POLITECNICO DI TORINO

Facoltà di Ingegneria
Corso di Laurea in Ingegneria Elettronica

Tesi di Laurea

Low complexity algorithms and architectures for odd-type DCT hardware implementation

Relatore:
Prof. Maurizio Martina

Candidato:
Luigi Crescenzi

Febbraio 2019

A mio nonno Francesco

Summary

This thesis is centred on the Discrete Cosine Transform Type V (DCT5). This transform has gained great interest in the context of the video coding standards known as Post-HEVC. In particular, the DCT5 was introduced in the JEM software developed by the Joint Video Experts Team (JVET).

The DCT5 is initially defined and algorithms are developed to efficiently perform the computation of this transform. More in detail, the DCT5 is firstly expressed in terms of a Discrete Fourier Transform (DFT). In fact, a length-N DCT5 is related to a length-$(2N-1)$ DFT. The possible algorithms for the DFT are therefore analyzed and several simplifications are made in order to reduce the computational complexity.

The first algorithm under consideration is the Winograd Fourier Transform Algorithm. This is studied for both prime and composite lengths of the DFT. Successively, the Prime Factor Algorithm is investigated and finally, Bluestein's and Rader's algorithms are evaluated.

By considering previous works, the DCT5 is then related to a Discrete Cosine Transform Type II (DCT2). Also in this case, the existing algorithms for the DCT2 are adapted to efficiently compute the DCT5. The DCT5 is then solved via Givens Rotations and finally, by referring to a previously published article, a direct factorization of the DCT5 matrix is also presented.

All the algorithms are developed for lengths 4, 8, 16 and 32, which are the ones of greatest interest in the context of the JEM software. Furthermore, a comparison of the algorithms is carried out. More in detail, they are compared as far as the computational complexity, the regularity and the presence of cascaded multiplications are concerned.

One of the algorithms that are presented is based on a paper by Selesnick and Burrus that describes how to efficiently compute a 31-point DFT. By making simplifications, such an algorithm is adapted to the computation of a 16-point DCT5. The floating-point version of the algorithm is successively used to obtain a fixed-point C-model. The C-model is inserted in the JEM software and the performances are evaluated by referring to the Bjøntegaard delta. According to this model, an architecture is therefore developed and synthesized and its performances are estimated in terms of frequency, area and power consumption.

Contents

Chapter 1

Discrete Cosine Transform Type V

The aim of this chapter is to introduce the Discrete Cosine Transform Type V (DCT5). In particular, a brief definition of the DCT5 is followed by an introduction to the possible algorithms that can be adopted to efficiently compute this transform. The derivation of these algorithms will be instead the main topic of the next chapters.

1.1 DCT5

A Discrete Cosine Transform (DCT) expresses a finite sequence of data points in terms of a sum of cosine functions oscillating at different frequencies. The DCT5 is one of the eight well-known DCTs. In particular, it belongs to the family of the odd-DCTs, together with the DCT6, DCT7 and DCT8. These DCTs have gained great interest in the context of the video coding standards known as Post-High Efficiency Video Coding (Post-HEVC). In particular, the DCT5 was introduced in the JEM software developed by the Joint Video Experts Team (JVET).

1.1.1 Definition

As stated in [1], the DCT5 of a sequence $\{x_k\}$ of N points can be computed by applying the equation:

$$Y_n = \frac{2}{\sqrt{2N-1}} T_n \sum_{k=0}^{N-1} T_k x_k \cos\left(nk \frac{2\pi}{2N-1}\right) \quad \text{for } n = 0,1,...,N-1 \tag{1.1}$$

where

$$T_n = \begin{cases} \frac{1}{\sqrt{2}} & \text{if } \ n = 0 \\ 1 & \text{if } \ n \neq 0 \end{cases}$$

$$T_k = \begin{cases} \frac{1}{\sqrt{2}} & \text{if } k = 0 \\ 1 & \text{if } k \neq 0 \end{cases}$$

As a consequence, given a vector

$$\boldsymbol{x} = \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N-1} \end{bmatrix}$$

the DCT5 can be calculated by performing the matrix product:

$$\boldsymbol{Y} = \begin{bmatrix} Y_0 \\ Y_1 \\ \vdots \\ Y_{N-1} \end{bmatrix} = \left[C_N^{\mathrm{V}}\right] \boldsymbol{x} \tag{1.2}$$

$$= \frac{2}{\sqrt{2N-1}} \begin{bmatrix} \frac{1}{2} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \cdots & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \cos\left(\frac{2\pi}{2N-1}\right) & \cos\left(\frac{4\pi}{2N-1}\right) & \cdots & \cos\left((N-1)\frac{2\pi}{2N-1}\right) \\ \frac{1}{\sqrt{2}} & \cos\left(\frac{4\pi}{2N-1}\right) & \cos\left(\frac{8\pi}{2N-1}\right) & \cdots & \cos\left(2(N-1)\frac{2\pi}{2N-1}\right) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{1}{\sqrt{2}} & \cos\left((N-1)\frac{2\pi}{2N-1}\right) & \cos\left(2(N-1)\frac{2\pi}{2N-1}\right) & \cdots & \cos\left((N-1)^2\frac{2\pi}{2N-1}\right) \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_{N-1} \end{bmatrix}$$

It should be underlined that the matrix $\left[C_N^{\mathrm{V}}\right]$ is symmetric and orthonormal. Therefore, it can be written:

$$\left[C_N^{\mathrm{V}}\right]^{-1} = \left[C_N^{\mathrm{V}}\right]$$

Computational complexity

As shown in table 1.1, the computational cost of the direct implementation of the matrix-vector multiplication described in equation 1.2 is equal to N^2 multiplications and $(N-1) \times N$ additions.

Table 1.1: *Computational complexity of the DCT5 implemented by MVM*

N_{DCT5}	*Number of multiplications*	*Number of additions*
4	16	12
8	64	56
16	256	240
32	1024	992
64	4096	4032
128	16 384	16 256

1.2 Algorithms for the DCT5

Several algorithms can be developed to reduce the computational complexity presented in subsection 1.1.1.

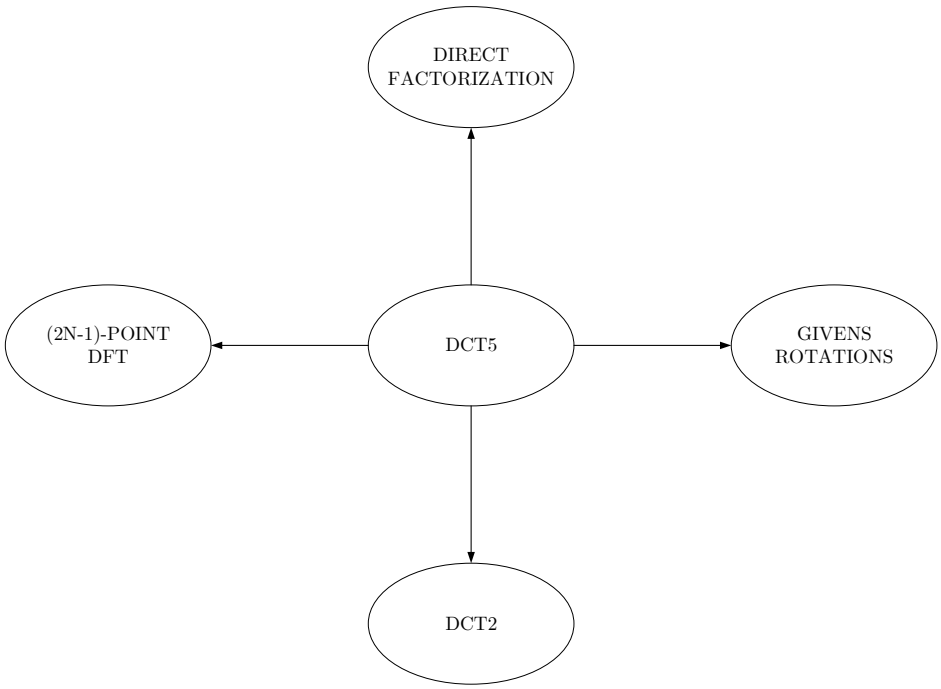


Figure 1.1: *Algorithms for the DCT5*

In particular, as depicted in figure 1.1:

- an N-point DCT5 can be related to a $(2N-1)$-point DFT. Since only $N = 2^n$ is of interest, the DFT can be solved by applying:
  - *Winograd Fourier Transform Algorithm*
  - *Prime Factor Algorithm*
  - *Bluestein's Algorithm*
  - *Rader's Algorithm*
- the DCT5 can be solved via DCT2
- the DCT5 can be solved by applying the *Givens rotations*.
- the DCT5 can be solved via *direct factorization*.

The following chapters contain a detailed description of these algorithms.

Chapter 2

DCT5 via DFT

This chapter shows how an N-point DCT5 can be mapped into a $(2N-1)$-point DFT and presents the possible algorithms for the DFT that can be adopted to compute the DCT5 of a given sequence $\{x_k\}$. In particular, the DCT5 will be firstly solved by adopting the Winograd Fourier transform algorithm. Secondly, algorithms based on the Prime Factor algorithm will be derived. Finally, Bluestein's algorithm and Rader's algorithm will be considered.

2.1 Direct Mapping between DCT5 and DFT

The DCT5 can be derived from the Discrete Fourier Transform by properly rearranging the input vector. In particular, the DCT5 can be expressed as a function of a DFT. In fact, since

$$Y_n = \frac{2}{\sqrt{2N-1}} T_n \sum_{k=0}^{N-1} T_k x_k \cos\left(nk\frac{2\pi}{2N-1}\right) \qquad \text{for } n = 0,1,...,N-1$$

it is possible to define a vector

$$\hat{\boldsymbol{x}} = \begin{bmatrix} \hat{x}_0 \\ \hat{x}_1 \\ \vdots \\ \hat{x}_{N-1} \end{bmatrix} = \begin{bmatrix} \frac{x_0}{\sqrt{2}} \\ x_1 \\ \vdots \\ x_{N-1} \end{bmatrix}$$

and rewrite the summation as

$$S = \sum_{k=0}^{N-1} T_k x_k \cos\left(nk\frac{2\pi}{2N-1}\right) = \sum_{k=0}^{N-1} \hat{x}_k \cos\left(nk\frac{2\pi}{2N-1}\right)$$

This expression can be written as a sum of two terms. The first one is related to the elements of $\hat{\boldsymbol{x}}$ having an even index, while the second one to those which have an odd

index. More in detail, we have

$$S = \sum_{k=0}^{\frac{N}{2}-1} \hat{x}_{2k} \cos\left(n\frac{2\pi}{2N-1}2k\right) + \sum_{k=0}^{\frac{N}{2}-1} \hat{x}_{2k+1} \cos\left(n\frac{2\pi}{2N-1}(2k+1)\right)$$

The second summation can be further rearranged as

$$\sum_{k=0}^{\frac{N}{2}-1} \hat{x}_{2k+1} \cos\left(n\frac{2\pi}{2N-1}(2k+1)\right) = \sum_{k=\frac{N}{2}}^{N-1} \hat{x}_{2N-2k-1} \cos\left(n\frac{2\pi}{2N-1}(2N-1-2k)\right)$$

Since $\cos(x)$ is an even and periodic function, it can be written

$$\sum_{k=\frac{N}{2}}^{N-1} \hat{x}_{2N-2k-1} \cos\left(n\frac{2\pi}{2N-1}(2N-1-2k)\right) = \sum_{k=\frac{N}{2}}^{N-1} \hat{x}_{2N-2k-1} \cos\left(n\frac{2\pi}{2N-1}2k\right)$$

and consequently

$$S = \sum_{k=0}^{\frac{N}{2}-1} \hat{x}_{2k} \cos\left(n\frac{2\pi}{2N-1}2k\right) + \sum_{k=\frac{N}{2}}^{N-1} \hat{x}_{2N-2k-1} \cos\left(n\frac{2\pi}{2N-1}2k\right)$$

A vector $\boldsymbol{q}$ can be defined so that

$$\begin{cases} q_k = \hat{x}_{2k} & \text{for } 0 \leq k \leq \frac{N}{2}-1 \\ q_{N-k-1} = \hat{x}_{2k+1} & \text{for } 0 \leq k \leq \frac{N}{2}-1 \end{cases}$$

and equation 1.1 can be expressed as

$$Y_n = \frac{2}{\sqrt{2N-1}} T_n \sum_{k=0}^{N-1} q_k \cos\left(n\frac{2\pi}{2N-1}2k\right)$$

By introducing the vector

$$\boldsymbol{l} = \begin{bmatrix} l_0 \\ l_1 \\ l_2 \\ l_3 \\ l_4 \\ l_5 \\ \vdots \\ l_{N-2} \\ l_{N-1} \\ l_N \\ l_{N+1} \\ l_{N+2} \\ \vdots \\ l_{2N-2} \end{bmatrix} = \begin{bmatrix} q_0 \\ 0 \\ q_1 \\ 0 \\ q_2 \\ 0 \\ \vdots \\ q_{N/2-1} \\ 0 \\ q_{N/2} \\ 0 \\ q_{N/2+1} \\ \vdots \\ q_{N-1} \end{bmatrix} = \begin{bmatrix} \hat{x}_0 \\ 0 \\ x_2 \\ 0 \\ x_4 \\ 0 \\ \vdots \\ x_{N-2} \\ 0 \\ x_{N-1} \\ 0 \\ x_{N-3} \\ \vdots \\ x_1 \end{bmatrix},$$

the equation

$$\sum_{k=0}^{N-1} q_k \cos\left(n\frac{2\pi}{2N-1}2k\right) = \sum_{k=0}^{2N-2} l_k \cos\left(n\frac{2\pi}{2N-1}k\right)$$

can be derived.

The mathematical expression of a non-normalized DFT applied to the vector $\boldsymbol{l}$ is:

$$Y_{\mathrm{F}n} = \sum_{k=0}^{2N-2} l_k \mathrm{e}^{-\mathrm{j}nk\frac{2\pi}{2N-1}} \qquad \text{for } n = 0,1,...,2N-2.$$

This gives the equation:

$$Y_{\mathrm{F}n} = \sum_{k=0}^{2N-2} l_k \cos\left(n\frac{2\pi}{2N-1}k\right) - \mathrm{j}\sum_{k=0}^{2N-2} l_k \sin\left(n\frac{2\pi}{2N-1}k\right)$$

As a consequence, equation 1.1 can be rewritten as

$$Y_n = \frac{2}{\sqrt{2N-1}} T_n \mathbf{Re}(Y_{\mathrm{F}n}) \qquad \text{for } n = 0,1,...,N-1$$

It is important to notice that if the length of the input vector is even, then the DFT should be performed on an odd length vector.

2.2 The Winograd Fourier Transform Algorithm (WFTA)

The Winograd Fourier Transform Algorithm (described in [2], [3], [4], [5], [6] and [7]) remarkably reduces the number of multiplications needed to compute the DFT. It does not increase the number of additions in many cases. This algorithm can be directly used to compute the DFT by taking into account the prime factorization of the input vector length, which is, for some cases, reported in table 2.1.

