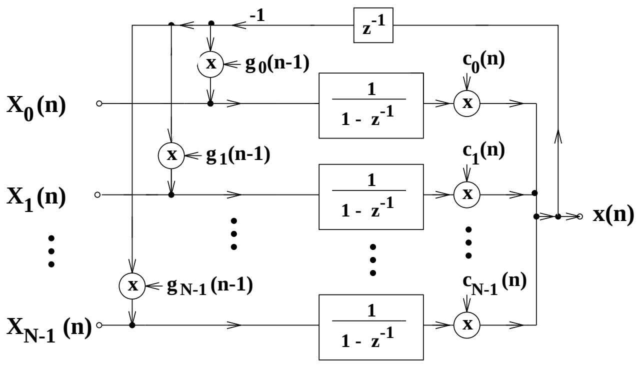


Figure 4.12: Periodic waveform signal synthesis: Basis function synthesizer

transposed version of the corresponding analyzer on Fig. 4.8.

Concerning implementation the time-recursive analyzer-synthesizer pairs introduced above offer several alternatives. Due to the nature of the S-V and V-S transforms a high-degree of parallelism is present within these structures and also the majority of the operations require only local interconnections. However, some limitations and difficulties come from the fact that linear combinations require additions with multiple inputs. Further limitations may occur in the case of the feedback systems since pipelining can be applied only to a limited extent.

4.5.4 Fast algorithms for time-recursive transforms

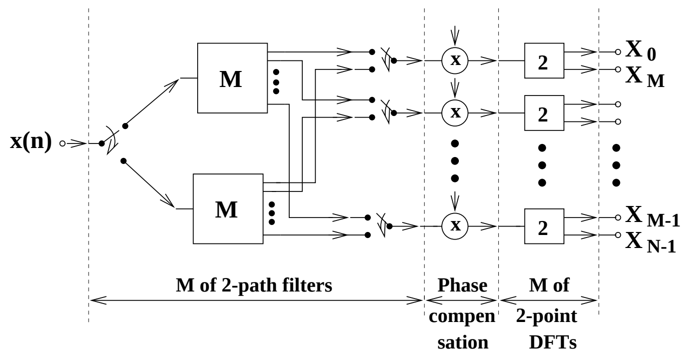
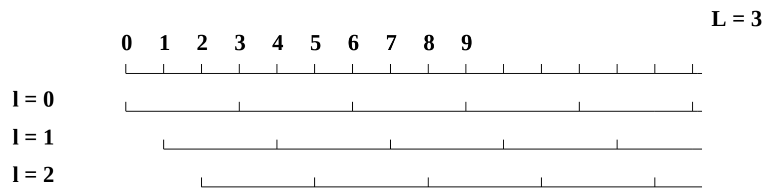


Figure 4.13: Polyphase decomposition for $M = N/2$ and $L = 2$

In this Subsection fast algorithms will be presented for time-recursive evaluation of certain orthogonal transformations. These algorithms combine the block-oriented fast transforms with a polyphase decomposition and result in a composite filter-bank suitable to operate at different output rates. Both the sliding window and the maximally decimated modes of operations can be provided. As a first step the polyphase decomposition will be given for the DFT. If we consider the definition of the DFT

$$X_m = \frac{1}{N} \sum_{n=0}^{N-1} x(n) e^{-j\frac{2\pi}{N}mn} \tag{4.56}$$

where $m = 0, 1, \ldots, N-1$, let's suppose that N can be written as $N = LM$ where L and M are integer numbers and let's decompose the sequence $x(n)$, $n = 0, 1, \ldots, N-1$, into L blocks each having M elements



With this decomposition (4.56) has the form of

$$
\begin{aligned}
X_m &= \frac{1}{N}\sum_{k=0}^{M-1} x(kL)e^{-j\frac{2\pi}{N}mkL} + \frac{1}{N}\sum_{k=0}^{M-1} x(kL+1)e^{-j\frac{2\pi}{N}m(kL+1)} + \ldots \\
&+ \frac{1}{N}\sum_{k=0}^{M-1} x(kL+L-1)e^{-j\frac{2\pi}{N}m(kL+L-1)} = \\
&= \frac{1}{L}\sum_{l=0}^{L-1} \mathrm{DFT}_M(m,l)e^{-j\frac{2\pi}{N}ml} \qquad (4.57)
\end{aligned}
$$

where $\mathrm{DFT}(m,l)$ denotes the lth DFT value from the L M-point DFTs

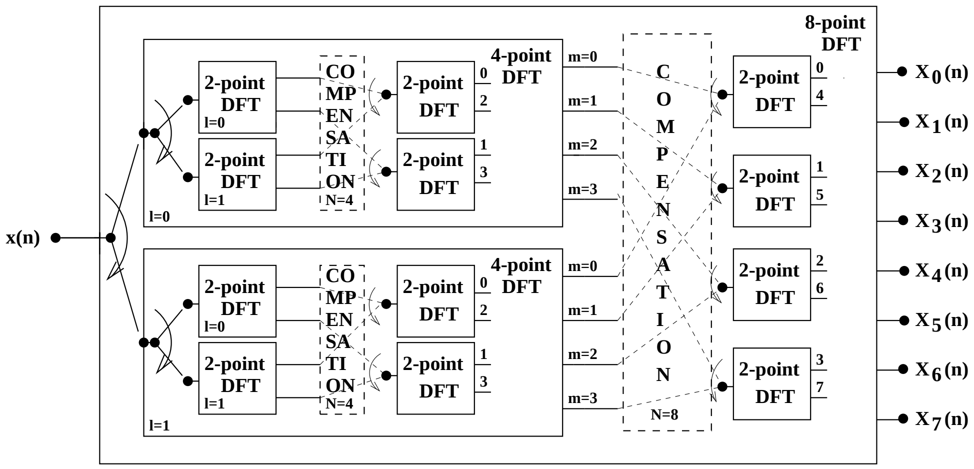


Figure 4.14: Polyphase DFT analyzer for $N = 8$

at the mth "frequency". A further investigation of (4.57) shows that for $m = 0, M, 2M, \ldots$ an L-point DFT is to be evaluated while for other values of m the $\mathrm{DFT}(m,l)$ values should be modified by a correction factor as follows

$$\begin{aligned}
X_m &= \mathrm{DFT}_L\left(\mathrm{DFT}_M(m,l)\right)\ , \\
&\qquad\qquad\qquad\qquad \text{if } m = 0, M, ..., (L-1)M \\
&= \mathrm{DFT}_L\left(\mathrm{DFT}_M(m,l)\ e^{-j\frac{2\pi}{N}l}\right)\ , \\
&\qquad\qquad\qquad\qquad \text{if } m = 1, M+1, ..., (L-1)M+1 \\
&\ \vdots \qquad\qquad\qquad\qquad\qquad \vdots \\
&= \mathrm{DFT}_L\left(\mathrm{DFT}_M(m,l)\ e^{-j\frac{2\pi}{N}(M-1)l}\right)\ , \\
&\qquad\qquad\qquad\qquad \text{if } m = M-1, 2M-1, ..., LM-1\ ,
\end{aligned} \tag{4.58}$$

i.e. to calculate (4.56) the first step is to evaluate L M-point DFTs

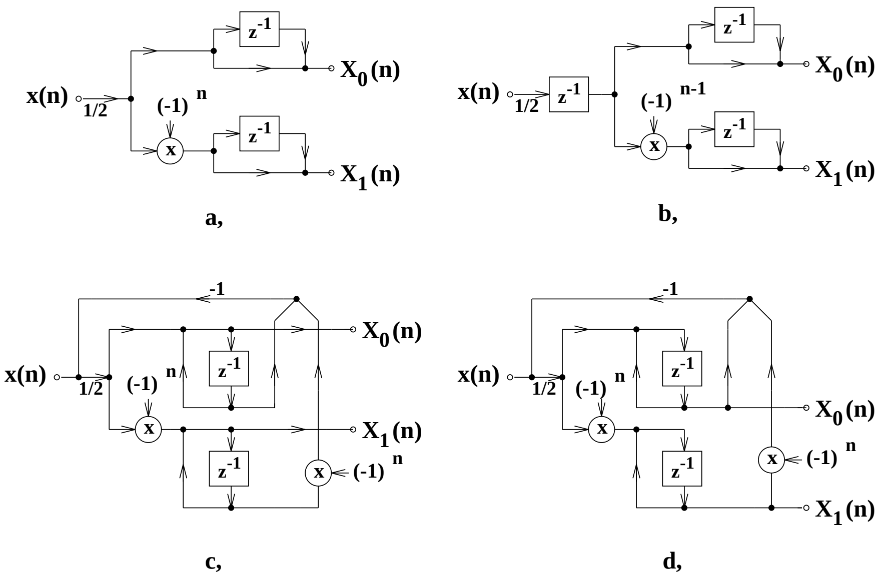


Figure 4.15: 2-point DFT structure versions: a, The simplest tree structure, b, tree structure in pipelineable form, c, resonator based version, d, resonator based in pipelineable form

which is followed by the corrections and finally by M L-point DFTs. If M and/or L are not relative primes further decompositions are possible