Table 2.1: *Prime Factorization of N_{DFT} for different lengths of the DCT5*

N_{DCT5}	N_{DFT}	N_{DFT} *Prime Factorization*
4	7	7
8	15	3×5
16	31	31
32	63	$3^2 \times 7$

2.2.1 Winograd Short-N DFT Modules

As stated in [4], Winograd short-N DFT modules are the building blocks for developing the WFTA for longer lengths. They are defined for powers of prime numbers and are

based on a fast cyclic convolution algorithm, which generally uses the theoretically minimum number of multiplications. In mathematical terms, Winograd's algorithm obtains a canonical decomposition of the DFT matrix as shown below:

$$[D_{N_{\mathrm{DFT}}}] = [S_{N_{\mathrm{DFT}}}]\,[C_{N_{\mathrm{DFT}}}]\,[T_{N_{\mathrm{DFT}}}]$$

where

- $[D_{N_{\mathrm{DFT}}}]$ is the $N_{\mathrm{DFT}} \times N_{\mathrm{DFT}}$ DFT matrix;
- $[S_{N_{\mathrm{DFT}}}]$ is an $N_{\mathrm{DFT}} \times J$ matrix having 0, 1, -1 only as elements;
- $[C_{N_{\mathrm{DFT}}}]$ is a $J \times J$ diagonal matrix with each element purely real or purely imaginary;
- $[T_{N_{\mathrm{DFT}}}]$ is a $J \times N_{\mathrm{DFT}}$ matrix having 0,1, -1 only as elements.

As a consequence, $[S_{N_{\mathrm{DFT}}}]$ and $[T_{N_{\mathrm{DFT}}}]$ are just addition matrices and $[C_{N_{\mathrm{DFT}}}]$ is the multiplier matrix. The number of needed multiplications is equal to J. If N_{DFT} is small, J is close to N_{DFT}.

2.2.2 WFTA for DCT5 ($N = 4$)

A DCT5 of length 4 can be mapped into a DFT of length 7 as described in section 2.1. In particular, the DCT5 equation can be written as

$$Y_n = \mathrm{c}_4 T_n \mathbf{Re}(Y_{\mathrm{F}n}) \qquad \text{for } n = 0,1,2,3 \tag{2.1}$$

where

- $\mathrm{c}_4 = \frac{2}{\sqrt{7}}$
- $\mathbf{Re}(Y_{\mathrm{F}n}) = \mathbf{Re}\left(\sum_{k=0}^{6} l_k \mathrm{e}^{-\mathrm{j}nk\frac{2\pi}{7}}\right)$

and

$$\boldsymbol{l} = \begin{bmatrix} l_0 \\ l_1 \\ l_2 \\ l_3 \\ l_4 \\ l_5 \\ l_6 \end{bmatrix} = \begin{bmatrix} \hat{x}_0 \\ 0 \\ \hat{x}_2 \\ 0 \\ \hat{x}_3 \\ 0 \\ \hat{x}_1 \end{bmatrix} = \begin{bmatrix} \frac{x_0}{\sqrt{2}} \\ 0 \\ x_2 \\ 0 \\ x_3 \\ 0 \\ x_1 \end{bmatrix}$$

Equation 2.1 can be further rearranged to give

$$Y_n = T_n \mathbf{Re}(\mathrm{c}_4 Y_{\mathrm{F}n}) = T_n \mathbf{Re}(\hat{Y}_{\mathrm{F}_n})$$

where:

$$\hat{\boldsymbol{Y}}_{\mathbf{F}} = \begin{bmatrix} \hat{Y}_{\mathrm{F}_0} \\ \hat{Y}_{\mathrm{F}_1} \\ \hat{Y}_{\mathrm{F}_2} \\ \hat{Y}_{\mathrm{F}_3} \\ \hat{Y}_{\mathrm{F}_4} \\ \hat{Y}_{\mathrm{F}_5} \\ \hat{Y}_{\mathrm{F}_6} \end{bmatrix} = \mathrm{c}_4\,[S_7]\,[C_7]\,[T_7]\,\boldsymbol{l} = [S_7]\left[\hat{C}_7\right][T_7]\,\boldsymbol{l}$$

In particular:

$$[T_7] = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & -1 & -1 & 0 & 1 \\ 0 & 0 & -1 & 1 & 1 & -1 & 0 \\ 0 & -1 & 1 & 0 & 0 & 1 & -1 \\ 0 & 1 & 1 & -1 & 1 & -1 & -1 \\ 0 & 1 & 0 & 1 & -1 & 0 & -1 \\ 0 & 0 & -1 & -1 & 1 & 1 & 0 \\ 0 & -1 & 1 & 0 & 0 & -1 & 1 \end{bmatrix}$$

$$[S_7] = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & -1 & 0 & -1 & 1 & -1 & 0 & -1 \\ 1 & 1 & 0 & -1 & 1 & -1 & 0 & 1 & -1 \\ 1 & 1 & 0 & -1 & 1 & 1 & 0 & -1 & 1 \\ 1 & 1 & -1 & 0 & -1 & -1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & -1 & -1 & -1 & 0 \end{bmatrix}$$

and $\left[\hat{C}_7\right]$ is a diagonal matrix whose main diagonal contains the elements of the vector

$$\hat{\boldsymbol{C}}_{\boldsymbol{7}} = \frac{2}{\sqrt{7}} \begin{bmatrix} 1 \\ \frac{1}{3}\left(\cos(\theta) + \cos(2\theta) + \cos(3\theta)\right) - 1 \\ \frac{1}{3}\left(2\cos(\theta) - \cos(2\theta) - \cos(3\theta)\right) \\ \frac{1}{3}\left(\cos(\theta) - 2\cos(2\theta) + \cos(3\theta)\right) \\ \frac{1}{3}\left(\cos(\theta) + \cos(2\theta) - 2\cos(3\theta)\right) \\ \frac{1}{3}\mathrm{j}\left(\sin(\theta) + \sin(2\theta) - \sin(3\theta)\right) \\ \frac{1}{3}\mathrm{j}\left(2\sin(\theta) - \sin(2\theta) + \sin(3\theta)\right) \\ \frac{1}{3}\mathrm{j}\left(\sin(\theta) - 2\sin(2\theta) - \sin(3\theta)\right) \\ \frac{1}{3}\mathrm{j}\left(\sin(\theta) + \sin(2\theta) + 2\sin(3\theta)\right) \end{bmatrix} = \begin{bmatrix} \hat{\mathrm{C}}_0 \\ \hat{\mathrm{C}}_1 \\ \hat{\mathrm{C}}_2 \\ \hat{\mathrm{C}}_3 \\ \hat{\mathrm{C}}_4 \\ \hat{\mathrm{C}}_5 \\ \hat{\mathrm{C}}_6 \\ \hat{\mathrm{C}}_7 \\ \hat{\mathrm{C}}_8 \end{bmatrix} \qquad \theta = -\frac{2\pi}{7}$$

By making use of these matrices, equation 2.1 yields an algorithm, which is composed of five main steps:

- Pre-Normalization
- Pre-Additions
- Multiplications
- Post-Additions
- Post-Normalization

In the following, each of these steps will be analyzed.

Pre-Normalization

The multiplication

$$M_{\mathrm{n1}} = \mathrm{C}_{\mathrm{norm}} x_0 = \frac{1}{\sqrt{2}} x_0$$

is performed.

Pre-Additions

The additions related to this step are derived from the matrix-vector multiplication

$$\boldsymbol{A} = [T_7] \begin{bmatrix} M_{\mathrm{n1}} \\ 0 \\ x_2 \\ 0 \\ x_3 \\ 0 \\ x_1 \end{bmatrix} = \begin{bmatrix} M_{\mathrm{n1}} + x_1 + x_2 + x_3 \\ x_1 + x_2 + x_3 \\ x_1 - x_3 \\ x_3 - x_2 \\ x_2 - x_1 \\ U \\ U \\ U \\ U \end{bmatrix}$$

The elements labeled as "U"are not of interest since they are only needed for the computation of the DFT imaginary part. The calculation of the other elements instead requires 6 sums. These are reported in table 2.2.

Table 2.2: *WFTA for DCT5 ($N = 4$): Pre-Additions*

Op. Name	*Operation*	*Op. Name*	*Operation*	*Op. Name*	*Operation*
a_0	$x_1 + x_2$	a_2	$x_2 - x_1$	a_4	$x_3 - x_2$
a_1	$x_3 + a_0$	a_3	$x_1 - x_3$	a_5	$M_{\mathrm{n1}} + a_1$

Hence, we can write:

$$\boldsymbol{A} = \begin{bmatrix} a_5 \\ a_1 \\ a_3 \\ a_4 \\ a_2 \\ U \\ U \\ U \\ U \end{bmatrix}$$

Multiplications

The multiplications (reported in table 2.3) derive from the matrix-vector product

$$\boldsymbol{M} = \left[\hat{C}_7\right] \boldsymbol{A} = \begin{bmatrix} \hat{\mathrm{C}}_0 a_5 \\ \hat{\mathrm{C}}_1 a_1 \\ \hat{\mathrm{C}}_2 a_3 \\ \hat{\mathrm{C}}_3 a_4 \\ \hat{\mathrm{C}}_4 a_2 \\ U \\ U \\ U \\ U \end{bmatrix} = \begin{bmatrix} M_0 \\ M_1 \\ M_2 \\ M_3 \\ M_4 \\ U \\ U \\ U \\ U \end{bmatrix}$$

Table 2.3: *WFTA for DCT5 ($N = 4$): Multiplications*

Op. Name	*Operation*	*Op. Name*	*Operation*	*Op. Name*	*Operation*
M_0	$\hat{\mathrm{C}}_0 a_5$	M_2	$\hat{\mathrm{C}}_2 a_3$	M_4	$\hat{\mathrm{C}}_4 a_2$
M_1	$\hat{\mathrm{C}}_1 a_1$	M_3	$\hat{\mathrm{C}}_3 a_4$		

Post-Additions

The additions related to this step are obtained by performing the matrix-vector multiplication

$$\boldsymbol{Z} = [S_7]\, \boldsymbol{M} = \begin{bmatrix} M_0 \\ M_0 + M_1 + M_2 + M_3 \\ M_0 + M_1 - M_2 - M_4 \\ M_0 + M_1 - M_3 + M_4 \\ M_0 + M_1 - M_3 + M_4 \\ M_0 + M_1 - M_2 - M_4 \\ M_0 + M_1 + M_2 + M_3 \end{bmatrix}$$

It should be highlighted that this result is obtained by setting the elements labeled as "U"equal to zero. In fact, since only the real part of the DFT is relevant to our goal, there is no need to take these elements into account.

Moreover, it should be noticed that only the first four components of $\boldsymbol{Z}$ have to be considered. Nevertheless, even though the other elements are not of interest, they are equal to those components which are instead significant. This is due to the fact that the inputs of the DFT are real. Indeed, the real part of the DFT of an odd-length real sequence is a palindromic sequence except for the first element.

All the elements of the vector $\boldsymbol{Z}$ can be computed by performing the additions presented in table 2.4. In particular, it can be written:

$$\boldsymbol{Z} = \begin{bmatrix} M_0 & a_{10} & a_{12} & a_{11} & a_{11} & a_{12} & a_{10} \end{bmatrix}^{\mathrm{T}}$$

Table 2.4: *WFTA for DCT5 ($N = 4$): Post-Additions*

Op. Name	*Operation*
a_6	$M_0 + M_1$
a_7	$M_2 + M_3$
a_8	$M_4 - M_3$

Op. Name	*Operation*
a_9	$-M_2 - M_4$
a_{10}	$a_6 + a_7$

Op. Name	*Operation*
a_{11}	$a_6 + a_8$
a_{12}	$a_6 + a_9$

Post-Normalization

The multiplication

$$M_{\mathrm{n2}} = \mathrm{C}_{\mathrm{norm}} M_0 = \frac{1}{\sqrt{2}} M_0$$

is performed.

Final Algorithm

The final version of the algorithm is reported in table 2.5. It should be highlighted that the computational cost of the algorithm is reduced to 7 multiplications and 13 additions.

Table 2.5: *WFTA applied to the DCT5 for $N = 4$*

Op. Name	*Operation*
M_{n1}	$\mathrm{C}_{\mathrm{norm}} x_0$
a_0	$x_1 + x_2$
a_1	$x_3 + a_0$
a_2	$x_2 - x_1$
a_3	$x_1 - x_3$
a_4	$x_3 - x_2$
a_5	$M_{\mathrm{n1}} + a_1$

Op. Name	*Operation*
M_0	$\hat{\mathrm{C}}_0 a_5$
M_1	$\hat{\mathrm{C}}_1 a_1$
M_2	$\hat{\mathrm{C}}_2 a_3$
M_3	$\hat{\mathrm{C}}_3 a_4$
M_4	$\hat{\mathrm{C}}_4 a_2$
a_6	$M_0 + M_1$

Op. Name	*Operation*
a_7	$M_2 + M_3$
a_8	$M_4 - M_3$
a_9	$-M_2 - M_4$
Y_0	$\mathrm{C}_{\mathrm{norm}} M_0$
Y_1	$a_6 + a_7$
Y_2	$a_6 + a_9$
Y_3	$a_6 + a_8$

Remarks

The DFT Module used to derive the algorithm presented in table 2.5 is built by following these steps:

1. the DFT is expressed as a function of a circular convolution by using Rader's theorem;
2. the circular convolution is solved by making use of the Winograd short convolution algorithms.

It is also possible to directly express the DCT5 as a function of a circular convolution. More in detail, by exploiting elementary trigonometric identities, it can be written:

$$\left[C_4^{\mathrm{V}}\right] = \frac{2}{\sqrt{7}} \begin{bmatrix} \frac{1}{2} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \cos\left(\frac{2\pi}{7}\right) & \cos\left(\frac{4\pi}{7}\right) & \cos\left(\frac{6\pi}{7}\right) \\ \frac{1}{\sqrt{2}} & \cos\left(\frac{4\pi}{7}\right) & \cos\left(\frac{6\pi}{7}\right) & \cos\left(\frac{2\pi}{7}\right) \\ \frac{1}{\sqrt{2}} & \cos\left(\frac{6\pi}{7}\right) & \cos\left(\frac{2\pi}{7}\right) & \cos\left(\frac{4\pi}{7}\right) \end{bmatrix}$$

By reversing the position of the third and fourth row, the matrix

$$\left[\hat{C}_4^{\mathrm{V}}\right] = \frac{2}{\sqrt{7}} \begin{bmatrix} \frac{1}{2} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \cos\left(\frac{2\pi}{7}\right) & \cos\left(\frac{4\pi}{7}\right) & \cos\left(\frac{6\pi}{7}\right) \\ \frac{1}{\sqrt{2}} & \cos\left(\frac{6\pi}{7}\right) & \cos\left(\frac{2\pi}{7}\right) & \cos\left(\frac{4\pi}{7}\right) \\ \frac{1}{\sqrt{2}} & \cos\left(\frac{4\pi}{7}\right) & \cos\left(\frac{6\pi}{7}\right) & \cos\left(\frac{2\pi}{7}\right) \end{bmatrix}$$

is obtained.

Since the submatrix

$$\left[C_{\mathrm{s}}\right] = \frac{2}{\sqrt{7}} \begin{bmatrix} \cos\left(\frac{2\pi}{7}\right) & \cos\left(\frac{4\pi}{7}\right) & \cos\left(\frac{6\pi}{7}\right) \\ \cos\left(\frac{6\pi}{7}\right) & \cos\left(\frac{2\pi}{7}\right) & \cos\left(\frac{4\pi}{7}\right) \\ \cos\left(\frac{4\pi}{7}\right) & \cos\left(\frac{6\pi}{7}\right) & \cos\left(\frac{2\pi}{7}\right) \end{bmatrix}$$

is a circulant matrix, the computation of the DCT5 can be derived from the circular convolution between the vectors

$$\boldsymbol{g} = \frac{2}{\sqrt{7}} \begin{bmatrix} \cos\left(\frac{2\pi}{7}\right) \\ \cos\left(\frac{6\pi}{7}\right) \\ \cos\left(\frac{4\pi}{7}\right) \end{bmatrix} \quad \text{and} \quad \boldsymbol{d} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

This circular convolution can be solved by adopting the algorithm reported in [6]. More in detail, by defining the constants:

$$\mathrm{G}_0 = \frac{2}{3\sqrt{7}} \left(\cos\frac{2\pi}{7} + \cos\frac{4\pi}{7} + \cos\frac{6\pi}{7} \right) \qquad \mathrm{G}_1 = \frac{2}{\sqrt{7}} \left(\cos\frac{2\pi}{7} - \cos\frac{4\pi}{7} \right)$$

$$\mathrm{G}_2 = \frac{2}{\sqrt{7}} \left(\cos\frac{6\pi}{7} - \cos\frac{4\pi}{7} \right) \qquad \mathrm{G}_3 = \frac{2}{3\sqrt{7}} \left(\cos\frac{2\pi}{7} + \cos\frac{6\pi}{7} - 2\cos\frac{4\pi}{7} \right)$$

$$\mathrm{G}_4 = \sqrt{\frac{2}{7}} \qquad \mathrm{G}_5 = \frac{1}{\sqrt{2}}$$

the algorithm presented in table 2.6 is derived.