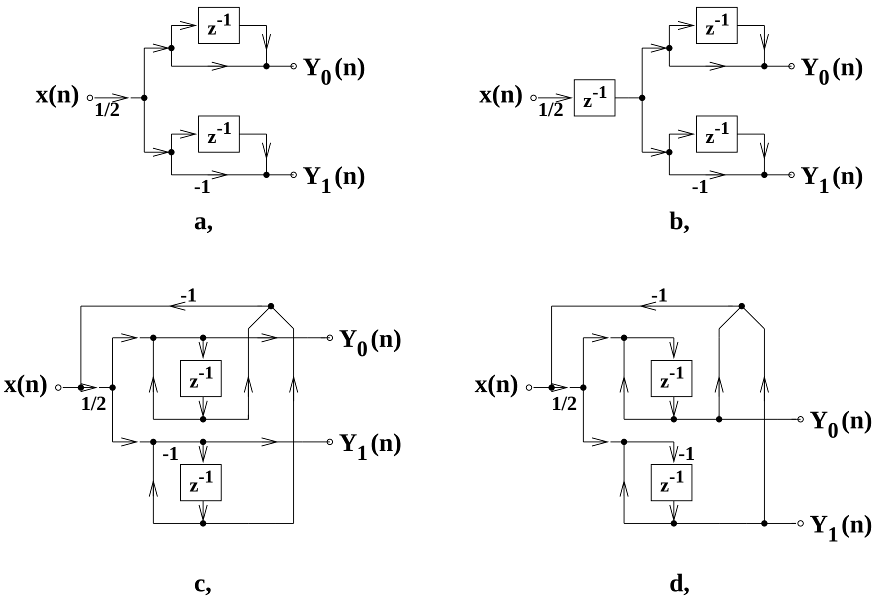


Figure 4.16: 2-point DFT filter-bank versions: a, The simplest tree structure, b, tree structure in pipelineable form, c, resonator based version, d, resonator based in pipelineable form

within the smaller size DFTs similarly to the FFT algorithms [OS75]. The whole decomposition procedure is illustrated by Fig. 4.13. Since the DFT blocks can be implemented using either the filter-bank on Fig. 4.4.b or on Fig. 4.5, the complete system can be operated continuously and also the notation X_m, $m = 0, 1, \ldots, N-1$ can be replaced $X_m(n)$ to emphasize the possible time dependence of the transformed domain representation. The input sequence to be transformed is distributed cyclically among L independent M-point DFTs, and accordingly input samples enter into a given DFT in every Lth step allowing the reduction of the sampling rate within each parallel DFT block. Also the outputs of the first DFTs are scanned cyclically and used to feed the next parallel DFT stage.

The embedding of the decomposition described by equation (4.57) and (4.58) is illustrated by Fig. 4.14 for $N = 8$. From this Figure it is easy to realize that the complete structure is highly parallel in which the processing steps are performed in a polyphase manner [CR83] where the internal blocks

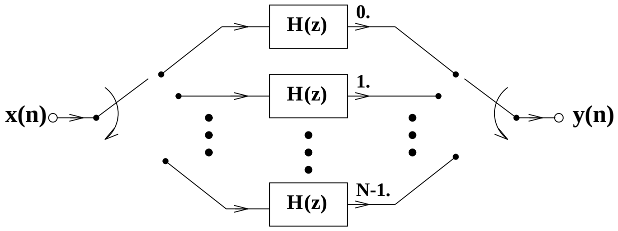


Figure 4.17: Block diagram of the N-path filter

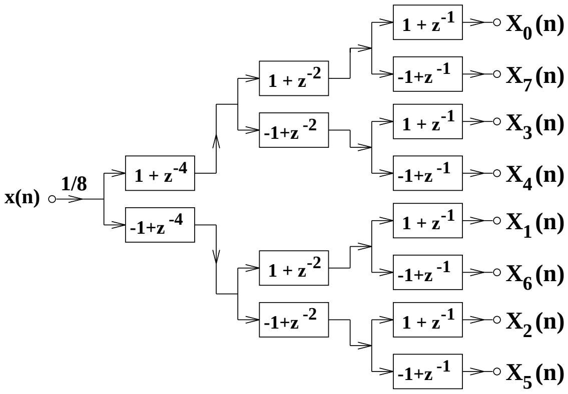


Figure 4.18: Block diagram of the Walsh-Hadamard analyzer for $N = 8$: Tree structure version

operate at a reduced rate. It is interesting to note also how is performed the serial to parallel conversion. To implement the 2-point DFTs many alternatives can be considered. Figs. 4.15 and 4.16 offer the most typical ones. For transformers producing the coefficients of the series expansion the versions of Fig. 4.15 can be used while for sliding transforms where the signal components should appear at the outputs the versions of Fig. 4.16 are suitable. If the timing of the application requires real-time behavior then the pipelined 2-point blocks are to be used. The delay caused by pipelining for a radix-2 case of order N will be $2(N-1)$ steps. Note that a serial to parallel conversion always produces a delay of N steps.

The structures on Figs. 4.13 and 4.14 can be investigated using the methods elaborated for polyphase filtering [CR83]. The basic scheme of

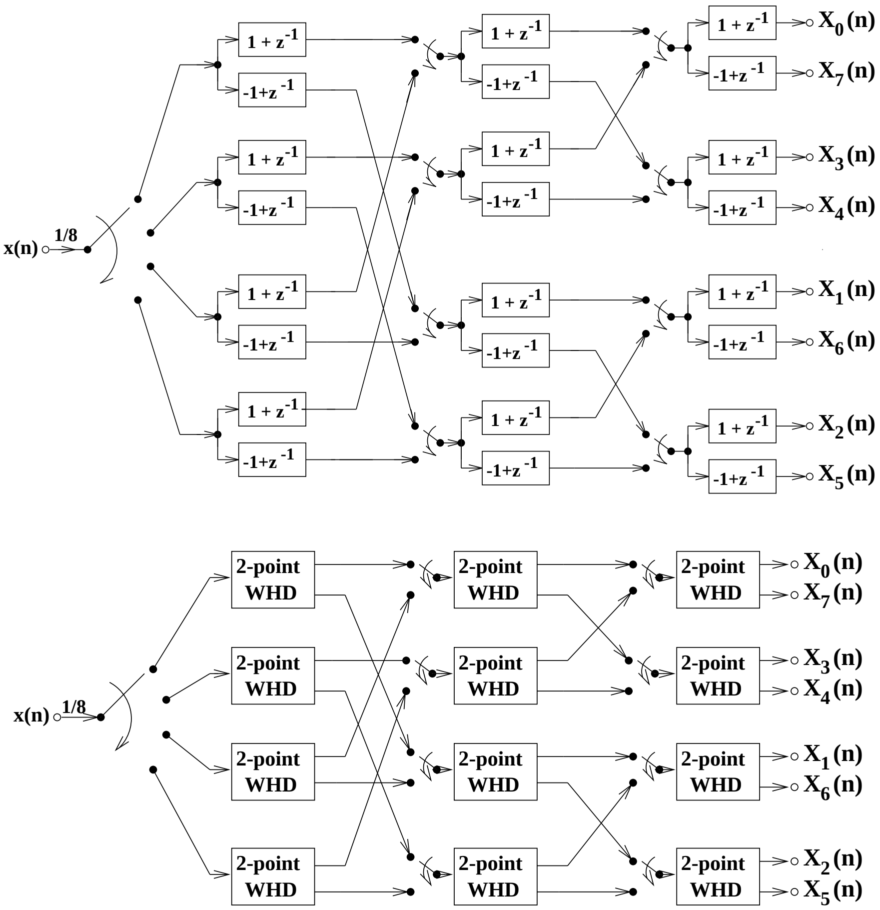


Figure 4.19: Block diagram of the Walsh-Hadamard analyzer for $N = 8$: Polyphase form built of uniform 2-point blocks

polyphase or N-path filters is given on Fig. 4.17. The overall transfer function of such a system is given by $H(z^{-N})$, because every path receives an input sample and produces an output sample only in every Nth step, for the overall transfer function this corresponds to an N-step delay. If the system on Fig. 4.13 is implemented without pipelining then the transfer function of the mth 2-path, M-point DFT filter has the form of

$$H_m(z^2) = \frac{1}{M}\ \left(1 - z^{-2M}\right)\ \frac{1}{1 - z_m\ z^{-2}} \tag{4.59}$$

where $z_m = e^{j(2\pi/M)m}$, $m = 0, 1,\ \ldots, M-1$. Equation (4.59) can be derived from the transfer function of the mth 1-path, M-point DFT filter

$$H_m(z) = \frac{1}{M}\ \left(1 - z^{-M}\right)\ \frac{1}{1 - z_m\ z^{-1}}\ . \tag{4.60}$$