It should be highlighted that the identity

$$\cos\frac{2\pi}{7} + \cos\frac{4\pi}{7} + \cos\frac{6\pi}{7} = -0.5$$

is exploited in the derivation of the algorithm.

Table 2.6: *Algorithm for the DCT5 ($N = 4$) executed by performing a circular convolution*

Op. Name	*Operation*	*Op. Name*	*Operation*	*Op. Name*	*Operation*
a_0	$x_1 + x_2$	M_2	$\mathrm{G}_2 a_3$	a_9	$a_7 - a_5$
a_1	$x_3 + a_0$	M_3	$\mathrm{G}_3 a_4$	Y_3	$a_9 - a_6$
a_2	$x_1 - x_3$	M_4	$\mathrm{G}_4 x_0$	Y_2	$a_7 + a_6$
a_3	$x_2 - x_3$	a_5	$M_1 - M_3$	a_{12}	$2M_0 + 4M_0$
a_4	$a_2 + a_3$	a_6	$M_2 - M_3$	a_{13}	$M_4 - a_{12}$
M_0	$\mathrm{G}_0 a_1$	a_7	$M_0 + M_4$	Y_0	$\mathrm{G}_5 a_{13}$
M_1	$\mathrm{G}_1 a_2$	Y_1	$a_7 + a_5$		

2.2.3 The Winograd large fast Fourier transform

Supposing that the length N_{DFT} of the DFT can be expressed as a product of two numbers ($N1$,$N2$) that are coprime, it can be demonstrated that

$$[P_o]\,\boldsymbol{Y_F} = ([D_{N1}] \otimes [D_{N2}])\,[P_i]\,\boldsymbol{l}$$

where:

- $[P_o]$ is a permutation matrix;
- $[D_{N1}]$, $[D_{N2}]$ are the DFT matrices for length N_1 and N_2;
- $\otimes$ is the symbol of the Kronecker product;
- $[P_i]$ is a permutation matrix.

Since:

$$[D_{N1}] = [S_{N1}]\,[C_{N1}]\,[T_{N1}]$$

and

$$[D_{N2}] = [S_{N2}]\,[C_{N2}]\,[T_{N2}]$$

it can be written

$$[D_{N1}] \otimes [D_{N2}] = ([S_{N1}] \otimes [S_{N2}])\,([C_{N1}] \otimes [C_{N2}])\,([T_{N1}] \otimes [T_{N2}]) \tag{2.2}$$

where the Kronecker products $[S] = [S_{N1}] \otimes [S_{N2}]$ and $[T] = [T_{N1}] \otimes [T_{N2}]$ are matrices of zeros and ones, and the Kronecker product $[C] = [C_{N1}] \otimes [C_{N2}]$ is again a diagonal matrix with each element purely real or purely imaginary.

Hence, equation 2.2 leads to

$$[Z] = [S_{N1}]\,([S_{N2}]\,[C_{N2\times N1}] \circ [T_{N2}]\,([T_{N1}]\,[z])^{\mathrm{T}})^{\mathrm{T}}$$

where:

- $[C_{N2\times N1}] = \boldsymbol{C_{N2}} \boldsymbol{C_{N1}^{\mathrm{T}}}$, with $\boldsymbol{C_{N2}}$ and $\boldsymbol{C_{N1}}$ being the vectors, which respectively contain the diagonal elements of the matrices $[C_{N2}]$ and $[C_{N1}]$;

- $\circ$ is the symbol of the element by element matrix multiplication;
- $[z]$ is an $N1 \times N2$ matrix containing the elements of the input vector $\boldsymbol{l}$;
- $[Z]$ is an $N1 \times N2$ matrix containing the elements of the output vector $\boldsymbol{Y_F}$.

In the following, the procedure needed to derive the matrix $[z]$ will be described as well as the one used to extract the elements of the vector $\boldsymbol{Y_F}$ from the matrix $[Z]$.

Input matrix

In order to build the matrix $[z]$, a permutation is firstly applied to the vector $\boldsymbol{l}$. Therefore, the vector $\hat{\boldsymbol{l}}$ is obtained as

$$\hat{\boldsymbol{l}} = [P_i]\,\boldsymbol{l}$$

where $[P_i]$ is the input permutation matrix. This matrix is derived according to the following steps:

1. A couple (n_1,n_2) is assigned to each element of $\hat{\boldsymbol{l}}$ with $0 \leq n_1 \leq (N1-1)$ and $0 \leq n_2 \leq (N2-1)$. More in detail, the couple (n_1,n_2) is obtained by progressively increasing n_2 and n_1. This is better explained by the following example. Consider, for instance, $N_{\mathrm{DFT}} = 15 = 3 \times 5 = N1 \times N2$. The assignments shown in table 2.7 are obtained.

2. The corresponding index of the vector $\boldsymbol{l}$ is derived by applying the equation:

$$i_{\boldsymbol{l}} = (n_1 \times N2 + n_2 \times N1) \bmod N_{\mathrm{DFT}}$$

   The indexes obtained for the case presented above are reported in table 2.7.

Table 2.7: *Permuted indexes calculation for* $N1 = 3$ *and* $N2 = 5$

$i_{\hat{\boldsymbol{l}}}$	(n_1,n_2)	$i_{\boldsymbol{l}}$
0	(0,0)	$(0 \times 5 + 0 \times 3) \bmod 15 = 0$
1	(0,1)	$(0 \times 5 + 1 \times 3) \bmod 15 = 3$
2	(0,2)	$(0 \times 5 + 2 \times 3) \bmod 15 = 6$
3	(0,3)	$(0 \times 5 + 3 \times 3) \bmod 15 = 9$
4	(0,4)	$(0 \times 5 + 4 \times 3) \bmod 15 = 12$
5	(1,0)	$(1 \times 5 + 0 \times 3) \bmod 15 = 5$
6	(1,1)	$(1 \times 5 + 1 \times 3) \bmod 15 = 8$
7	(1,2)	$(1 \times 5 + 2 \times 3) \bmod 15 = 11$
8	(1,3)	$(1 \times 5 + 3 \times 3) \bmod 15 = 14$
9	(1,4)	$(1 \times 5 + 4 \times 3) \bmod 15 = 2$
10	(2,0)	$(2 \times 5 + 0 \times 3) \bmod 15 = 10$
11	(2,1)	$(2 \times 5 + 1 \times 3) \bmod 15 = 13$
12	(2,2)	$(2 \times 5 + 2 \times 3) \bmod 15 = 1$
13	(2,3)	$(2 \times 5 + 3 \times 3) \bmod 15 = 4$
14	(2,4)	$(2 \times 5 + 4 \times 3) \bmod 15 = 7$

3. The permutation matrix is built by setting the elements in position $(i_{\hat{\boldsymbol{l}}}, i_{\boldsymbol{l}})$ equal to one. Therefore, for the case presented above, the ones of the permutation matrix are in positions: (0,0), (1,3), (2,6), (3,9), (4,12), (5,5), (6,8), (7,11), (8,14), (9,2), (10,10), (11,13), (12,1), (13,4) and (14,7). Hence, for this particular case, the matrix is:

$$[P_i] = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

The elements of $\hat{\boldsymbol{l}}$ are then ordered along the rows of the matrix $[z]$. In particular, for the case presented above, it can be written:

$$[z] = \begin{bmatrix} \hat{l}_0 & \hat{l}_1 & \hat{l}_2 & \hat{l}_3 & \hat{l}_4 \\ \hat{l}_5 & \hat{l}_6 & \hat{l}_7 & \hat{l}_8 & \hat{l}_9 \\ \hat{l}_{10} & \hat{l}_{11} & \hat{l}_{12} & \hat{l}_{13} & \hat{l}_{14} \end{bmatrix}$$

Output matrix

The vector $\boldsymbol{Y_F}$ is stored in the two-dimensional array $[Z]$ by starting in the upper left corner and listing the components down the "extended diagonal ". Since the number of rows and the number of columns are relatively prime, the extended diagonal passes through every element of the array. For the case considered above, we have:

$$[Z] = \begin{bmatrix} Z_{0,0} & Z_{0,1} & Z_{0,2} & Z_{0,3} & Z_{0,4} \\ Z_{1,0} & Z_{1,1} & Z_{1,2} & Z_{1,3} & Z_{1,4} \\ Z_{2,0} & Z_{2,1} & Z_{2,2} & Z_{2,3} & Z_{2,4} \end{bmatrix} = \begin{bmatrix} Y_{\mathrm{F}0} & Y_{\mathrm{F}6} & Y_{\mathrm{F}12} & Y_{\mathrm{F}3} & Y_{\mathrm{F}9} \\ Y_{\mathrm{F}10} & Y_{\mathrm{F}1} & Y_{\mathrm{F}7} & Y_{\mathrm{F}13} & Y_{\mathrm{F}4} \\ Y_{\mathrm{F}5} & Y_{\mathrm{F}11} & Y_{\mathrm{F}2} & Y_{\mathrm{F}8} & Y_{\mathrm{F}14} \end{bmatrix}$$

2.2.4 WFTA for DCT5 ($N = 8$)

A DCT5 of length 8 can be mapped into a DFT of length 15 as described in section 2.1. In particular, the DCT5 equation can be written as

$$Y_n = \mathrm{c}_8 T_n \mathbf{Re}(Y_{\mathrm{F}n}) \qquad \text{for } n = 0,1,...,7 \tag{2.3}$$

where

- $\mathrm{c}_8 = \frac{2}{\sqrt{15}}$
- $\mathbf{Re}(Y_{\mathrm{F}n}) = \mathbf{Re}\left(\sum_{k=0}^{14} l_k \mathrm{e}^{-\mathrm{j}nk\frac{2\pi}{15}}\right)$

and

$$\boldsymbol{l} = \begin{bmatrix} l_0 \\ l_1 \\ l_2 \\ l_3 \\ l_4 \\ l_5 \\ l_6 \\ l_7 \\ l_8 \\ l_9 \\ l_{10} \\ l_{11} \\ l_{12} \\ l_{13} \\ l_{14} \end{bmatrix} = \begin{bmatrix} \hat{x}_0 \\ 0 \\ \hat{x}_2 \\ 0 \\ \hat{x}_4 \\ 0 \\ \hat{x}_6 \\ 0 \\ \hat{x}_7 \\ 0 \\ \hat{x}_5 \\ 0 \\ \hat{x}_3 \\ 0 \\ \hat{x}_1 \end{bmatrix} = \begin{bmatrix} \frac{x_0}{\sqrt{2}} \\ 0 \\ x_2 \\ 0 \\ x_4 \\ 0 \\ x_6 \\ 0 \\ x_7 \\ 0 \\ x_5 \\ 0 \\ x_3 \\ 0 \\ x_1 \end{bmatrix}$$

Equation 2.3 can be further rearranged to give

$$Y_n = T_n \mathbf{Re}(\mathrm{c}_8 Y_{\mathrm{F}n}) \ = T_n \mathbf{Re}(\hat{Y}_{\mathrm{F}n})$$

The elements of $\hat{\mathbf{Y}}_{\mathbf{F}}$ are contained in the matrix $[Z]$ which is calculated by applying the equation

$$[Z] = \mathrm{c}_8 \left[S_3\right] \left(\left[S_5\right] \left[C_{5\times 3}\right] \circ \left[T_5\right] \left(\left[T_3\right] [z]\right)^{\mathrm{T}}\right)^{\mathrm{T}} = \left[S_3\right] \left(\left[S_5\right] \left[\hat{C}_{5\times 3}\right] \circ \left[T_5\right] \left(\left[T_3\right] [z]\right)^{\mathrm{T}}\right)^{\mathrm{T}} \quad (2.4)$$

where

$$\left[\hat{C}_{5\times 3}\right] = \mathrm{c}_8 \left[C_{5\times 3}\right]$$

Equation 2.4 yields an algorithm composed of the following steps:

- Pre-Normalization
- Pre-Additions
- Multiplications
- Post-Additions
- Post-Normalization

In the following, each of these steps will be analyzed.

Pre-Normalization

The multiplication

$$M_{\mathrm{n1}} = \mathrm{C}_{\mathrm{norm}} x_0 = \frac{1}{\sqrt{2}} x_0$$

is performed.

Pre-Additions

The additions related to this step are obtained by performing the matrix products

$$[A] = [T_5] \left([T_3] \, [z] \right)^{\mathrm{T}}$$

In particular, the input permutation matrix $[P_i]$ is applied to the vector $\boldsymbol{l}$:

$$\hat{\boldsymbol{l}} = [P_i] \, \boldsymbol{l} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} l_0 \\ l_1 \\ l_2 \\ l_3 \\ l_4 \\ l_5 \\ l_6 \\ l_7 \\ l_8 \\ l_9 \\ l_{10} \\ l_{11} \\ l_{12} \\ l_{13} \\ l_{14} \end{bmatrix} = \begin{bmatrix} l_0 \\ l_3 \\ l_6 \\ l_9 \\ l_{12} \\ l_5 \\ l_8 \\ l_{11} \\ l_{14} \\ l_2 \\ l_{10} \\ l_{13} \\ l_1 \\ l_4 \\ l_7 \end{bmatrix}$$

The elements of $\hat{\boldsymbol{l}}$ are then ordered along the rows of the matrix $[z]$:

$$[z] = \begin{bmatrix} l_0 & l_3 & l_6 & l_9 & l_{12} \\ l_5 & l_8 & l_{11} & l_{14} & l_2 \\ l_{10} & l_{13} & l_1 & l_4 & l_7 \end{bmatrix} = \begin{bmatrix} M_{\mathrm{n1}} & 0 & x_6 & 0 & x_3 \\ 0 & x_7 & 0 & x_1 & x_2 \\ x_5 & 0 & 0 & x_4 & 0 \end{bmatrix}$$

Since:

$$[T_3] = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & -1 \end{bmatrix} \quad \text{and} \quad [T_5] = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & -1 & -1 & 1 \\ 0 & 1 & 0 & 0 & -1 \\ 0 & 1 & -1 & 1 & -1 \\ 0 & 0 & -1 & 1 & 0 \end{bmatrix}$$

it can be derived

$$
\begin{aligned}
&[A] = [T_5]\left([T_3]\,[z]\right)^{\mathrm{T}} \\
&= \begin{bmatrix}
M_{\mathrm{n1}} + x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 & x_1 + x_2 + x_4 + x_5 + x_7 & U \\
x_1 + x_2 + x_3 + x_4 + x_6 + x_7 & x_1 + x_2 + x_4 + x_7 & U \\
-x_1 + x_2 + x_3 - x_4 - x_6 + x_7 & -x_1 + x_2 - x_4 + x_7 & U \\
U & U & -x_2 + x_7 \\
U & U & x_1 - x_2 - x_4 + x_7 \\
U & U & x_1 - x_4
\end{bmatrix}
\end{aligned}
$$

where the elements labeled as "U"are not calculated because they would only be used to compute the imaginary part of the DFT, which is not needed. Fourteen additions are instead required to compute the other elements of the matrix $[A]$. These are reported in table 2.8.