It is important to note that equation (4.59) differs from the N-point DFT filter only in one term, i.e. instead of having

$$\frac{1}{2}\ \frac{1}{1 - z_m^{\frac{1}{2}}\ z^{-1}} \tag{4.61}$$

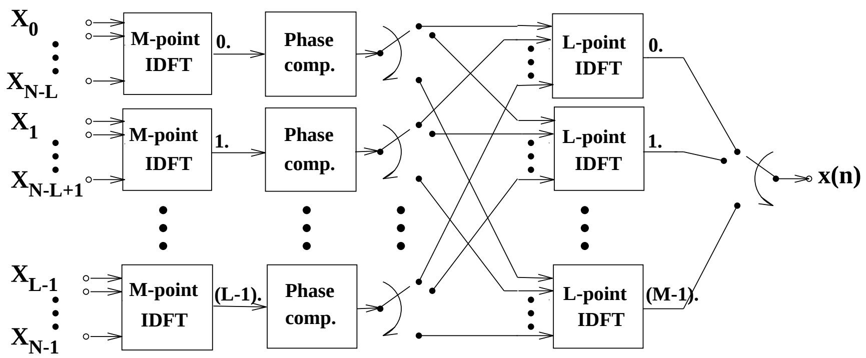


Figure 4.20: Parallel decomposition of the inverse DFT structure

with $z_m^{(1/2)} = e^{j(2\pi/N)m}$, $m = 0, 1,\ \ldots, M-1$, we have

$$\frac{1}{1 - z_m\ z^{-2}} = \frac{1}{\left(1 + z_m^{\frac{1}{2}}\ z^{-1}\right)\ \left(1 - z_m^{\frac{1}{2}}\ z^{-1}\right)}\ . \tag{4.62}$$

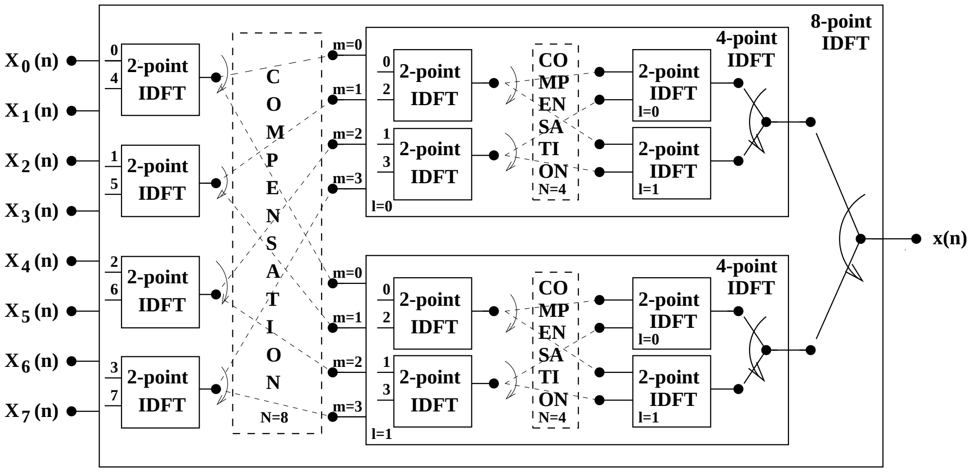


Figure 4.21: Polyphase DFT synthesizer for $N = 8$

To get the N-point DFT filter equation (4.59) should be multiplied by

$$\frac{1}{2}\left(1 + z_m^{\frac{1}{2}}\ z^{-1}\right) \tag{4.63}$$

The implementation of equation (4.63) can be solved by simple transposition circuits together with the first channel of a 2-point DFT transformer. For $m = M, M+1, \ldots, 2M-1 (= N-1)$ using

$$e^{j\frac{2\pi}{N}m} = e^{j\frac{2\pi}{N}(m-M+M)} = -e^{j\frac{2\pi}{N}(m-M)} \tag{4.64}$$

(4.61) is replaced by

$$\frac{1}{2}\ \frac{1}{1 + z_m^{\frac{1}{2}}\ z^{-1}} \tag{4.65}$$

and the counterpart of (4.63) will be

$$\frac{1}{2}\left(1 - z_m^{\frac{1}{2}}\ z^{-1}\right)\quad . \tag{4.66}$$

Equations (4.63) and (4.66) will lead to such a 2-point DFT block which calculates the mth and the $(m+M)$th DFT components.

The fast and time-recursive evaluation technique of the DFT described above can be generalized to all basis systems which can be decomposed in

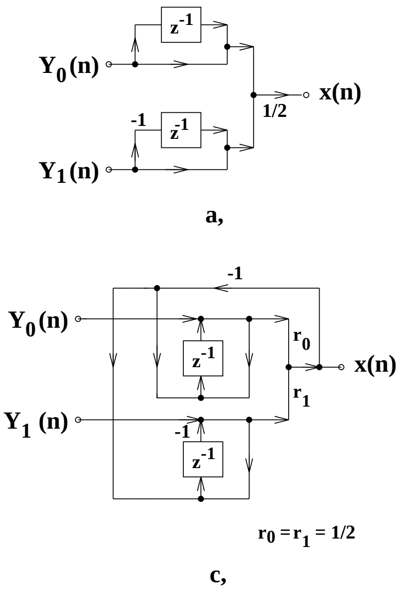


Figure 4.22: 2-point DFT synthesizer filter-bank versions: a, The simplest tree structure, b, tree structure in pipelineable form, c, resonator based version, d, resonator based in pipelineable form

a similar way. The theoretical background of this possibility can be summarized as follows. Let's suppose to have two M dimensional orthogonal matrices $\underline{\underline{T}}_1$ and $\underline{\underline{T}}_2$. Based on this matrices we can compose an orthogonal $2M$ dimensional $\underline{\underline{T}}$ matrix

$$\underline{\underline{T}} = \begin{bmatrix} \begin{bmatrix} a_{11}\, \underline{\underline{I}}_M \end{bmatrix} & \begin{bmatrix} a_{12}\, \underline{\underline{I}}_M \end{bmatrix} \\ \begin{bmatrix} a_{21}\, \underline{\underline{I}}_M \end{bmatrix} & \begin{bmatrix} a_{22}\, \underline{\underline{I}}_M \end{bmatrix} \end{bmatrix} \begin{bmatrix} \underline{\underline{T}}_1 & \underline{\underline{0}} \\ \underline{\underline{0}} & \underline{\underline{T}}_2 \end{bmatrix}, \tag{4.67}$$

where $\underline{\underline{I}}_M$ is the M dimensional unit matrix, and

$$\underline{\underline{A}} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \tag{4.68}$$

is orthogonal. This composition can be extended toward larger scale $\underline{\underline{A}}$ matrices, as well.

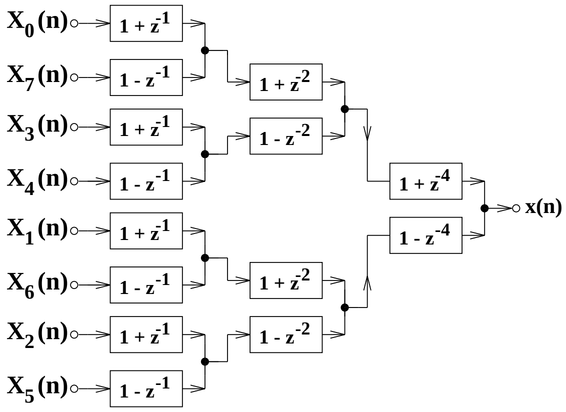


Figure 4.23: Block diagram of the Walsh-Hadamard synthesizer for $N = 8$: Tree structure version

The N-path interpretation of such systems results in the required analysis structures. Let's suppose that the input sequence is decomposed into $L = 2$ groups each having a separate processing unit. Every second sample enters into the first while the others into the second unit. This decomposition can have the following formulation

$$\underline{\underline{T}}\,\underline{x} = [\underline{t}_0\ \ \underline{0}\ \ \underline{t}_2\ \ \underline{0}\ \ \dots\ \ \underline{t}_{N-2}\ \ \underline{0}]\,\underline{x} + [\underline{0}\ \ \underline{t}_1\ \ \underline{0}\ \ \underline{t}_3\ \ \dots\ \ \underline{0}\ \ \underline{t}_{N-1}]\,\underline{x} \tag{4.69}$$

where $\underline{t}_m$ denotes the mth column of $\underline{\underline{T}}$, $m = 0, 1, \dots, N-1$, and $\underline{x}$ stands for the vector of the last N input samples

$$\underline{x}^T = [x_{n-N+1},\ x_{n-N+2},\ \dots,\ x_n]\ . \tag{4.70}$$

By introducing $\underline{\underline{T}}_1 = [\underline{t}_0, \underline{t}_2,\ \dots\ , \underline{t}_{N-2}]$, $\underline{\underline{T}}_2 = [\underline{t}_1, \underline{t}_3,\ \dots\ , \underline{t}_{N-1}]$, and $\underline{x}_1^T = [x_{n-N+1},\ x_{n-N+3},\ \dots\ ,\ x_{n-1}]$, $\underline{x}_2^T = [x_{n-N+2},\ x_{n-N+4},\ \dots\ ,\ x_n]$ equation (4.69) can be expressed as

$$\underline{\underline{T}}\,\underline{x} = \underline{\underline{T}}_1\,\underline{x}_1 + \underline{\underline{T}}_2\,\underline{x}_2 \tag{4.71}$$

where $\underline{\underline{T}}_1$ and $\underline{\underline{T}}_2$ are in general of dimension $N * M$ $(N = 2M)$, and $\underline{x}_1$, $\underline{x}_2$ of dimension M. This decomposition offers some parallelization, but reduction in the computational complexity is available only if some symmetry properties are present. In such cases the rows of the $\underline{\underline{T}}_1$ and $\underline{\underline{T}}_2$ matrices are to be considered only up to the index $M - 1$ and the

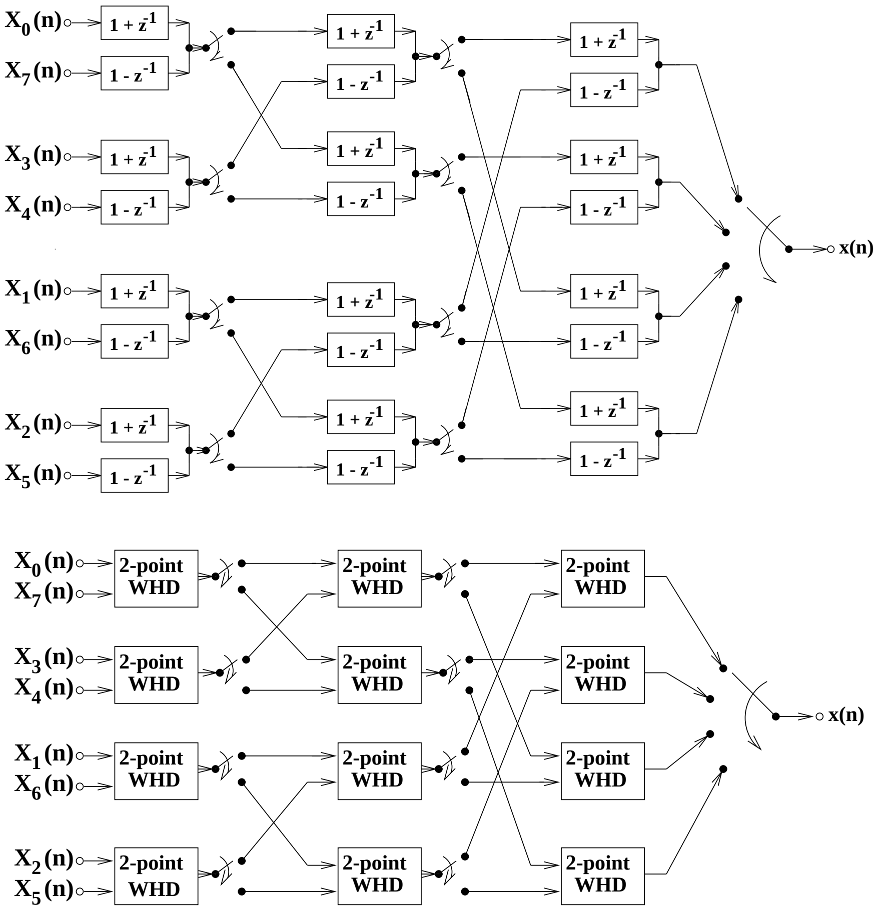


Figure 4.24: Block diagram of the Walsh-Hadamard synthesizer for $N = 8$: Polyphase form built of uniform 2-point blocks

multiplication with the rows of indices $M+m$, $m = 0, 1, \ldots, M-1$ can be replaced with multiplications of rows of indices m ($m = 0, 1, \ldots, M-1$), and finally the partial products are to be combined properly. This happens in the second phase of the development described by equation (4.57) where in our case the result of the transformation can be expressed as the difference of the two terms of equation (4.71) for indices $m = M, M+1, \ldots, N-1$

$$\underline{\underline{T}}\,\underline{x} = \underline{\underline{T}}_1\,\underline{x}_1 - \underline{\underline{T}}_2\,\underline{x}_2 \quad . \tag{4.72}$$

Equations (4.71) and (4.72) together perform a multiplication by a $2*2$ orthogonal matrix for the indices $m = 0, 1, \ldots, M-1$. In this example this $2*2$ orthogonal matrix is the 2-point DFT matrix. A simple generalization of this decomposition results in the form described by (4.67) and (4.68).

For the case of the Walsh-Hadamard transformation a tree-structure solution (see Fig. 4.18) [Vet83] has been suggested. The block diagram of the polyphase version, which can be derived using equations (4.69)-(4.72), is given for $N = 8$ in Fig. 4.19.

The derivation of the (V-S) transform versions is straightforward. The idea of polyphase decomposition can be applied also for the inverse transforms. The resulting structures are fast basis-generating filter-banks, therefore the typical application is to repeat the input vector in every Nth step and thus the generation of a periodic waveform. Fig. 4.20 gives the idea of the decomposition for the DFT, and for the radix-2 case Fig. 4.21 provides a more detailed block diagram. Some possible 2-point synthesizer filter-banks which can serve as building blocks of radix-2 filter-banks are given on Fig. 4.22.

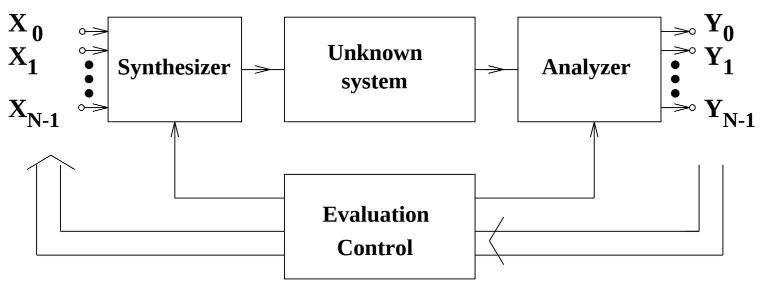


Figure 4.25: Block diagram of the multi-sine measuring setup

An efficient signal generator can be derived from the fast Walsh-Hadamard (V-S) transformer. The tree structure version is shown on Fig. 4.23

while the polyphase form is presented on Fig. 4.24.

The time-recursive transformer pairs have very many practical applications. Probably the most important ones are related to transform domain signal processing [Shy92]. Recently there is an increasing interest to apply transformer-pairs also in multi-sine measuring setups, where the synchronized signal generation and analysis offer exceptional performance (see Fig. 4.25) [Var96a],[Var96b].

The application of the above detailed time-recursive transformer pairs for transform domain sibnal processing may offer new possibilities in the parallelization of the adaptation procedure [VT97]. In the next section it will be shown that these sliding transformers may contribute to a better distribution of the computational load along time and therefore enable higher sampling rates.