Table 2.8: *WFTA for DCT5 ($N = 8$): Pre-Additions*

Op. Name	*Operation*	*Op. Name*	*Operation*
a_0	$x_1 + x_4$	a_7	$a_3 + a_2$
a_1	$x_2 + x_7$	a_8	$x_3 - x_6$
a_2	$x_3 + x_6$	a_9	$a_5 + a_8$
a_3	$a_0 + a_1$	a_{10}	$x_7 - x_2$
a_4	$a_3 + x_5$	a_{11}	$x_1 - x_4$
a_5	$a_1 - a_0$	a_{12}	$a_{10} + a_{11}$
a_6	$a_4 + a_2$	a_{13}	$a_6 + M_{\mathrm{n1}}$

Therefore, it can be written:

$$
[A] = \begin{bmatrix}
a_{13} & a_4 & U \\
a_7 & a_3 & U \\
a_9 & a_5 & U \\
U & U & a_{10} \\
U & U & a_{12} \\
U & U & a_{11}
\end{bmatrix}
$$

Multiplications

The multiplications (reported in table 2.9) derive from the element-by-element matrix product

$$
[M] = \left[\hat{C}_{5\times 3}\right] \circ [A]
$$

The matrix $\left[\hat{C}_{5\times 3}\right]$ can be obtained from the diagonal elements of the matrices $[C_5]$ and $[C_3]$. Since

$$[C_3] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \mathrm{k}_1 & 0 \\ 0 & 0 & \mathrm{jk}_2 \end{bmatrix} \quad \text{and} \quad [C_5] = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & \mathrm{k}_3 & 0 & 0 & 0 & 0 \\ 0 & 0 & \mathrm{k}_4 & 0 & 0 & 0 \\ 0 & 0 & 0 & \mathrm{jk}_5 & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathrm{jk}_6 & 0 \\ 0 & 0 & 0 & 0 & 0 & \mathrm{jk}_7 \end{bmatrix}$$

it follows that

$$\left[\hat{C}_{5\times 3}\right] = c_8 \begin{bmatrix} 1 \\ \mathrm{k}_3 \\ \mathrm{k}_4 \\ \mathrm{jk}_5 \\ \mathrm{jk}_6 \\ \mathrm{jk}_7 \end{bmatrix} \begin{bmatrix} 1 & \mathrm{k}_1 & \mathrm{jk}_2 \end{bmatrix} = \begin{bmatrix} \mathrm{c}_8 & \mathrm{c}_8\mathrm{k}_1 & \mathrm{jc}_8\mathrm{k}_2 \\ \mathrm{c}_8\mathrm{k}_3 & \mathrm{c}_8\mathrm{k}_1\mathrm{k}_3 & \mathrm{jc}_8\mathrm{k}_2\mathrm{k}_3 \\ \mathrm{c}_8\mathrm{k}_4 & \mathrm{c}_8\mathrm{k}_1\mathrm{k}_4 & \mathrm{jc}_8\mathrm{k}_2\mathrm{k}_4 \\ \mathrm{jc}_8\mathrm{k}_5 & \mathrm{jc}_8\mathrm{k}_1\mathrm{k}_5 & -\mathrm{c}_8\mathrm{k}_2\mathrm{k}_5 \\ \mathrm{jc}_8\mathrm{k}_6 & \mathrm{jc}_8\mathrm{k}_1\mathrm{k}_6 & -\mathrm{c}_8\mathrm{k}_2\mathrm{k}_6 \\ \mathrm{jc}_8\mathrm{k}_7 & \mathrm{jc}_8\mathrm{k}_1\mathrm{k}_7 & -\mathrm{c}_8\mathrm{k}_2\mathrm{k}_7 \end{bmatrix}$$

where:

- $\mathrm{k}_1 = \cos\left(-\frac{2\pi}{3}\right) - 1$
- $\mathrm{k}_2 = \sin\left(-\frac{2\pi}{3}\right)$
- $\mathrm{k}_3 = \frac{1}{2}\left[\cos\left(-\frac{2\pi}{5}\right) + \cos\left(-\frac{4\pi}{5}\right)\right] - 1$
- $\mathrm{k}_4 = \frac{1}{2}\left[\cos\left(-\frac{2\pi}{5}\right) - \cos\left(-\frac{4\pi}{5}\right)\right]$
- $\mathrm{k}_5 = \sin\left(-\frac{2\pi}{5}\right) + \sin\left(-\frac{4\pi}{5}\right)$
- $\mathrm{k}_6 = \sin\left(-\frac{4\pi}{5}\right)$
- $\mathrm{k}_7 = \sin\left(-\frac{2\pi}{5}\right) - \sin\left(-\frac{4\pi}{5}\right)$

The multiplications derived from the imaginary elements of $\left[\hat{C}_{5\times 3}\right]$ are not needed for real input vectors. Hence, it is sufficient to consider

$$\left[\hat{C}_{5\times 3}\right] = \begin{bmatrix} \mathrm{c}_8 & \mathrm{c}_8\mathrm{k}_1 & U \\ \mathrm{c}_8\mathrm{k}_3 & \mathrm{c}_8\mathrm{k}_1\mathrm{k}_3 & U \\ \mathrm{c}_8\mathrm{k}_4 & \mathrm{c}_8\mathrm{k}_1\mathrm{k}_4 & U \\ U & U & -\mathrm{c}_8\mathrm{k}_2\mathrm{k}_5 \\ U & U & -\mathrm{c}_8\mathrm{k}_2\mathrm{k}_6 \\ U & U & -\mathrm{c}_8\mathrm{k}_2\mathrm{k}_7 \end{bmatrix} = \begin{bmatrix} \mathrm{C}_0 & \mathrm{C}_1 & U \\ \mathrm{C}_2 & \mathrm{C}_3 & U \\ \mathrm{C}_4 & \mathrm{C}_5 & U \\ U & U & \mathrm{C}_6 \\ U & U & \mathrm{C}_7 \\ U & U & \mathrm{C}_8 \end{bmatrix}$$

The matrix $[M]$ can consequently be obtained as:

$$[M] = \begin{bmatrix} \mathrm{C}_0 & \mathrm{C}_1 & U \\ \mathrm{C}_2 & \mathrm{C}_3 & U \\ \mathrm{C}_4 & \mathrm{C}_5 & U \\ U & U & \mathrm{C}_6 \\ U & U & \mathrm{C}_7 \\ U & U & \mathrm{C}_8 \end{bmatrix} \circ \begin{bmatrix} a_{13} & a_4 & U \\ a_7 & a_3 & U \\ a_9 & a_5 & U \\ U & U & a_{10} \\ U & U & a_{12} \\ U & U & a_{11} \end{bmatrix}$$

$$= \begin{bmatrix} \mathrm{C}_0 a_{13} & \mathrm{C}_1 a_4 & U \\ \mathrm{C}_2 a_7 & \mathrm{C}_3 a_3 & U \\ \mathrm{C}_4 a_9 & \mathrm{C}_5 a_5 & U \\ U & U & \mathrm{C}_6 a_{10} \\ U & U & \mathrm{C}_7 a_{12} \\ U & U & \mathrm{C}_8 a_{11} \end{bmatrix} = \begin{bmatrix} M_0 & M_3 & U \\ M_1 & M_4 & U \\ M_2 & M_5 & U \\ U & U & M_6 \\ U & U & M_7 \\ U & U & M_8 \end{bmatrix}$$

Table 2.9: *WFTA for DCT5 ($N = 8$): Multiplications*

Op. Name	*Operation*	*Op. Name*	*Operation*	*Op. Name*	*Operation*
M_0	$\mathrm{C}_0 a_{13}$	M_3	$\mathrm{C}_1 a_4$	M_6	$\mathrm{C}_6 a_{10}$
M_1	$\mathrm{C}_2 a_7$	M_4	$\mathrm{C}_3 a_3$	M_7	$\mathrm{C}_7 a_{12}$
M_2	$\mathrm{C}_4 a_9$	M_5	$\mathrm{C}_5 a_5$	M_8	$\mathrm{C}_8 a_{11}$

Post-Additions

The additions related to this step are obtained by performing the matrix products

$$[Z] = [S_3] \left([S_5] [M] \right)^{\mathrm{T}}$$

Since

$$[S_5] = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & -1 & 0 \\ 1 & 1 & -1 & 0 & 1 & 1 \\ 1 & 1 & -1 & 0 & -1 & -1 \\ 1 & 1 & 1 & -1 & 1 & 0 \end{bmatrix} \quad \text{and} \quad [S_3] = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 1 & -1 \end{bmatrix}$$

the columns of $[Z]$ are

$$Z(:,0) = \begin{bmatrix} M_0 \\ M_0 + M_3 \\ M_0 + M_3 \end{bmatrix}$$

$$Z(:,1) = \begin{bmatrix} M_0 + M_1 + M_2 \\ M_0 + M_1 + M_2 + M_3 + M_4 + M_5 + M_6 - M_7 \\ M_0 + M_1 + M_2 + M_3 + M_4 + M_5 - M_6 + M_7 \end{bmatrix}$$

$$Z(:,2) = \begin{bmatrix} M_0 + M_1 - M_2 \\ M_0 + M_1 - M_2 + M_3 + M_4 - M_5 + M_7 + M_8 \\ M_0 + M_1 - M_2 + M_3 + M_4 - M_5 - M_7 - M_8 \end{bmatrix}$$
$$Z(:,3) = \begin{bmatrix} M_0 + M_1 - M_2 \\ M_0 + M_1 - M_2 + M_3 + M_4 - M_5 - M_7 - M_8 \\ M_0 + M_1 - M_2 + M_3 + M_4 - M_5 + M_7 + M_8 \end{bmatrix}$$
$$Z(:,4) = \begin{bmatrix} M_0 + M_1 + M_2 \\ M_0 + M_1 + M_2 + M_3 + M_4 + M_5 - M_6 + M_7 \\ M_0 + M_1 + M_2 + M_3 + M_4 + M_5 + M_6 - M_7 \end{bmatrix}.$$

The additions described in table 2.10 can be performed to compute all the elements of the matrix $[Z]$.

Table 2.10: *WFTA for DCT5 ($N = 8$): Post-Additions*

Op. Name	*Operation*	*Op. Name*	*Operation*	*Op. Name*	*Operation*
a_{14}	$M_0 + M_1$	a_{19}	$a_{17} - M_5$	a_{24}	$a_{23} + a_{20}$
a_{15}	$a_{14} + M_2$	a_{20}	$M_6 - M_7$	a_{25}	$a_{23} - a_{20}$
a_{16}	$a_{14} - M_2$	a_{21}	$M_7 + M_8$	a_{26}	$a_{16} + a_{19}$
a_{17}	$M_3 + M_4$	a_{22}	$M_0 + M_3$	a_{27}	$a_{26} + a_{21}$
a_{18}	$a_{17} + M_5$	a_{23}	$a_{15} + a_{18}$	a_{28}	$a_{26} - a_{21}$

Hence, the matrix $[Z]$ can be expressed as

$$[Z] = \begin{bmatrix} M_0 & a_{15} & a_{16} & a_{16} & a_{15} \\ a_{22} & a_{24} & a_{27} & a_{28} & a_{25} \\ a_{22} & a_{25} & a_{28} & a_{27} & a_{24} \end{bmatrix}$$

It should be highlighted that only the first 8 elements aligned along the extended diagonal have to be considered. In fact:

$$[Z] = \mathbf{Re}\left(\begin{bmatrix} \hat{Y}_{\mathrm{F}0} & \hat{Y}_{\mathrm{F}6} & U & \hat{Y}_{\mathrm{F}3} & U \\ U & \hat{Y}_{\mathrm{F}1} & \hat{Y}_{\mathrm{F}7} & U & \hat{Y}_{\mathrm{F}4} \\ \hat{Y}_{\mathrm{F}5} & U & \hat{Y}_{\mathrm{F}2} & U & U \end{bmatrix} \right)$$

where the components labeled as "U"are not needed for the computation of the DCT5 even though they are equal to the elements that are instead significant.

Post-Normalization

The multiplication

$$M_{\mathrm{n}2} = \mathrm{C}_{\mathrm{norm}} M_0 = \frac{1}{\sqrt{2}} M_0$$

is performed.

Final Algorithm

The final version of the algorithm is reported in table 2.11. It should be highlighted that the computational cost of the algorithm is reduced to 11 multiplications and 29 additions.

Table 2.11: *WFTA applied to the DCT5 for $N = 8$*

Op. Name	*Operation*	*Op. Name*	*Operation*	*Op. Name*	*Operation*
M_{n1}	$\mathrm{C}_{\mathrm{norm}} x_0$	M_1	$\mathrm{C}_2 a_7$	a_{22}	$M_0 + M_3$
a_0	$x_1 + x_4$	M_2	$\mathrm{C}_4 a_9$	a_{23}	$a_{15} + a_{18}$
a_1	$x_2 + x_7$	M_3	$\mathrm{C}_1 a_4$	a_{24}	$a_{23} + a_{20}$
a_2	$x_3 + x_6$	M_4	$\mathrm{C}_3 a_3$	a_{25}	$a_{23} - a_{20}$
a_3	$a_0 + a_1$	M_5	$\mathrm{C}_5 a_5$	a_{26}	$a_{16} + a_{19}$
a_4	$a_3 + x_5$	M_6	$\mathrm{C}_6 a_{10}$	a_{27}	$a_{26} + a_{21}$
a_5	$a_1 - a_0$	M_7	$\mathrm{C}_7 a_{12}$	a_{28}	$a_{26} - a_{21}$
a_6	$a_4 + a_2$	M_8	$\mathrm{C}_8 a_{11}$	M_{n2}	$\mathrm{C}_{\mathrm{norm}} M_0$
a_7	$a_3 + a_2$	a_{14}	$M_0 + M_1$	Y_0	M_{n2}
a_8	$x_3 - x_6$	a_{15}	$a_{14} + M_2$	Y_1	a_{24}
a_9	$a_5 + a_8$	a_{16}	$a_{14} - M_2$	Y_2	a_{28}
a_{10}	$x_7 - x_2$	a_{17}	$M_3 + M_4$	Y_3	a_{16}
a_{11}	$x_1 - x_4$	a_{18}	$a_{17} + M_5$	Y_4	a_{25}
a_{12}	$a_{10} + a_{11}$	a_{19}	$a_{17} - M_5$	Y_5	a_{22}
a_{13}	$a_6 + M_{\mathrm{n1}}$	a_{20}	$M_6 - M_7$	Y_6	a_{15}
M_0	$\mathrm{C}_0 a_{13}$	a_{21}	$M_7 + M_8$	Y_7	a_{27}

2.3 The Prime Factor Algorithm (PFA)

The indexing scheme proposed for the WFTA is a way of organizing a linear array of $N_{\mathrm{DFT}} = N1 \times N2$ numbers into an $N1$ by $N2$ array, but in such a way that a one-dimensional Fourier transform can be converted into a true two-dimensional Fourier transform. This two-dimensional Fourier transform can also be solved by computing $N1$ DFTs of length $N2$ and $N2$ DFTs of length $N1$. This is the approach followed in the Prime Factor Algorithm ([4], [5], [6] and [7]).

2.3.1 The steps of the PFA

Supposing that the length N_{DFT} of the DFT can be expressed as a product of two numbers ($N1$,$N2$) that are coprime, the PFA can be adopted to solve the DFT. In particular, the DFT can be computed according to the following steps:

1. index transform of the input sequence
2. computation of $N1$ DFTs of length $N2$
3. computation of $N2$ DFTs of length $N1$

4. index transform of the output sequence

In the following, each of these steps will be described in detail.

Input Mapping

The input mapping is obtained by following the same procedure proposed for the Winograd large fast Fourier transform. In particular, a permutation is applied to the input vector $\boldsymbol{l}$. Therefore, the vector $\hat{\boldsymbol{l}}$ is obtained as

$$\hat{\boldsymbol{l}} = [P_i]\,\boldsymbol{l}$$

where $[P_i]$ is the input permutation matrix.

The matrix $[P_i]$ can be constructed according to the following steps:

1. A couple (n_1,n_2) is assigned to each element of $\hat{\boldsymbol{l}}$ with $0 \leq n_1 \leq (N1-1)$ and $0 \leq n_2 \leq (N2-1)$. More in detail, the couple (n_1,n_2) is obtained by progressively increasing n_2 and n_1. This is better explained by the following example. Consider, for instance, $N_{\mathrm{DFT}} = 15 = 3 \times 5 = N1 \times N2$. The assignments shown in table 2.12 are obtained.
2. The corresponding index of the vector $\boldsymbol{l}$ is derived by applying the equation:

$$i_{\boldsymbol{l}} = (n_1 \times N2 + n_2 \times N1) \bmod N_{\mathrm{DFT}}$$

The indexes obtained for the case presented above are reported in table 2.12.

Table 2.12: *Permuted indexes calculation for* $N1 = 3$ *and* $N2 = 5$

$i_{\hat{\boldsymbol{l}}}$	(n_1,n_2)	$i_{\boldsymbol{l}}$
0	(0,0)	$(0 \times 5 + 0 \times 3) \bmod 15 = 0$
1	(0,1)	$(0 \times 5 + 1 \times 3) \bmod 15 = 3$
2	(0,2)	$(0 \times 5 + 2 \times 3) \bmod 15 = 6$
3	(0,3)	$(0 \times 5 + 3 \times 3) \bmod 15 = 9$
4	(0,4)	$(0 \times 5 + 4 \times 3) \bmod 15 = 12$
5	(1,0)	$(1 \times 5 + 0 \times 3) \bmod 15 = 5$
6	(1,1)	$(1 \times 5 + 1 \times 3) \bmod 15 = 8$
7	(1,2)	$(1 \times 5 + 2 \times 3) \bmod 15 = 11$
8	(1,3)	$(1 \times 5 + 3 \times 3) \bmod 15 = 14$
9	(1,4)	$(1 \times 5 + 4 \times 3) \bmod 15 = 2$
10	(2,0)	$(2 \times 5 + 0 \times 3) \bmod 15 = 10$
11	(2,1)	$(2 \times 5 + 1 \times 3) \bmod 15 = 13$
12	(2,2)	$(2 \times 5 + 2 \times 3) \bmod 15 = 1$
13	(2,3)	$(2 \times 5 + 3 \times 3) \bmod 15 = 4$
14	(2,4)	$(2 \times 5 + 4 \times 3) \bmod 15 = 7$

3. The permutation matrix is built by setting the elements in position $(i_{\hat{\boldsymbol{l}}}, i_{\boldsymbol{l}})$ equal to one. Thus, for the case presented above, the ones of the permutation matrix are in positions: (0,0), (1,3), (2,6), (3,9), (4,12), (5,5), (6,8), (7,11), (8,14), (9,2), (10,10), (11,13), (12,1), (13,4) and (14,7).

The vector $\hat{\boldsymbol{l}}$ can therefore be obtained by performing the matrix-vector multiplication:

$$\hat{\boldsymbol{l}} = [P_i]\,\boldsymbol{l} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} l_0 \\ l_1 \\ l_2 \\ l_3 \\ l_4 \\ l_5 \\ l_6 \\ l_7 \\ l_8 \\ l_9 \\ l_{10} \\ l_{11} \\ l_{12} \\ l_{13} \\ l_{14} \end{bmatrix} = \begin{bmatrix} l_0 \\ l_3 \\ l_6 \\ l_9 \\ l_{12} \\ l_5 \\ l_8 \\ l_{11} \\ l_{14} \\ l_2 \\ l_{10} \\ l_{13} \\ l_1 \\ l_4 \\ l_7 \end{bmatrix}$$

Computation of $N1$ DFTs of length $N2$

This is the first stage of the PFA. The elements of the vector $\hat{\boldsymbol{l}}$ are used as inputs for the $N1$ DFTs of length $N2$ (figure 2.1 for $N1 = 3$ and $N2 = 5$) .

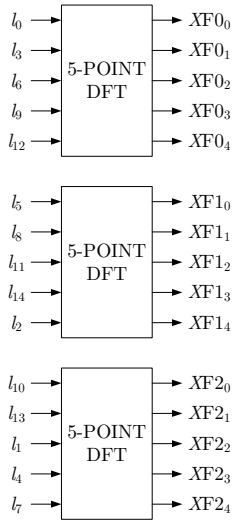


Figure 2.1: *First stage of the PFA for* $N1 = 3$ *and* $N2 = 5$

Computation of $N2$ DFTs of length $N1$

This is the last stage of the PFA. The elements XFj_i are used as inputs for the i-th DFT of length $N1$ (figure 2.2 for $N1 = 3$ and $N2 = 5$).

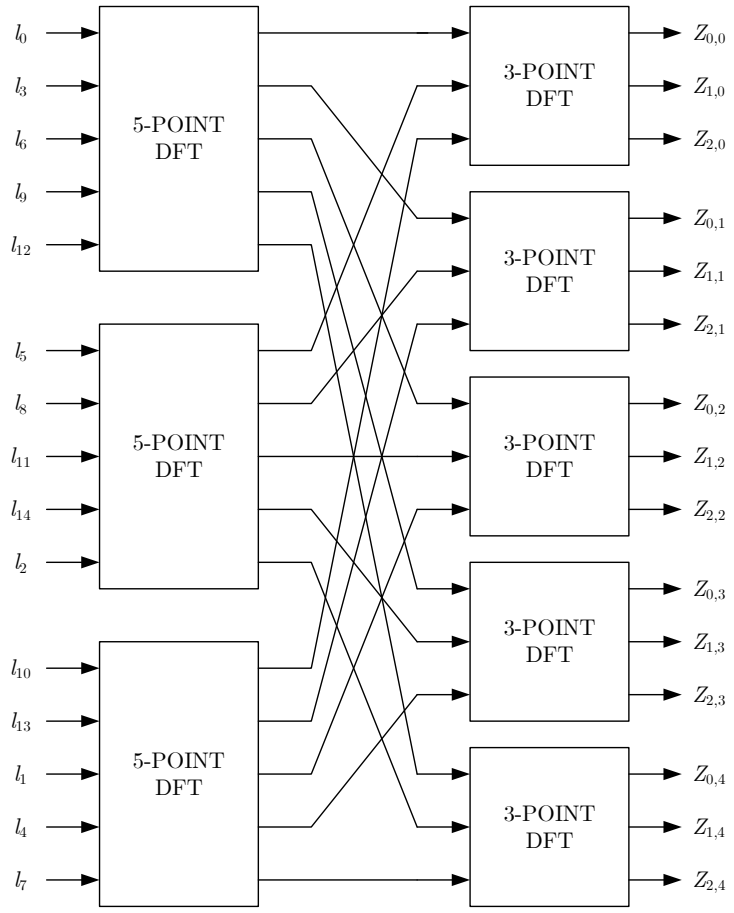


Figure 2.2: *First and second stage of the PFA for $N1 = 3$ and $N2 = 5$*

Output mapping

The outputs of each DFT of length $N1$ represent a column of the matrix $[Z]$. The elements of the vector $\boldsymbol{Y_F}$ are stored in the matrix $[Z]$ starting in the upper left corner and listing the components down the "extended diagonal". Since the number of rows and the number of columns are relatively prime, the extended diagonal passes through every element of the array. For the case considered in the previous paragraphs (figure 2.3), it can be written:

$$[Z] = \begin{bmatrix} Z_{0,0} & Z_{0,1} & Z_{0,2} & Z_{0,3} & Z_{0,4} \\ Z_{1,0} & Z_{1,1} & Z_{1,2} & Z_{1,3} & Z_{1,4} \\ Z_{2,0} & Z_{2,1} & Z_{2,2} & Z_{2,3} & Z_{2,4} \end{bmatrix} = \begin{bmatrix} Y_{\mathrm{F}0} & Y_{\mathrm{F}6} & Y_{\mathrm{F}12} & Y_{\mathrm{F}3} & Y_{\mathrm{F}9} \\ Y_{\mathrm{F}10} & Y_{\mathrm{F}1} & Y_{\mathrm{F}7} & Y_{\mathrm{F}13} & Y_{\mathrm{F}4} \\ Y_{\mathrm{F}5} & Y_{\mathrm{F}11} & Y_{\mathrm{F}2} & Y_{\mathrm{F}8} & Y_{\mathrm{F}14} \end{bmatrix}$$

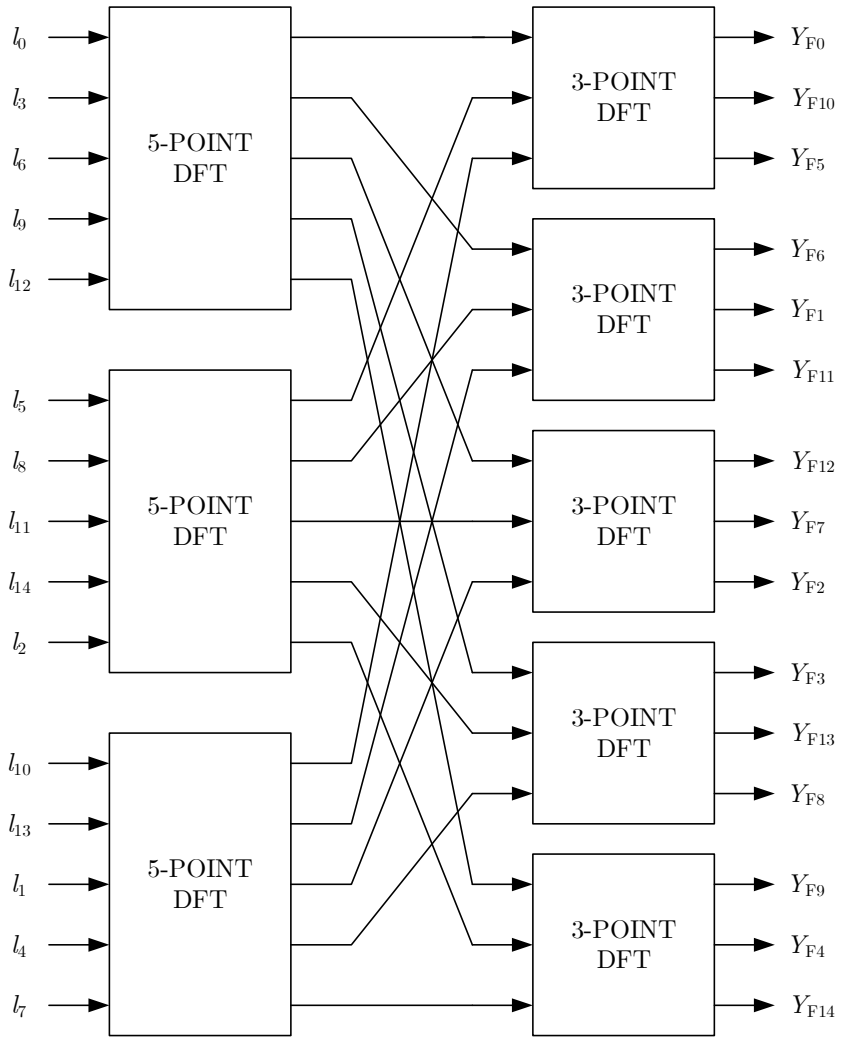


Figure 2.3: *PFA for* $N1 = 3$ *and* $N2 = 5$

2.3.2 PFA for DCT5 ($N = 8$)

A DCT5 of length 8 can be mapped into a DFT of length 15 as described in section 2.1. In particular, the DCT5 equation can be written as

$$Y_n = \mathrm{c}_8 T_n \mathbf{Re}(Y_{\mathrm{F}n}) \qquad \text{for } n = 0,1,...,7$$

where

- $\mathrm{c}_8 = \frac{2}{\sqrt{15}}$
- $\mathbf{Re}(Y_{\mathrm{F}n}) = \mathbf{Re}\left(\sum_{k=0}^{14} l_k \mathrm{e}^{-\mathrm{j}nk\frac{2\pi}{15}}\right)$

and

$$\boldsymbol{l} = \begin{bmatrix} l_0 \\ l_1 \\ l_2 \\ l_3 \\ l_4 \\ l_5 \\ l_6 \\ l_7 \\ l_8 \\ l_9 \\ l_{10} \\ l_{11} \\ l_{12} \\ l_{13} \\ l_{14} \end{bmatrix} = \begin{bmatrix} \hat{x}_0 \\ 0 \\ \hat{x}_2 \\ 0 \\ \hat{x}_4 \\ 0 \\ \hat{x}_6 \\ 0 \\ \hat{x}_7 \\ 0 \\ \hat{x}_5 \\ 0 \\ \hat{x}_3 \\ 0 \\ \hat{x}_1 \end{bmatrix} = \begin{bmatrix} x_0 T_k|_{k=0} \\ 0 \\ x_2 \\ 0 \\ x_4 \\ 0 \\ x_6 \\ 0 \\ x_7 \\ 0 \\ x_5 \\ 0 \\ x_3 \\ 0 \\ x_1 \end{bmatrix} = \begin{bmatrix} \frac{x_0}{\sqrt{2}} \\ 0 \\ x_2 \\ 0 \\ x_4 \\ 0 \\ x_6 \\ 0 \\ x_7 \\ 0 \\ x_5 \\ 0 \\ x_3 \\ 0 \\ x_1 \end{bmatrix}$$

For the sake of simplicity, the normalization factors can be initially neglected. Hence, we can neglect c_8 and consider:

$$T_n = 1 \qquad T_k = 1 \qquad \forall n,k$$

As a consequence, the vector $\boldsymbol{l}$ can be redefined as

$$\begin{aligned} \boldsymbol{l} &= \begin{bmatrix} l_0 & l_1 & l_2 & l_3 & l_4 & l_5 & l_6 & l_7 & l_8 & l_9 & l_{10} & l_{11} & l_{12} & l_{13} & l_{14} \end{bmatrix}^{\mathrm{T}} \\ &= \begin{bmatrix} x_0 & 0 & x_2 & 0 & x_4 & 0 & x_6 & 0 & x_7 & 0 & x_5 & 0 & x_3 & 0 & x_1 \end{bmatrix}^{\mathrm{T}} \end{aligned}$$

The input permutation matrix is therefore applied to this vector in order to obtain the vector $\hat{\boldsymbol{l}}$. Thus, we have

$$\hat{\boldsymbol{l}} = [P_i]\,\boldsymbol{l} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} l_0 \\ l_1 \\ l_2 \\ l_3 \\ l_4 \\ l_5 \\ l_6 \\ l_7 \\ l_8 \\ l_9 \\ l_{10} \\ l_{11} \\ l_{12} \\ l_{13} \\ l_{14} \end{bmatrix} = \begin{bmatrix} l_0 \\ l_3 \\ l_6 \\ l_9 \\ l_{12} \\ l_5 \\ l_8 \\ l_{11} \\ l_{14} \\ l_2 \\ l_{10} \\ l_{13} \\ l_1 \\ l_4 \\ l_7 \end{bmatrix} = \begin{bmatrix} x_0 \\ 0 \\ x_6 \\ 0 \\ x_3 \\ 0 \\ x_7 \\ 0 \\ x_1 \\ x_2 \\ x_5 \\ 0 \\ 0 \\ x_4 \\ 0 \end{bmatrix}$$