4.5.5 Fast sliding transforms in transform-domain adaptive filtering

In recent years transform-domain adaptive filtering methods became very popular especially for those applications where filters with very long impulse responses are to be considered [Shy92]. The basic idea is to apply the fast Fourier Transformation (FFT) for signal segments and to perform adaptation in the frequency domain controlled by the FFT of an appropriate error sequence. There are several algorithms based on this approach [Shy92] and further improvements can be achieved ([CL96]). The formulation of the available methods follows two different concepts. The first one considers transformations as a "single" operation to be performed on data sequences (block-oriented approach), while the other emphasizes the role of multirate analyzer and synthesizer filter-banks.

In this subsection, using the results concerning fast sliding transformation algorithms ([Var95],[Var96a]), a link is developed which helps to identify the common elements of the two approaches. The fast sliding transformers perform exact transformations, however, operate as polyphase filter-banks. These features together may offer further advantages in real-time applications, especially when standard DSP processors are considered for implementation.

There are two major configurations for transform-domain adaptive filtering (See Fig. 4.26). First let us consider standard (not sliding) transformers. Here transformers perform a serial to parallel conversion while inverses convert to the opposite direction. Adaptation is controlled by the transformed input and error signals once for each input block, i.e. decimation is an inherent operation within these algorithms. Investigations related to the block-oriented approach, however, rarely consider real-time aspects

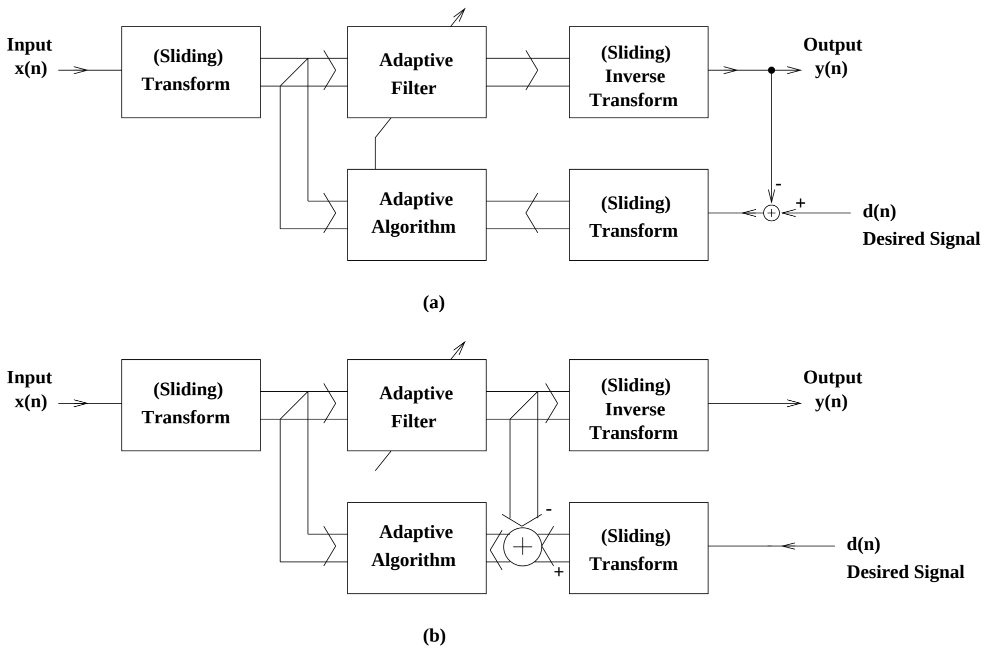


Figure 4.26: Frequency-domain adaptive filter configurations

of signal processing. Typically it is supposed that sampling frequency is relatively low compared to the computational power of the signal processors, and therefore, if a continuous flow of signal blocks must be processed, the block-period is enough to compute the transformations and the filter updating equations. Moreover, in the case of the widely used Overlap-Save and Overlap-Add methods (see e.g. [Shy92]) filter updating requires further transformations, therefore further computational power is needed.

Filter-bank and consequently fast sliding transformation techniques do not offer extra savings in computations, however, they may provide a much better distribution of the computational load with time. This means that potentially they give better behavior when real-time requirements are to be met.

The analyzer bank (see Fig. 4.14) can be operated as well as a sliding-window DFT or as a block-oriented transformer. This latter means that the parallel outputs are maximally decimated as it is typical with serial to parallel converters. However, if overlapped data segments are to be transformed, the structure is well suited to support decimation by any integer number. The widely used Overlap-Save Method (see later) concatenates two blocks of size N to perform linear convolution and calculates $2N$-point FFTs. A $2N$-point sliding transformer can easily produce output in every Nth step.

In certain applications it may be advantageous to produce signal components instead of the Fourier coefficients as it is dictated by the definition of the DFT. In this case the filter-bank is a set of band-path filters with center frequencies corresponding to the Nth roots of unity values. Such DFT filter-banks can be easily derived using the ideas valid for the DFT transformers. With this DFT filter-bank approach transform-domain signal processing can have the following interpretation: The input signal to be processed first is decomposed by a filter-bank into components and the actual processing is performed on these components. The modified components enter into a N-input single output filter-bank (the so-called synthesizer bank, Figs. 4.20 and 4.21) which produces the output sequence.

The concept of transform-domain adaptive filtering offers real advantages if adaptive FIR filters with very long impulse responses are to be handled. The first important aspect is the possible parallelization described above achievable using fast transformation algorithms. The second is the applicability of block adaptive filtering: the parallel channels enable decimation and therefore a coefficient update only once in every Nth step. In the meantime, however, a much better gradient estimate can be derived.

Fig. 4.26 shows two possible forms of frequency-domain adaptive filtering. In the first version adaptation is controlled by the time-domain difference of the filter output and the desired signal. The adaptive filter performs N multiplications using the N-dimensional weighting vector generated by the adaptive algorithm. The adaptation mechanism can be of any kind controlled by an error signal, however, in the majority of the applications the least-mean-square (LMS) algorithm is preferred (see e.g. [Shy92]) for its relative simplicity. Adaptation can be performed in every step, however, drastic reduction of the computations can be achieved only if the transformer outputs are maximally decimated. The techniques developed for this particular case are the so-called block adaptive filtering methods. [Shy92] gives a very detailed analysis of the most important approaches. It is emphasized that the classical problem of linear versus circular convolution appears also in this context. This problem must be handled because in the majority of the applications a continuous data flow is to be processed and therefore the dependence of the neighboring blocks can not be neglected without consequences. The correct solution is either the Overlap-Save or the Overlap-Add Method. Both require calculations where two subsequent data blocks are to be concatenated and double-sized transformations are to be performed.

If we consider the system of Fig. 4.26.a from timing point of view it is important to observe that at least two transformations must be calculated within the adaptation loop. If real-time requirements are also be fulfilled the time needed for these calculations may be a limiting factor. For the deeper analysis let us investigate e.g. the famous Overlap-Save Method. (Fig. 4.27) The Overlap-Save sectioning method performs a linear convolution between a finite-length sequence and an infinite-length sequence by appropriately partioning the data. The finite-length sequence (in our case the adaptive weights) has N elements; after appending N zeros, a $2N$-point FFT is computed. For the infinite-length input sequence, the most recent N data samples are concatenated with the previous block of N samples; a $2N$-point DFT of this extended data vector is then computed. The product of the transformed sequences is processed by a $2N$-point IFFT, yielding a block of output samples. The first N points of this output frame are discarded, while the last N points are the desired output samples of a linear convolution. So, moreover, if we investigate more thoroughly e.g. the Overlap-Save Method it turns out that the calculation of the proper gradient requires the calculation of two further transformations, i.e. there are altogether four transformations within the loop. This may cause considerable delay and performance degradation especially critical in tracking nonstationary signals.

The adaptation of the transform-domain adaptive filtering scheme on Fig. 4.26.b is controlled by an error vector calculated in the transform-domain. Due to this solution the transformer blocks are out of the adaptation loop, therefore the delay within the loop can be kept at a lower level. Here the adaptation is completely parallel and is to be performed separately for every "channel". The operation executed in this scheme corresponds to the circular convolution which may cause performance degradation due to severe aliasing effects.

In the literature of the analysis filter-banks the so-called subband adaptive filters (see e.g. [Shy92]) are suggested for such and similar purposes which at the prize of smaller $L < N$ decimation rates provide better aliasing suppression. The fast sliding transformers are in fact efficiently implemented special filter-banks. If they are maximally decimated they suffer from the side-effects of the circular convolution. In order to reduce aliasing the application of $L = N/2$ can be advised. If the the number of the adaptive filter channels is lower than the size of the sliding transformer the standard windowing techniques (see e.g. [RG75]) can be used for channel-filter design.