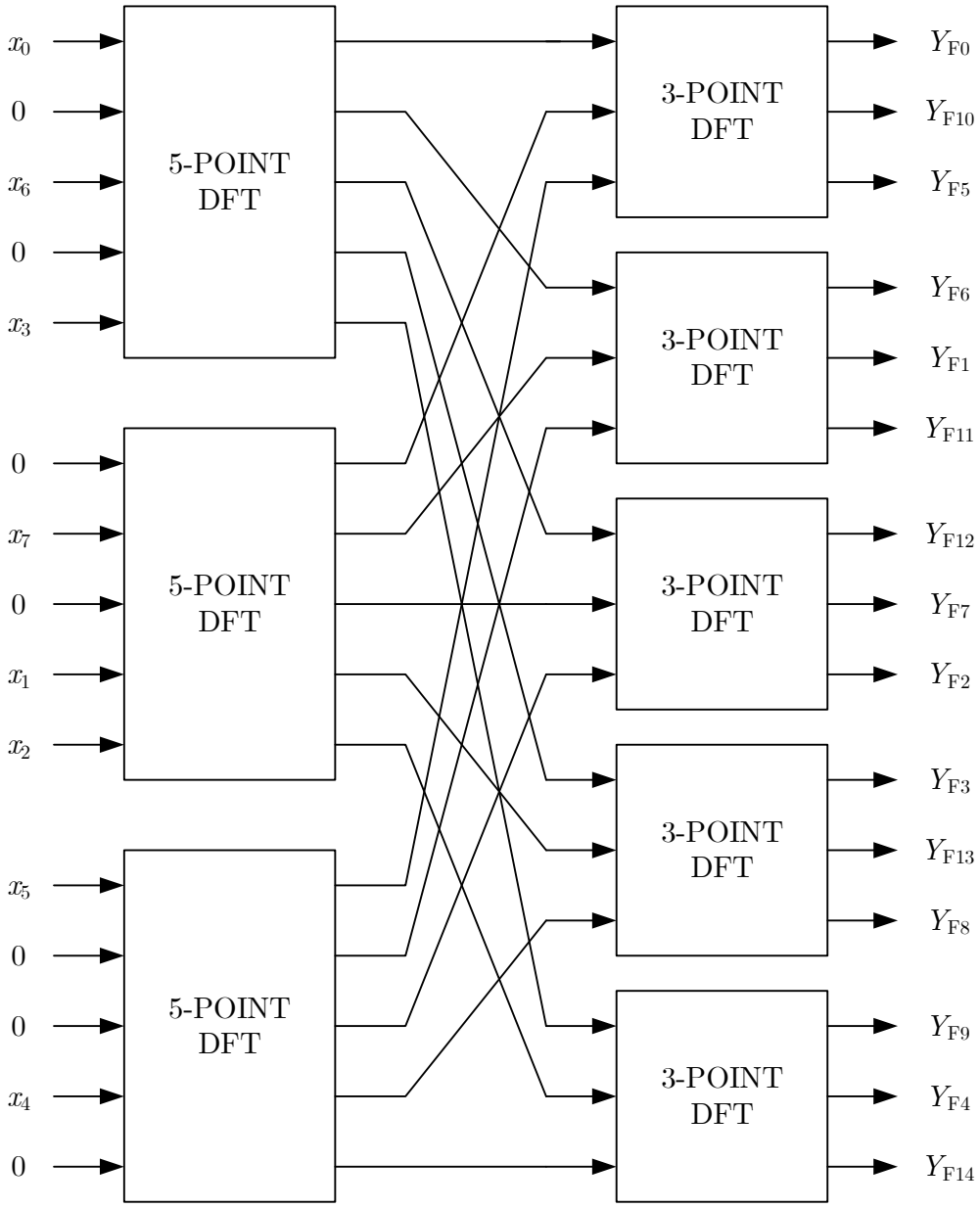


Figure 2.4: *PFA for DCT5 with* $N = 8$ *(*$N1 = 3$*,* $N2 = 5$*)*

The vector $\hat{\boldsymbol{l}}$ is then used as input for the first stage of the PFA as depicted in figure 2.4. The real part of the elements $Y_{\mathrm{F}n}$ represented in this figure are the outputs generated by the DCT5. More in detail, only the first eight elements should be considered. Nevertheless, it should be remembered that the real part of the DFT of an odd-length real sequence is a palindromic sequence except for the first element. Hence, it can be written

$$\mathbf{Re}(Y_{\mathrm{F}8}) = \mathbf{Re}(Y_{\mathrm{F}7}) \qquad \mathbf{Re}(Y_{\mathrm{F}9}) = \mathbf{Re}(Y_{\mathrm{F}6}) \qquad \dots \qquad \mathbf{Re}(Y_{\mathrm{F}14}) = \mathbf{Re}(Y_{\mathrm{F}1})$$

Several simplifications can be made to the scheme presented in figure 2.4. First of all, the Signal Flow Graph (SFG) of the 3-point and 5-point DFTs can be analyzed. These are respectively reported in figures 2.5 and 2.6. Moreover, the values of the constants C_i are presented in table 2.13.

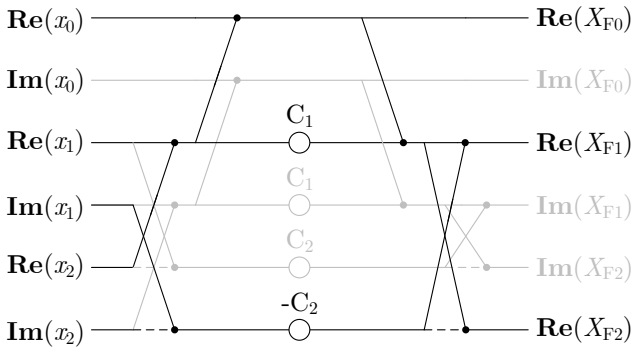


Figure 2.5: *Simplified 3-point DFT SFG (simplifications made for $N1 = 3$, $N2 = 5$)*

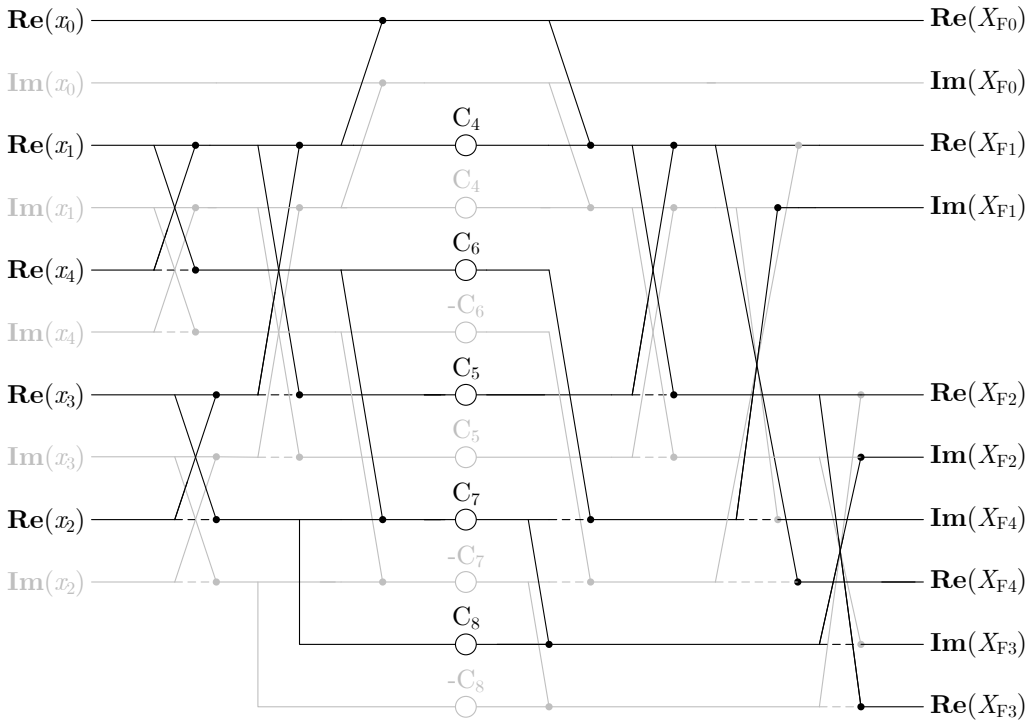


Figure 2.6: *Simplified 5-point DFT SFG (simplifications made for $N1 = 3$, $N2 = 5$)*

Concerning the computation of the DCT5 (in particular, when it is chosen $N1 = 3$ and $N2 = 5$), the gray lines in figure 2.5 can be neglected since only the real part of the elements $Y_{\mathrm{F}n}$ is needed. Similarly, the gray lines in figure 2.6 can be neglected since the inputs of the DCT5 are real.

Table 2.13: *Values of the constants C_i*

Constant	*Value*
C_1	$\cos\left(-\frac{2\pi}{3}\right) - 1$
C_2	$\sin\left(-\frac{2\pi}{3}\right)$
C_4	$\left[\cos\left(-\frac{2\pi}{5}\right) + \cos(-\frac{4\pi}{5})\right]/2 - 1$
C_5	$\left[\cos(-\frac{2\pi}{5}) - \cos(-\frac{4\pi}{5})\right]/2$

Constant	*Value*
C_6	$\sin(-\frac{2\pi}{5}) + \sin(-\frac{4\pi}{5})$
C_7	$\sin(-\frac{4\pi}{5})$
C_8	$\sin(-\frac{2\pi}{5}) - \sin(-\frac{4\pi}{5})$

The 3-point and 5-point DFT SFGs can be used to build the SFG of the 8-point DCT5 as depicted in figure 2.7.

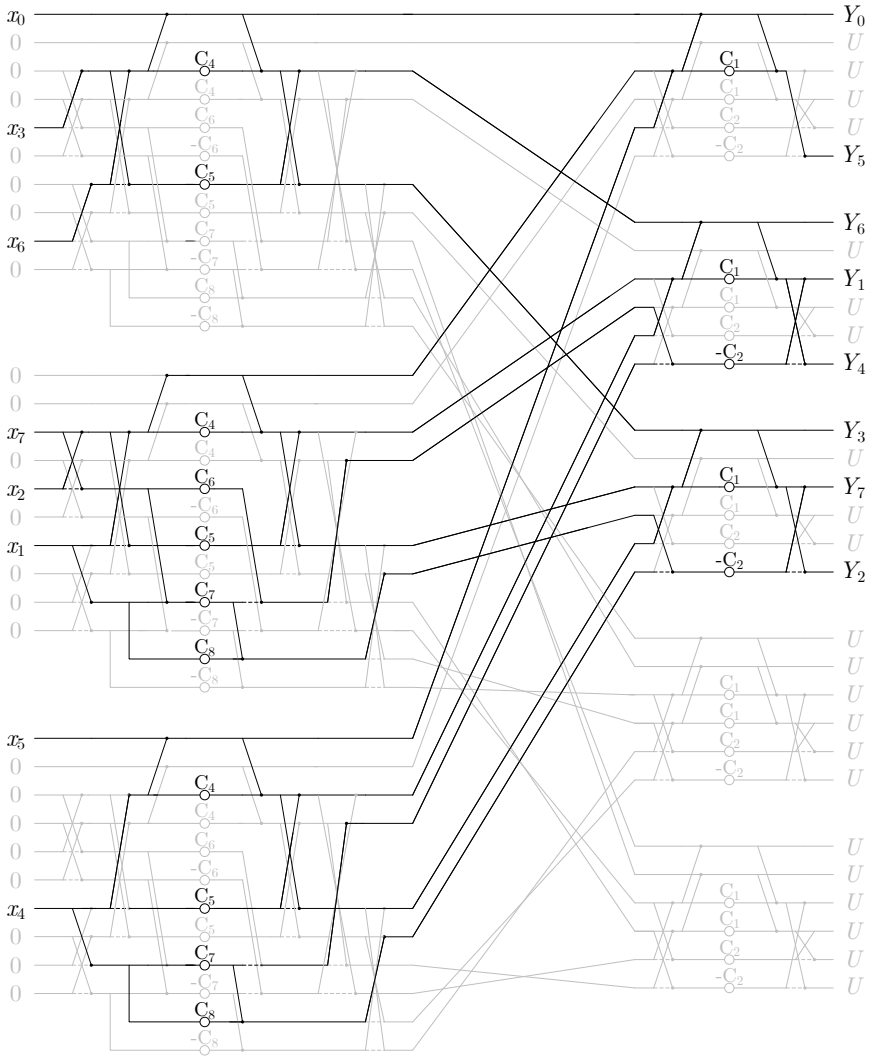


Figure 2.7: *Non-normalized 8-point DCT5 SFG ($N1 = 3$, $N2 = 5$)*

All the paths coming from null inputs or connected to unused outputs are drawn using gray lines. Moreover, it should be highlighted that the palindromicity of the sequence $\mathbf{Re}(\{Y_{F1},Y_{F2},...,Y_{F13},Y_{F14}\})$ is exploited in the derivation of the SFG presented in figure 2.7.

Finally, the normalization factors can be introduced so that the normalized version of the DCT5 is obtained. The resulting SFG is presented in figure 2.8.

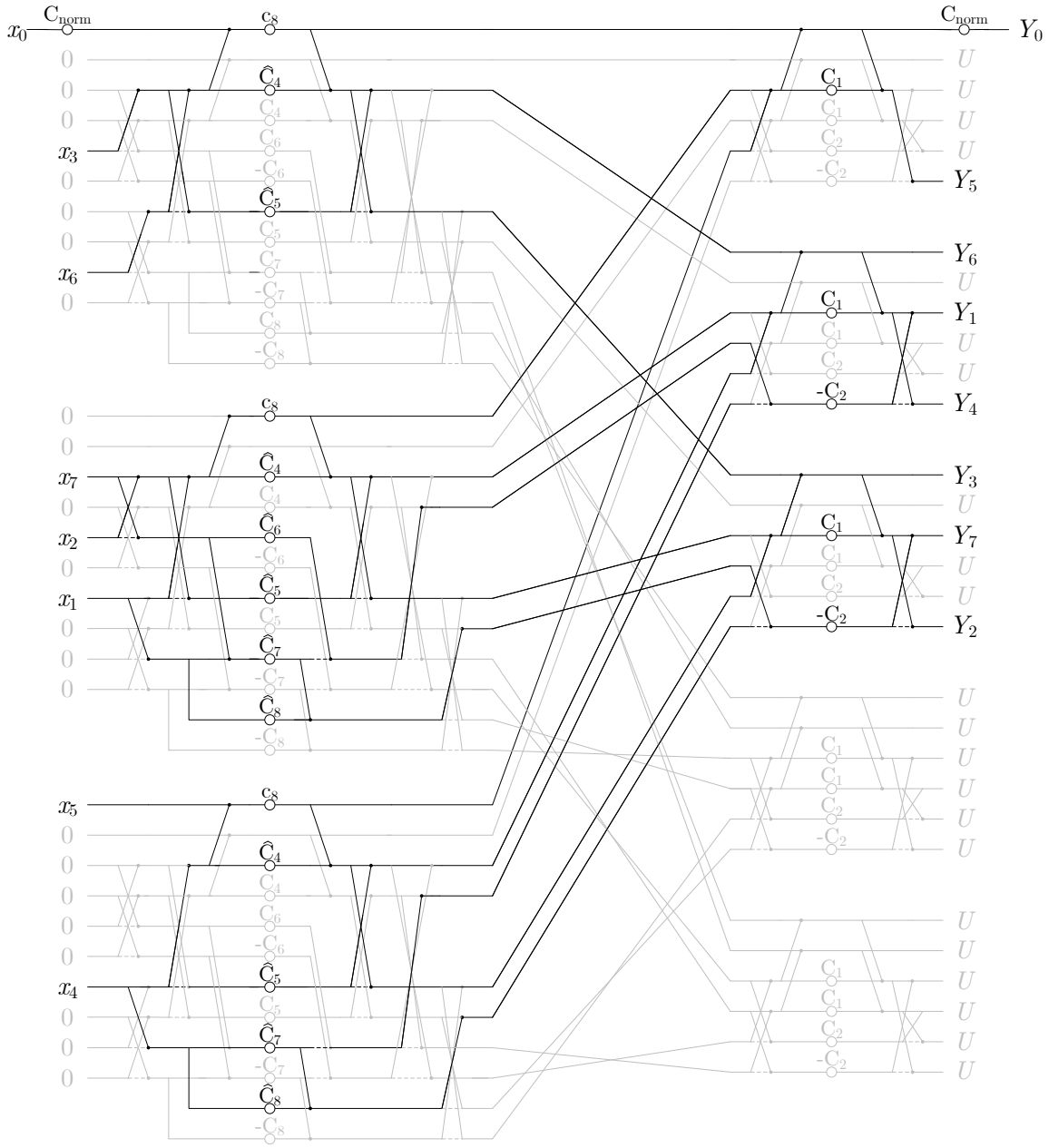


Figure 2.8: *Normalized 8-point DCT5 SFG ($N1 = 3$, $N2 = 5$)*

The constants $\hat{\mathrm{C}}_i$ reported in figure 2.8 are obtained as

$$\hat{\mathrm{C}}_i = \mathrm{c}_8 \mathrm{C}_i \qquad \mathrm{c}_8 = \frac{2}{\sqrt{15}}$$

while the constant $\mathrm{C}_{\mathrm{norm}}$ is equal to $\frac{1}{\sqrt{2}}$.