The timing conditions of the frequency-domain adaptive filters using the Overlap-Save Method also offer some ways for the parallelism. This

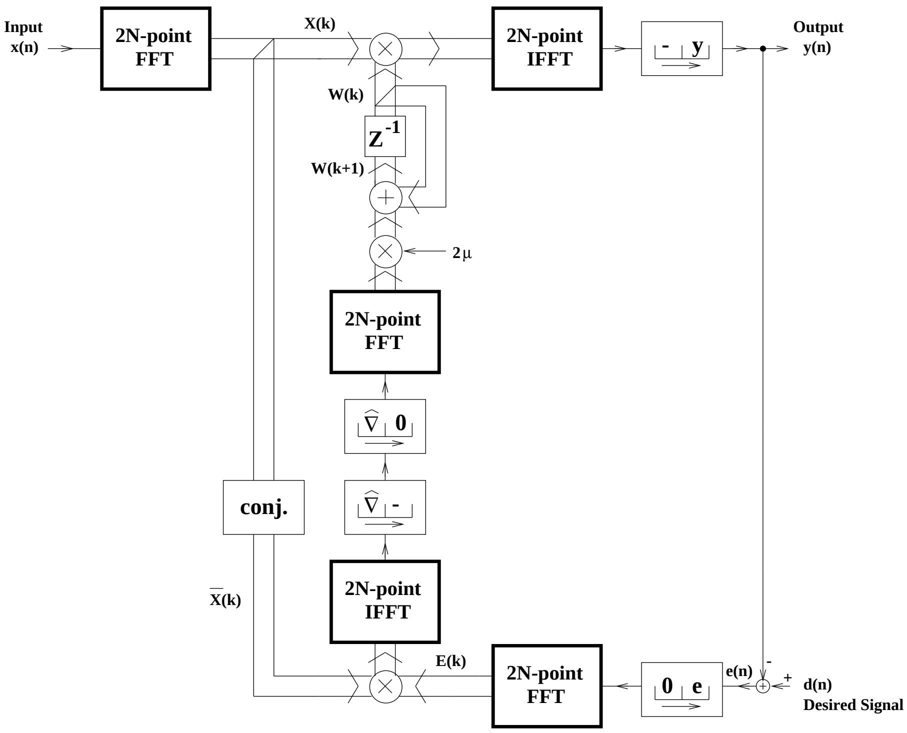


Figure 4.27: Block diagram of the Overlapped-Save Method

technique is carefully described in [Shy92] therefore here only the most critical elements are emphasized. Other techniques can be analyzed rather similarly.

Fig. 4.28 shows the timing diagram of the standard block-oriented solution. Here the acquisition of one complete data block (N samples) is followed by the processing of this data vector. A continuous sequence of data blocks can be processed if the execution time of one block adaptation is less or equal to the corresponding acqusition time ($t_e \leq t_a$). One block adaptation consists of several steps, among them the transformation of the input and the error sequences, respectively (see Fig. 4.26.a). The generation of the output sequence requires an additional transformation. Due to the requirements of the linear convolution all these transformations work on double blocks, i.e. on $2N$ data points. The calculation of the gradient and

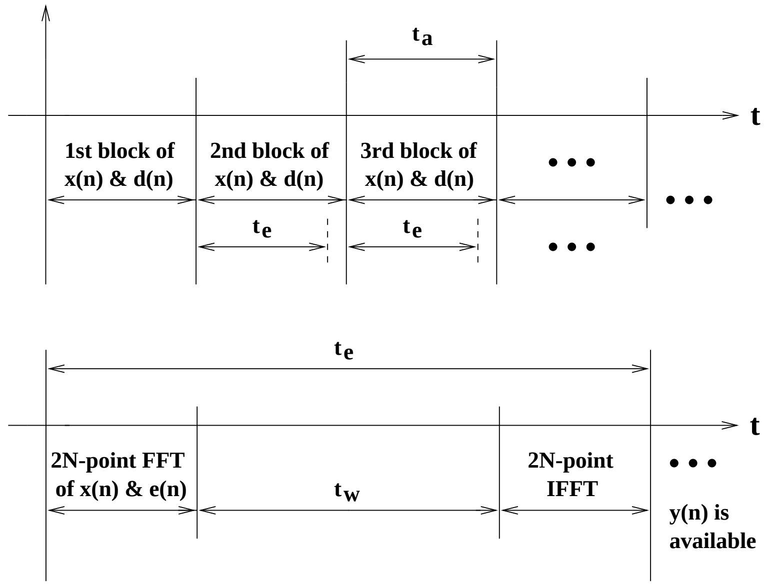


Figure 4.28: Timing diagram of the standard overlap-save frequency-domain adaptive algorithms (t_a denotes the acquisition time of one data block of N samples, t_e stands for the execution time of one block adaptation, and t_w for the calculation time of the gradient and the W update). The end-to-end delay is of N samples.

coefficient update requires further two such transformations. A detailed analysis of the steps using the $2N$-point transformations shows that none of them is "complete", i.e. some savings in the computations are possible.

With the introduction of the sliding transformers the acquisition of the input blocks and the processing can be overlapped, since the sliding transformers can start working already before having the complete block. If we permit an end-to-end delay of $2N$ samples then the processing can be extended for three acqusition intervals (see Figs. 4.29 and 4.30). During the first interval the data acqusition is combined with the transformation, the second can be devoted for finishing the transformation, for updating the coefficient vector and to start the inverse transformation which can be continued in the third interval because the data sampling performed parallelly provides the $d(n)$ samples (see Fig. 4.26) sequentially. At the prize of larger end-to-end delay the execution time can be extended for more blocks, as well.

In the case of the Overlap-Save Method the gradient calculation consists of an inverse transformation, some simple manipulations and a transformation. Since with the sliding inverse transformation a parallel to serial conversion is performed and the sliding transformer implements a serial to parallel conversion, further overlapping in the execution is possible as it is indicated on Figs. 4.29 and 4.30.

The above considerations can result savings if the granularity of the hardware and software elements of implementation enable smooth distribution of the computational load. If separate hardware units are available for the sliding transformations then the parallelism of the execution can be considerably improved. It is important to note that the precedence conditions of block processing do not support such parallelizations: the complete data block must be available for a block-oriented operation like a signal transformation.

To illustrate the achievable gain using fast sliding transforms here a comparison is made concerning the timing and computational conditions of the two approaches. In the usual implementations of the conventional method data acqusition and processing are separated in time, i.e. after the arrival of a complete input data block an efficient DSP program calculates the transformed values. The sliding FFT introduced in [Var95] has the same computational complexity as the traditional algorithm, however, in the proposed polyphase filter-bank data acquisition and processing overlap, therefore calculations start well before the complete block becomes available. There are very many ways to characterize the complexity of the FFT, one possible figure can be the number of complex multiplications. This figure for the maximally decimated sliding N-point FFT equals (see [Var95])

$$C1 = \frac{N}{4} \log_2 \left(\frac{N}{8} \right) + 1 \, , \tag{4.73}$$

i.e. this figure is one of the possible characterizations. These operations, however, are performed partly during the data acqusition, and the number of complex multiplications which can not be executed before the arrival of the last sample of the block remains only

$$C2 = \frac{N}{2} - \log_2 N \, . \tag{4.74}$$

As an example for $N = 1024$ $C1 = 1793$ and $C2 = 502$, i.e. the possible time gain due to the overlap can be considerable. Similar figures can be given for other operations, as well.

Remark: In this section everywhere the term "transform-domain" was used instead of the explicit term "frequency-domain". The reason for this is that all the above developments can be extended for other type of transformations, as well. The application of other transformers may reduce either the computational load or in certain cases improve the adaptation performance.

4.5.6 Implementation of recursive discrete transforms

There are very many solutions to perform matrix-vector products and therefore very many alternatives to implement discrete transforms [Kun88] as a block operation. Further considerations are required, however, if for some reason the time-recursive versions promise better overall performance, therefore in this subsection the implementation of recursive discrete transforms is concerned.

Recursive transformations may serve on one hand as special serial-to-parallel converters where with the parallelization the speed of the processing can be increased, while on the other can contribute to feature extraction, i.e. to data reduction which i unavoidable in almost every signal processing system.

The usual time-recursive transformers are maximally decimated/interpolated analysis/synthesis filter-banks, i.e. their implementation requires very similar hardware and software platform as the realization of FIR/IIR filters has. The simplest solution s to have a separate FIR filter for every channel, i.e. for an $N*N$ transformation N filters of order N. In subsection 2.4.3 a simple feedback structure was presented as a possible alternative for almost all kind of transformations, and finally 2.4.4 introduces the structure corresponding to the fast transformation algorithms. The feedback structure offers interesting generalizations as infinite-memory and/or adaptive transformations, while the fast transformer structure reflects the efficiency of the corresponding fast algorithms.

The direct utilization of time-recursive transformers, possibly with the exception of some dedicated products, is going to be typical in larger scale application systems, often called as embedded systems, where a collection of programmable parts surrounded by ASICs and other standard components interacts continuously, i.e. in real-time with an environment through sensors and actuators. The programmable parts include micro-controllers and Digital Signal Processors (DSPs), while among the ASICs the field-programmable gate-arrays (FPGAs) play a very important role.

If more DSPs are available the actual implementation is a special mixture of hardware and software. Recent advances in system design [Bal97] offer nice tools to handle such environments. Due to the parallel nature of the transformer structures a wide range of alternative solutions is available depending on the amount of hardware facilities.

From realization point of view the transformations differ considerably from each other. Concerning computational conplexity the Walsh-Hadamard transformer seems to be one of the most attractive since for this transformation except for a single normalization the multiplications can be reduced to simple additions/substructions. An FPGA based Walsh-Hadamard transformer is described in [Feh91], where the data processing is serial and therefore the amount of logic and arithmetic units does not depend on the actual wordlength of the processing. For other transformations like the Fourier family, and several others, the multiplications can't be avoided. Special bit-serial aritmetics are developed for FPGA based multipliers [Feh92b] and complete building blocks like resonators. In the case of XILINX XC4000 FPGA elements the 2's-complement fixed point LSB first bit-serial arithmetic fully utilizes the architectural benefits of the selected device.

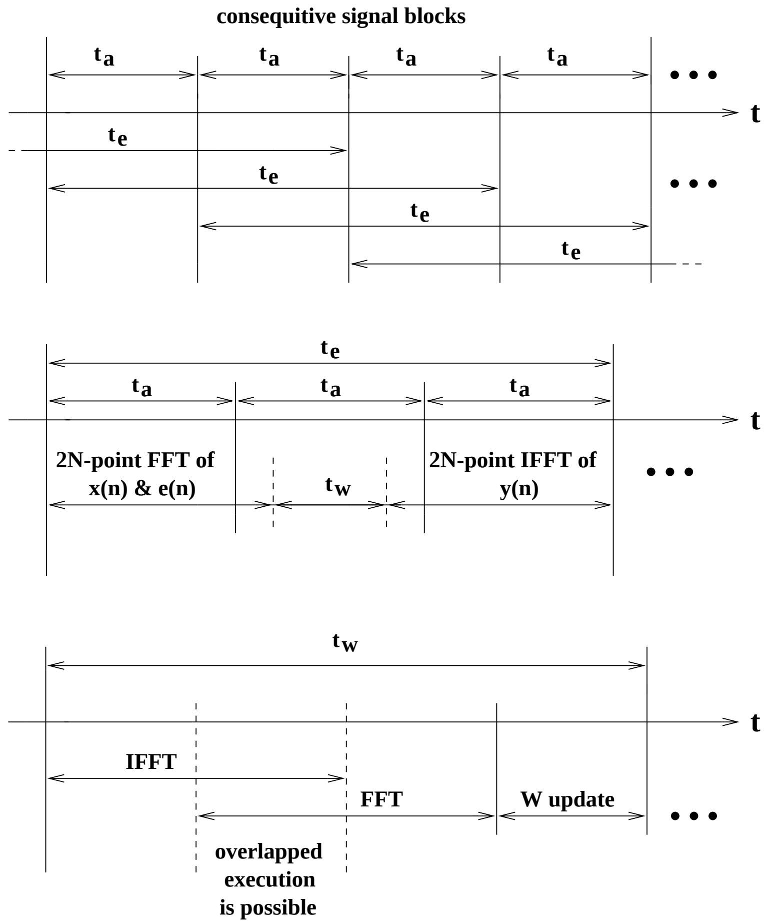


Figure 4.29: Timing diagrams of a possible frequency-domain adaptive algorithm using sliding transformations. (t_a denotes the acquisition time of one data block, t_e stands for the execution time of one block adaptation, and t_w for the calculation time of the gradient and the W update). The end-to-end delay is of $2N$ samples.

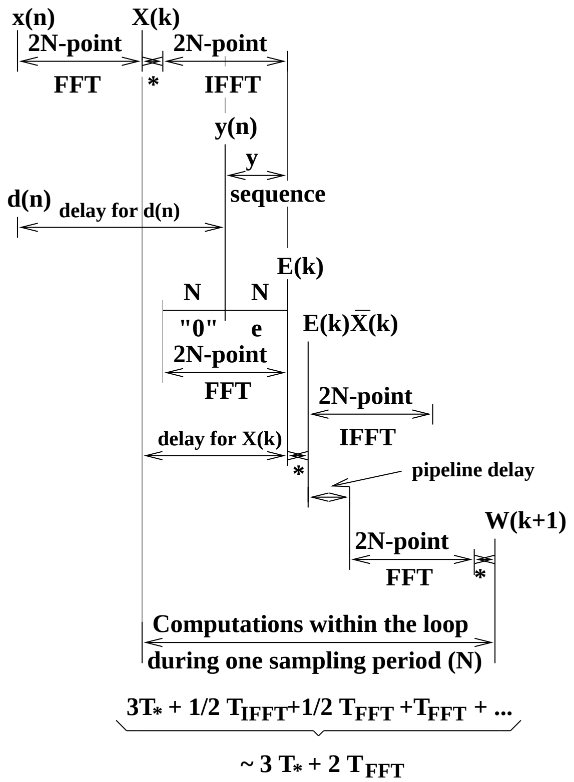


Figure 4.30: Timing diagram of the sliding window version

References and further readings

[AR75] N. Ahmed, K.R. Rao, *Orthogonal Transforms for Digital Signal Processing*, Springer-Verlag, New York, 1975.

[Bal97] F. Balarin et al., *Hardware-Software Co-Design of Embedded Systems,* Kluwer Academic Publishers, Dordrecht, The Netherlands, 1997. pp. 320.

[Bor94] V. Boriakoff, "FFT Computation With Systolic Arrays, A New Architecture", *IEEE Transactions on Circuits and Systems II.: Analog and Digital Signal Processing*, Vol. 41, No. 4, pp. 278-284, Apr. 1994.

[Bra84] R.N. Bracewell, "The Fast Hartley Transform", *Proceedings of the IEEE*, Vol.72, No. 8, pp. 1010-1018, Aug. 1984.

[BZ92] N. Botros and W. Zakhem, "FFT Processor Using Filed Programmable Gate Arrays", *Proceedings of the Fifth Annual IEEE International ASIC Conference and Exhibit*, Sept. 21-25, 1992, Rochester, New York, pp. 115-118.

[CB92] J. Choi and V. Boriakoff, "A New Linear Systolic Array for FFT Computation", *IEEE Transactions on Circuits and Systems II.: Analog and Digital Signal Processing*, Vol. 39, No. 4, pp. 236-239, Apr. 1992.

[CH92] Shing-Chow Chan and Ka-Leung Ho, "Fast Algorithms for Computing the Discrete Cosine Transform", *IEEE Transactions on Circuits and Systems II.: Analog and Digital Signal Processing*, Vol. 39, No. 3, pp. 185-190, March 1992.

[CL96] H.-C. Chiang, J.-C. Liu, "Fast Algorithm for FIR Filtering in the Transform Domain", *IEEE Transactions on Signal Processing*, Vol. 44, No.1, pp.126-129, Jan. 1996.

[CR83] R.E. Crochiere, L.R. Rabiner, *Multirate Digital Signal Processing.* Prentice-Hall, Englewood Cliffs, NJ, 1983.

[CS95] Lap-Pui Chau and Wan-Chi Siu, "Direct Formulation for the Realization of Discrete Cosine Transform Using Recursive Structure", *IEEE Transactions on Circuits and Systems II.: Analog and Digital Signal Processing*, Vol. 42, No. 1, pp. 50-52, Jan. 1995.

[Dra+87] B.L. Drake, F.T. Luk, J.M. Speiser and J.J. Symanski, "SLAPP: A Systolic Linear Algebra Parallel Processor", *Computer*, Vol. 20, No. 7, pp. 45-49, July 1987.

[Feh91] B. Fehér, "Field-Programmable Gate Array Implementation of the Recursive Walsh-Hadamard Transformation", *Proceedings of the Second International Conf. on DSP Applications and Technology*, Oct. 28-31, 1991, Berlin, Germany, pp. 250-254.