It should also be highlighted that the algorithm represented in figure 2.8 only requires 21 multiplications and 36 sums. The list of the needed operations is reported in table 2.14.

Table 2.14: *PFA applied to the DCT5 for $N = 8$ ($N1 = 3$, $N2 = 5$)*

Op. Name	*Operation*
M_{n1}	$\mathrm{C}_{\mathrm{norm}} x_0$
a_0	$x_3 + x_6$
a_1	$M_{\mathrm{n1}} + a_0$
a_2	$x_3 - x_6$
M_0	$\mathrm{c}_8 a_1$
M_1	$\hat{\mathrm{C}}_4 a_0$
M_2	$\hat{\mathrm{C}}_5 a_2$
a_3	$M_0 + M_1$
a_4	$a_3 + M_2$
a_5	$a_3 - M_2$
a_6	$x_2 + x_7$
a_7	$x_7 - x_2$
a_8	$a_6 + x_1$
a_9	$a_6 - x_1$
a_{10}	$a_7 + x_1$
M_3	$\mathrm{c}_8 a_8$
M_4	$\hat{\mathrm{C}}_4 a_8$
M_5	$\hat{\mathrm{C}}_6 a_7$
M_6	$\hat{\mathrm{C}}_5 a_9$
M_7	$\hat{\mathrm{C}}_7 a_{10}$
M_8	$\hat{\mathrm{C}}_8 x_1$
a_{11}	$M_3 + M_4$
a_{12}	$a_{11} + M_6$
a_{13}	$a_{11} - M_6$
a_{14}	$M_5 - M_7$
a_{15}	$M_7 + M_8$
a_{16}	$x_4 + x_5$
M_9	$\mathrm{c}_8 a_{16}$
M_{10}	$\hat{\mathrm{C}}_4 x_4$
M_{11}	$-\hat{\mathrm{C}}_5 x_4$
M_{12}	$\hat{\mathrm{C}}_7 x_4$
M_{13}	$\hat{\mathrm{C}}_8 x_4$
a_{17}	$M_9 + M_{10}$
a_{18}	$a_{17} + M_{11}$
a_{19}	$a_{17} - M_{11}$
a_{20}	$M_{12} + M_{13}$
a_{21}	$M_3 + M_9$
a_{22}	$M_0 + a_{21}$
M_{n2}	$\mathrm{C}_{\mathrm{norm}} a_{22}$
M_{14}	$\mathrm{C}_1 a_{21}$
a_{23}	$a_{22} + M_{14}$
a_{24}	$a_{12} + a_{18}$
a_{25}	$a_4 + a_{24}$
a_{26}	$a_{14} + M_{12}$
M_{15}	$\mathrm{C}_1 a_{24}$
M_{16}	$-\mathrm{C}_2 a_{26}$
a_{27}	$a_{25} + M_{15}$
a_{28}	$a_{27} + M_{16}$
a_{29}	$a_{27} - M_{16}$
a_{30}	$a_{13} + a_{19}$
a_{31}	$a_{30} + a_5$
M_{17}	$\mathrm{C}_1 a_{30}$
a_{32}	$a_{15} - a_{20}$
M_{18}	$-\mathrm{C}_2 a_{32}$
a_{33}	$M_{17} + a_{31}$
a_{34}	$a_{33} + M_{18}$
a_{35}	$a_{33} - M_{18}$
Y_0	M_{n2}
Y_1	a_{28}
Y_2	a_{35}
Y_3	a_{31}
Y_4	a_{29}
Y_5	a_{23}
Y_6	a_{25}
Y_7	a_{34}

Remarks

The same procedure presented above can be repeated by choosing $N1 = 5$ and $N2 = 3$. In this case, the scheme presented in figure 2.9 is obtained.

The SFGs presented in figures 2.10 and 2.11 can therefore be considered. By making the same simplifications discussed in the previous paragraph, the SFG presented in figure 2.12 is derived. The algorithm shown in this figure only requires 18 multiplications and 32 additions. The list of the needed operations is reported in table 2.15.

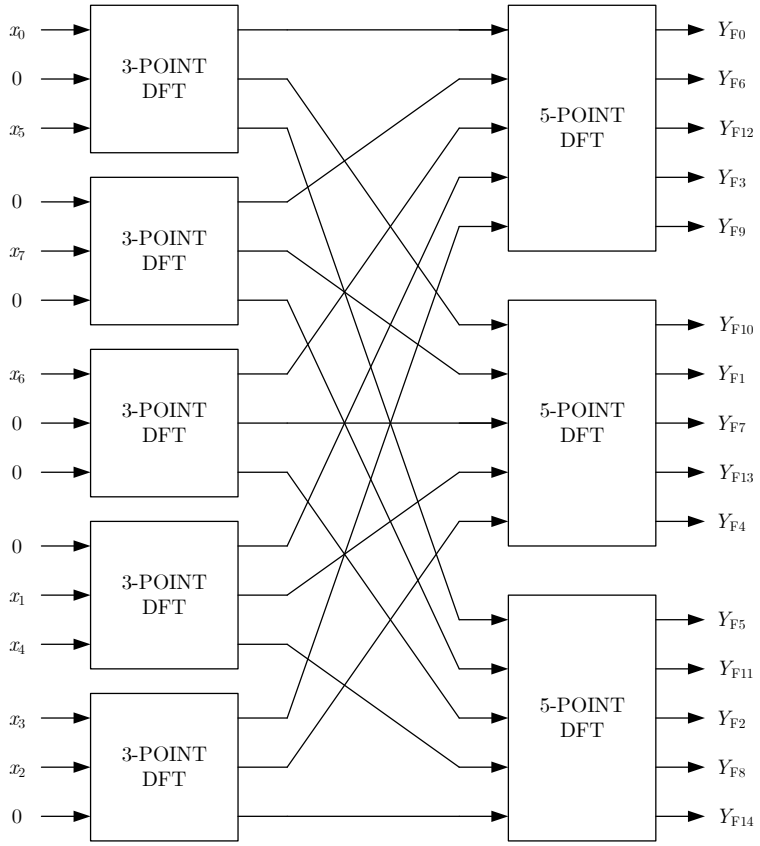


Figure 2.9: *PFA for DCT5 with $N = 8$ ($N1 = 5$, $N2 = 3$)*

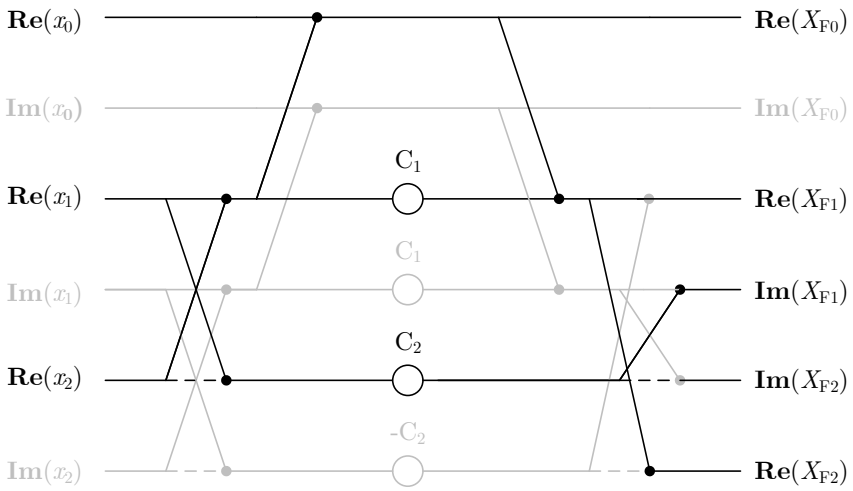


Figure 2.10: *Simplified 3-point DFT SFG (simplifications made for $N1 = 5$, $N2 = 3$)*

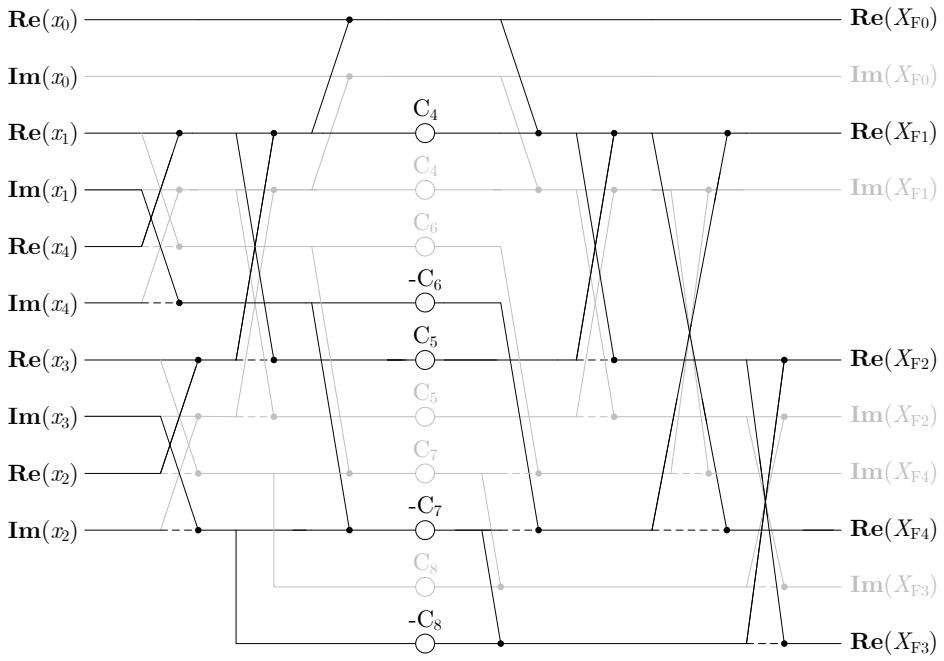


Figure 2.11: *Simplified 5-point DFT SFG (simplifications made for $N1 = 5$, $N2 = 3$)*

Table 2.15: *PFA applied to the DCT5 for $N = 8$ ($N1 = 5$, $N2 = 3$)*

Op. Name	*Operation*
M_{n1}	$\mathrm{C}_{\mathrm{norm}} x_0$
a_0	$x_5 + M_{\mathrm{n1}}$
M_0	$\mathrm{C}_1 x_5$
a_1	$M_0 + a_0$
M_1	$\mathrm{C}_1 x_7$
M_2	$\mathrm{C}_2 x_7$
a_2	$x_7 + M_1$
a_3	$x_1 + x_4$
a_4	$x_1 - x_4$
M_3	$\mathrm{C}_1 a_3$
M_4	$\mathrm{C}_2 a_4$
a_5	$M_3 + a_3$
a_6	$x_2 + x_3$
M_5	$\mathrm{C}_1 x_2$
M_6	$\mathrm{C}_2 x_2$
a_7	$M_5 + a_6$
a_8	$x_7 + a_6$

Op. Name	*Operation*
a_9	$a_3 + x_6$
a_{10}	$a_8 + a_9$
a_{11}	$a_8 - a_9$
a_{12}	$a_{10} + a_0$
M_7	$\mathrm{c}_8 a_{12}$
Y_0	$\mathrm{C}_{\mathrm{norm}} M_7$
M_8	$\hat{\mathrm{C}}_4 a_{10}$
M_9	$\hat{\mathrm{C}}_5 a_{11}$
a_{13}	$M_7 + M_8$
Y_6	$a_{13} + M_9$
Y_3	$a_{13} - M_9$
a_{16}	$a_2 + a_7$
a_{17}	$M_2 - M_6$
a_{18}	$a_5 + x_6$
a_{19}	$a_{16} + a_{18}$
a_{20}	$a_{16} - a_{18}$
a_{21}	$a_{17} + M_4$

Op. Name	*Operation*
a_{22}	$a_{19} + a_1$
Y_5	$\mathrm{c}_8 a_{22}$
M_{11}	$\hat{\mathrm{C}}_4 a_{19}$
M_{12}	$-\hat{\mathrm{C}}_6 a_{17}$
M_{13}	$\hat{\mathrm{C}}_5 a_{20}$
M_{14}	$-\hat{\mathrm{C}}_7 a_{21}$
M_{15}	$-\hat{\mathrm{C}}_8 M_4$
a_{23}	$Y_5 + M_{11}$
a_{24}	$a_{23} + M_{13}$
a_{25}	$a_{23} - M_{13}$
a_{26}	$M_{12} - M_{14}$
a_{27}	$M_{14} + M_{15}$
Y_1	$a_{24} + a_{26}$
Y_4	$a_{24} - a_{26}$
Y_7	$a_{25} + a_{27}$
Y_2	$a_{25} - a_{27}$

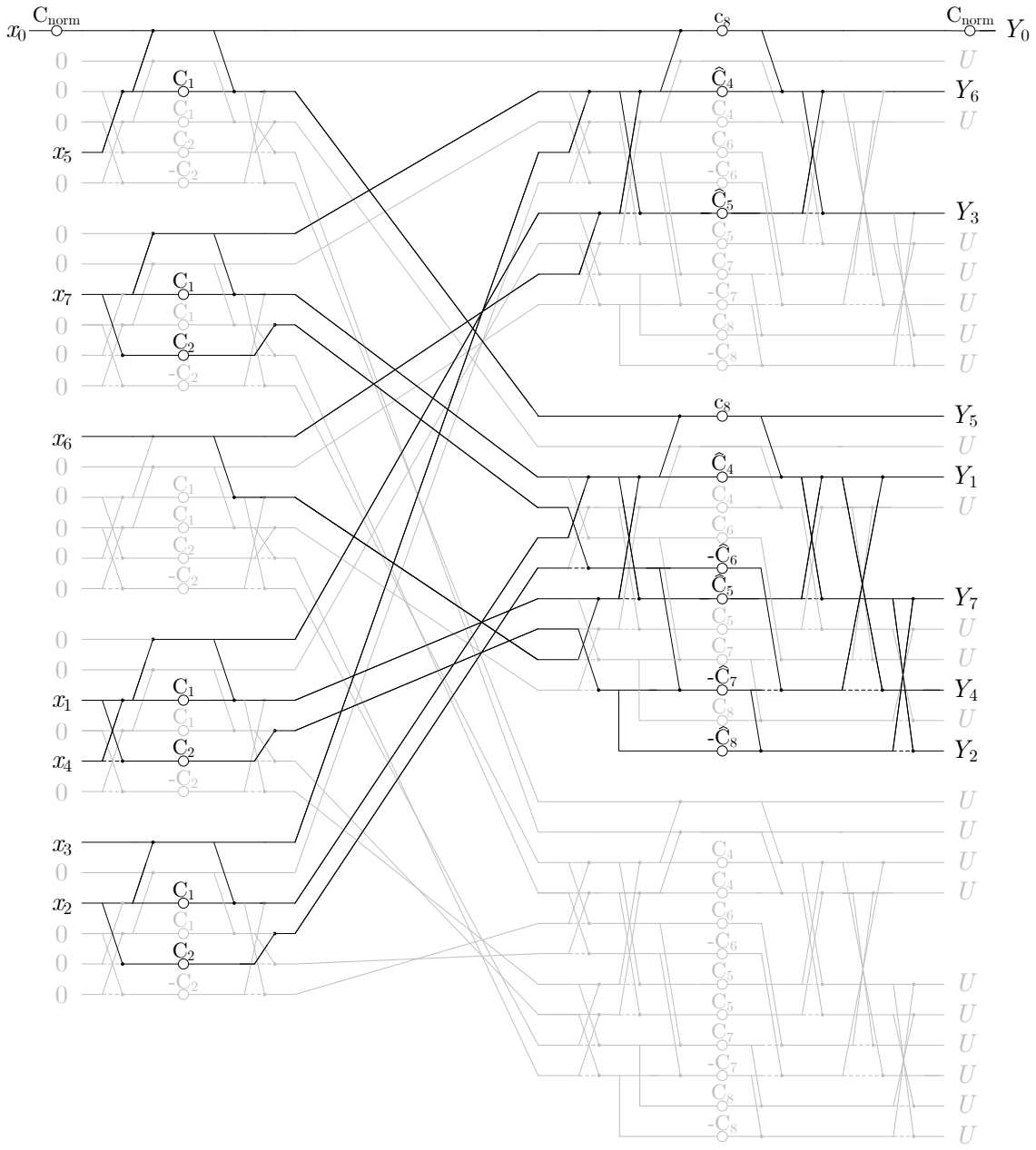


Figure 2.12: *Normalized 8-point DCT5 SFG ($N1 = 5$, $N2 = 3$)*

2.4 Bluestein's Algorithm

Bluestein's FFT algorithm (also called the *chirp z-transform algorithm*) is a fast Fourier transform algorithm, which computes the discrete Fourier transform of arbitrary sizes (including prime sizes) by reformulating the DFT as a convolution ([4], [6], [8] and [9]).