[Feh92a] B. Fehér, "Digital Signal Processing Using Programmable Logic Elements", *MicroCAD/SYSTEM'92*, Febr. 25-28, 1992, Miskolc, Hungary.

[Feh92b] B. Fehér, "Resonator Based Digital Filters Using Filed Programmable Gate Array Elements", *Proceedings of the Fifth Annual IEEE International ASIC Conference and Exhibit*, Sept. 21-25, 1992, Rochester, New York, pp. 119-122.

[GB73] B. Gold and T. Bially, "Parallelism in Fast Fourier Transform Hardware", *IEEE Transactions on Audio and Electroacoustics*, Vol. 21, No. 1, pp. 5-16, Febr. 1973.

[GW95] S.F. Gorman, J.M. Wills, "Partial Column FFT Pipelines", *IEEE Transactions on Circuits and Systems II.: Analog and Digital Signal Processing*, Vol. 42, No. 6, pp. 414-423, June 1995.

[Hos80] G.H. Hostetter, "Recursive Discrete Fourier Transformation", *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. ASSP-28, No. 2, pp. 184-190, Apr. 1980.

[Jes80] C.R. Jesshope, "The Implementation of Fast Radix 2 Transforms on Array Processors", *IEEE Transactions on Computers*, Vol. 29, pp. 20-27, Jan. 1980.

[Jon92] D. L. Jones, "Learning Characteristics of Transpose-Form LMS Adaptive Filters", *IEEE Transactions on Circuits and Systems-II: Analog and Digital Signal Processing*, Vol. 39, No. 10, pp. 745-749, Oct. 1992.

[KM93] B.T. Kelley and V.K. Madisetti, "Efficient VLSI Architectures for Arithmetic Fourier Transform (AFT)", *IEEE Transactions on Signal Processing*, Vol. 41, No. 1, pp. 365-383, Jan. 1993.

[Kun88] Kung S.Y. Kung, *VLSI Array Processors*. Prentice Hall, Englewood Cliffs, NJ, 1988.

[Lav96] P. Lavoie, "A High-Speed CMOS Implementation of the Winograd Fourier Transform Algorithm", *IEEE Transactions on Signal Processing*, Vol. 44, No. 8, pp. 2121-2126, Aug. 1996.

[LC93] K.J. Ray Liu and C-T. Chiu, "Unified Parallel Lattice Structures for Time-Recursive Discrete Cosine/Sine/Hartley Transforms", *IEEE Transactions on Signal Processing*, Vol. 41, No. 3, pp. 1357-1377, March 1993.

[Liu93] K.J. Ray Liu, "Novel Parallel Architectures for Short-Time Fourier Transform", *IEEE Transactions on Circuits and Systems II.: Analog and Digital Signal Processing*, Vol. 40, No. 12, pp. 786-790, Dec. 1993.

[NS87] A. Norton and A. J. Silberger, "Parallelization and Performance Analysis of the Cooley-Tukey FFT Algorithm for Shared-Memory Architectures", *IEEE Transactions on Computers*, Vol. 36, No. 5, pp. 581-591, May 1987.

[OS75] A.V. Oppenheim, R.W.Schafer, *Digital Signal Processing*. Englewood Cliffs, NJ: Prentice-Hall, 1975.

[Pea68] M.C. Pease, "An Adaptation of the Fast Forier Transform for Parallel Processing", *Journal of the Association for Computing Machinery*, Vol. 15, No. 2, pp. 252-254, Apr. 1968.

[Pec86] G. Péceli, "A Common Structure for Recursive Discrete Transforms", *IEEE Transactions on Cicuits and Systems*, Vol. 33, pp. 1035-1036, Oct. 1986.

[Pec89] G. Péceli, "Resonator-Based Digital Filters", *IEEE Transactions on Circuits and Systems*, Vol. 36, pp. 156-159, Jan. 1989.

[PF90] G. Péceli, B. Fehér, "Digital Filters Based on Recursive Walsh-Hadamard Transformation", *IEEE Transactions on Circuits and Systems*, Vol. 37, pp. 150-152, Jan. 1990.

[PMP96] M. Padmanabham, K. Martin, G. Péceli, *Feedback-Based Orthogonal Filters*, Kluwer Academic Publishers, Boston/London/ Dordrecht, 1996.

[RG75] L.R. Rabiner, B. Gold, *Theory and Application of Digital Signal Processing*, Prentice-Hall, Englewood Cliffs, NJ, 1975.

[SB85] K.O. Siomalas, B.A. Bowen, "Synthesis of Efficient Pipelined Architectures for Implementing DSP Operations", *IEEE Transactions on Acoustics, Speech and Signal Processing*, Vol. 33, No. 6, pp. 1499-1508, Dec. 1985.

[She+96] M-H. Sheu, S-F. Cheng and M-D. Shieh, "A Pipelined VLSI with Module Structure Design for Discrete Wavelet Transforms", *Proceedings of the IEEE International Symp. on Circiuts and Systems*, May 1996, Atlanta, USA, Vol. IV, pp. 352-355.

[Shy92] J.J. Shynk, "Frequency-Domain and Multirate Adaptive Filtering", *IEEE Signal Processsing Magazine*, pp. 14-37, Jan. 1992.

[SJ90] K. Sapiecha and R. Jarocki, "Modular Architecture for High Performance Implementation of the FFT Algorithm", *IEEE Transactions on Computers*, Vol. 39, No. 12, pp. 1464-1468, Dec. 1990.

[SWL87] M.T. Sun, L. Wu and M.L. Liou, "A Concurrent Architecture for VLSI Implementation of Discrete Cosine Transforms", *IEEE Transactions on Circuits and Systems*, Vol. 34, No. 8, pp. 992-994, Aug. 1987.

[TC93] D.W. Tufts and H. Chen, "Iterative Realization of the Arithmetic Fourier Transform", *IEEE Transactions on Signal Processing*, Vol. 41, No. 1, pp. 152-161, Jan. 1993.

[TF81] S. Tzafestas and G. Frangakis, "A Fast Walsh-Hadamard Signal Processing System", *Mathematics and Computers in Simulation* XXIII, pp. 163-169, 1981.

[Tru+88] T.K. Truong, I.S. Reed, I-S. Hsu, H-C. Shyu and H.M. Shao, "A Pipeline Design of a Fast Prime Factor DFT on a Finite Field", *IEEE Transactions on Computers*, Vol. 37, No. 3, pp. 266-273, 1988.

[TH78] Y. Tadokoro, T. Higuchi, "Discrete Fourier Transform Computation Via the Walsh Transform", *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. 42, No. 9, pp. 614-616, Sep. 1995.

[TH79] Y. Tadokoro and T. Higuchi, "Discrete Fourier Transform Via Walsh Transform", *IEEE Transactions on Acoustics, Speech and Signal Processing*, Vol. 27, pp. 295-296, June 1979.

[Var95] A.R. Várkonyi-Kóczy, "A Recursive Fast Fourier Transformation Algorithm," *IEEE Transactions on Circuits and Systems-II Analog and Digital Signal Processing*, Vol. 42, No. 9, pp. 614-616, Sep. 1995.

[Var96a] A.R. Várkonyi-Kóczy, "Multi-Sine Synthesis and Analysis via Walsh-Hadamard Transformation", *1996 IEEE International Symposium on Circuits and Systems, ISCAS'96*, Atlanta, USA, May 12-15, 1996, Vol. II, pp. 457-460.

[Var96b] A.R. Várkonyi-Kóczy, "Synchronized Multi-Sine Measurements via DSP Methods", *IEEE Instrumentation and Measurement Technology Conference, IMTC'96*, Brussels, Belgium, June 4-6, 1996, pp. 1056-1060.

[Vet83] M. Vetterli, "Tree Structures for Orthogonal Transforms and Applications to the Hadamard Transform", *Signal Processing*, Vol. 5, No. 6, pp. 473-484, 1983.

[VT97] A.R. Várkonyi-Kóczy and S. Theodoridis, "Fast Sliding Transforms in Transform-Domain Adaptive Filtering," *1997 IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP'97*, Munich, Germany, Apr. 20-24, 1997, Vol. 3, pp. 2009-2012.

[WD84] E.H. Wold and A.M. Despain, "Pipeline and Parallel-Pipeline FFT Processors for VLSI Implementations", *IEEE Transactions on Computers*, Vol. 33, No. 5, pp. 414-425, May 1984.

[Whi89] S.A. White, "Applications of Distributed Arithmetic to Digital Signal Processing: A Tutorial Review", *IEEE ASSP Magazine*, pp. 4-19, July 1989.