In order to analyze this algorithm, we can begin by rewriting the DFT as:

$$Y_{\mathrm{F}n} = \sum_{k=0}^{N_{\mathrm{DFT}}-1} l_k \mathrm{e}^{-\mathrm{j}nk\frac{2\pi}{N_{\mathrm{DFT}}}} = \sum_{k=0}^{N_{\mathrm{DFT}}-1} l_k W^{nk} \qquad W = \mathrm{e}^{-\mathrm{j}\frac{2\pi}{N_{\mathrm{DFT}}}}$$

Since

$$nk = \frac{1}{2}\left[n^2 + k^2 - (n-k)^2\right]$$

it follows that

$$Y_{\mathrm{F}n} = W^{\frac{1}{2}n^2} \sum_{k=0}^{N_{\mathrm{DFT}}-1} l_k W^{\frac{1}{2}k^2} W^{-\frac{1}{2}(n-k)^2} \tag{2.5}$$

Therefore, it is possible to define

$$\hat{l}_k = l_k W^{\frac{1}{2}k^2}, \quad h_{n-k} = W^{-\frac{1}{2}(n-k)^2}, \quad \hat{Y}_{\mathrm{F}n} = Y_{\mathrm{F}n} W^{-\frac{1}{2}n^2}$$

and rewrite equation 2.5 as

$$\hat{Y}_{\mathrm{F}n} = \sum_{k=0}^{N_{\mathrm{DFT}}-1} \hat{l}_k h_{n-k}$$

Note that $[\hat{Y}_{\mathrm{F}0}, \hat{Y}_{\mathrm{F}1}, \ldots, \hat{Y}_{\mathrm{F}(N_{\mathrm{DFT}}-1)}]^{\mathrm{T}}$ are the middle N_{DFT} elements obtained from the linear convolution of the length-N_{DFT} vector

$$\hat{\boldsymbol{l}} = \begin{bmatrix} \hat{l}_0 & \hat{l}_1 & \ldots & \hat{l}_{N_{\mathrm{DFT}}-1} \end{bmatrix}^{\mathrm{T}}$$

and the length-$(2N_{\mathrm{DFT}}-1)$ vector

$$\boldsymbol{f} = \begin{bmatrix} h_{-N_{\mathrm{DFT}}+1} & h_{-N_{\mathrm{DFT}}+2} & \ldots & h_{-1} & h_0 & h_1 & \ldots & h_{N_{\mathrm{DFT}}-1} \end{bmatrix}^{\mathrm{T}}$$

This linear convolution can be computed by means of a cyclic convolution. More in detail, the cyclic convolution can be performed between vectors having length equal to a power of two. In particular, it can be considered the smallest power of two that is greater than or equal to $2N_{\mathrm{DFT}} - 1$. We can indicate this number with the letter M. Hence, the circular convolution is performed between two length-M vectors. The first vector $\hat{\boldsymbol{l}}_{\mathbf{zp}}$ is obtained by zero-padding the vector $\hat{\boldsymbol{l}}$. Hence,

$$\hat{\boldsymbol{l}}_{\mathbf{zp}} = \begin{bmatrix} \hat{l}_0 & \hat{l}_1 & \ldots & \hat{l}_{N_{\mathrm{DFT}}-1} & 0 & 0 & 0 & \ldots & 0 \end{bmatrix}^{\mathrm{T}}$$

where the number of appended zeros is equal to $M - N_{\mathrm{DFT}}$. On the other hand, the second vector is

$$\hat{\boldsymbol{f}} = \begin{bmatrix} h_0 & h_1 & \ldots & h_{N_{\mathrm{DFT}}-1} & 0 & \ldots & 0 & h_{-N_{\mathrm{DFT}}+1} & h_{-N_{\mathrm{DFT}}+2} & \ldots & h_{-1} \end{bmatrix}^{\mathrm{T}}$$

The elements of the vector $\hat{\boldsymbol{Y}}_{\mathbf{F}}$ therefore are the first N_{DFT} elements obtained from the circular convolution of $\hat{\boldsymbol{l}}_{\mathbf{zp}}$ with $\hat{\boldsymbol{f}}$. This circular convolution can be computed according to the Circular Convolution Theorem. This theorem states that the circular convolution $\boldsymbol{c}$ of two vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ can be obtained as:

$$\boldsymbol{c} = \mathrm{IDFT}(\mathrm{DFT}(\boldsymbol{a}) \circ \mathrm{DFT}(\boldsymbol{b}))$$

where $\circ$ is the symbol of the element-by-element vector product.

2.4.1 Bluestein's algorithm steps

The algorithm is composed of the following steps:

1. Computation of $\hat{\boldsymbol{l}}$: this is performed according to the formula

$$\hat{l}_k = l_k W^{\frac{1}{2}k^2}$$

Hence, the vector $\hat{\boldsymbol{l}}$ is

$$\hat{\boldsymbol{l}} = \begin{bmatrix} l_0 W^0 \\ l_1 W^{\frac{1}{2}} \\ l_2 W^2 \\ \vdots \\ l_{N_{\mathrm{DFT}}-1} W^{\frac{(N_{\mathrm{DFT}}-1)^2}{2}} \end{bmatrix}$$

2. Definition of the number M: this is the smallest power of two that is greater than or equal to $2N_{\mathrm{DFT}} - 1$.

3. Definition of the vector $\hat{\boldsymbol{l}}_{\mathbf{zp}}$: this is done by padding the vector $\hat{\boldsymbol{l}}$ with zeros so that a length-M vector is obtained.

$$\hat{\boldsymbol{l}}_{\mathbf{zp}} = \begin{bmatrix} \hat{\boldsymbol{l}} \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

4. Definition of the vector $\hat{\boldsymbol{F}}$: the length-M vector $\hat{\boldsymbol{f}}$ is defined as

$$\hat{\boldsymbol{f}} = \begin{bmatrix} h_0 \\ h_1 \\ h_2 \\ \vdots \\ h_{N_{\mathrm{DFT}}-1} \\ 0 \\ \vdots \\ 0 \\ h_{-N_{\mathrm{DFT}}+1} \\ h_{-N_{\mathrm{DFT}}+2} \\ \vdots \\ h_{-1} \end{bmatrix} = \begin{bmatrix} W^0 \\ W^{-\frac{1}{2}} \\ W^{-2} \\ \vdots \\ W^{-\frac{1}{2}(N_{\mathrm{DFT}}-1)^2} \\ 0 \\ \vdots \\ 0 \\ W^{-\frac{1}{2}(-N_{\mathrm{DFT}}+1)^2} \\ W^{-\frac{1}{2}(-N_{\mathrm{DFT}}+2)^2} \\ \vdots \\ W^{-\frac{1}{2}} \end{bmatrix}$$

The vector $\hat{\boldsymbol{F}}$ is therefore obtained as:

$$\hat{\boldsymbol{F}} = \mathrm{DFT}(\hat{\boldsymbol{f}})$$

5. Computation of $\hat{\boldsymbol{L}}_{\mathbf{zp}}$: this is obtained by performing the DFT of the vector $\hat{\boldsymbol{l}}_{\mathbf{zp}}$. Hence, it can be written
$$\hat{\boldsymbol{L}}_{\mathbf{zp}} = \mathrm{DFT}(\hat{\boldsymbol{l}}_{\mathbf{zp}})$$

6. Computation of the vector $\boldsymbol{M}$: this is obtained by performing the element-by-element vector product
$$\boldsymbol{M} = \hat{\boldsymbol{L}}_{\mathbf{zp}} \circ \hat{\boldsymbol{F}}$$

7. Computation of the vector $\hat{\boldsymbol{Y}}_{\mathbf{F}}$: this is obtained by computing the IDFT of the vector $\boldsymbol{M}$. Therefore
$$\hat{\boldsymbol{Y}}_{\mathbf{F}} = \mathrm{IDFT}(\boldsymbol{M})$$

8. Computation of the vector $\boldsymbol{Y}_{\mathbf{F}}$: this is obtained according to the following formula
$$Y_{\mathrm{F}n} = \hat{Y}_{\mathrm{F}n} W^{\frac{1}{2}n^2} \qquad \text{for } n = 0,1,...,N_{\mathrm{DFT}} - 1.$$
Thus, the vector $\boldsymbol{Y}_{\mathbf{F}}$ is:
$$\boldsymbol{Y}_{\mathbf{F}} = \begin{bmatrix} \hat{Y}_{\mathrm{F}0} W^{0} \\ \hat{Y}_{\mathrm{F}1} W^{\frac{1}{2}} \\ \hat{Y}_{\mathrm{F}2} W^{2} \\ \vdots \\ \hat{Y}_{\mathrm{F}(N_{\mathrm{DFT}}-1)} W^{\frac{(N_{\mathrm{DFT}}-1)^2}{2}} \end{bmatrix}$$

2.4.2 Bluestein's algorithm for DCT5

Bluestein's algorithm can be employed to compute the DCT5. In fact, a DCT5 of length N can be mapped into a DFT of length $2N-1$. Hence, the algorithm can be used to compute the DFT and consequently obtain the results produced by the DCT5.

In particular, the vector $\boldsymbol{l}$ is obtained starting from the input vector $\boldsymbol{x}$ and then it is used as input for the DFT as depicted in figure 2.13. The real parts of the elements $[Y_{\mathrm{F}0},Y_{\mathrm{F}1},...,Y_{\mathrm{F}N}]$ represented in this figure, are the outputs of the DCT5 (for simplicity considered in non-normalized form).

Table 2.16: *Values of N_{DFT} and M for different lengths of the DCT5*

N	N_{DFT}	M
4	7	16
8	15	32
16	31	64
32	63	128

Hence, as presented in table 2.16, a DCT5 of length 4 is mapped into a DFT of length 7 and computed by using a module which performs a DFT of length 16. Similarly, a DCT5

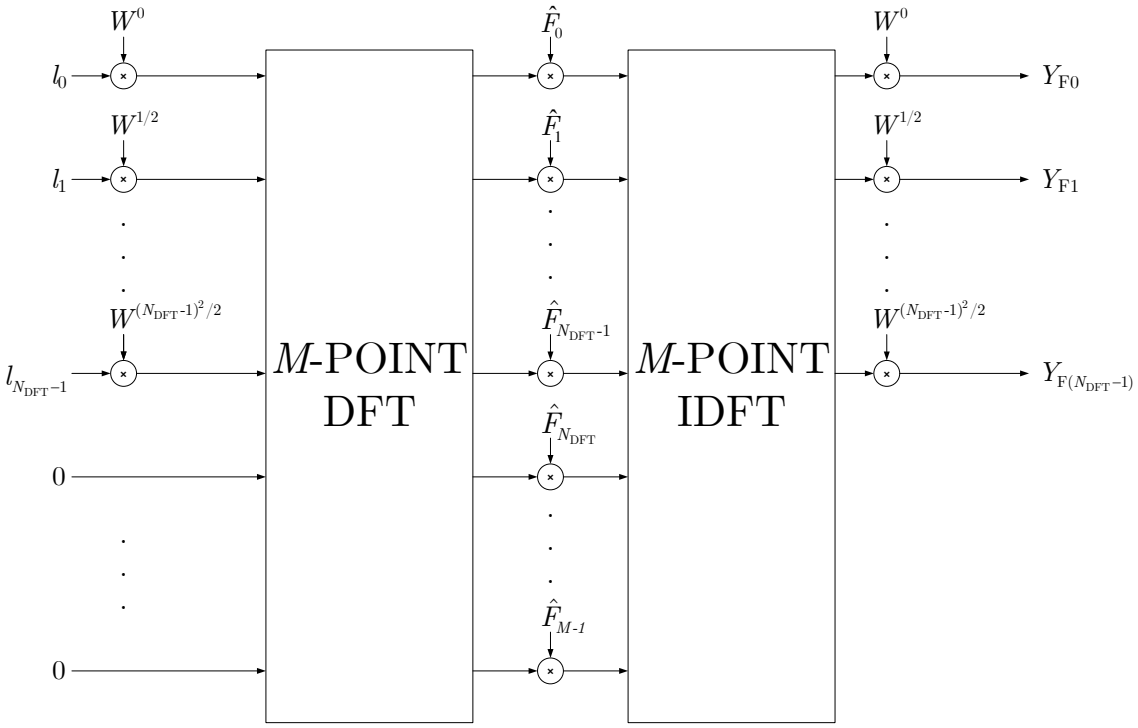


Figure 2.13: *Bluestein's Algorithm*

of length 8 is computed by means of a DFT of length 32 and the DCT5 of length 16 and 32 are respectively performed by adopting modules that compute DFTs of length 64 and 128.

It should be highlighted that a possible algorithm that can be adopted to compute these DFTs is the radix-2 DIT algorithm. This algorithm allows sharing submodules among the different lengths of the DCT5. Moreover, the same module, which is used to compute the DFT, can also be employed to compute the IDFT.

In the following, the radix-2 algorithm will be shortly presented by referring to the results reported in [4]. Furthermore, it will be discussed how to share the submodules of the radix-2 algorithm among the different lengths of the DCT5 and a possible technique that can be adopted to compute the IDFT by means of the DFT will be also described. Finally, the computational complexity will be analyzed as far as the DCT5 is concerned.

The Radix-2 DIT Algorithm

The DFT of a length-M vector $\boldsymbol{b}$ can be expressed as

$$X_{\mathrm{F}n} = \sum_{k=0}^{M-1} b_k \mathrm{e}^{-\mathrm{j}nk\frac{2\pi}{M}} = \sum_{k=0}^{M-1} b_k \mathrm{W}_M^{nk} \qquad \mathrm{W}_M = \mathrm{e}^{-\mathrm{j}\frac{2\pi}{M}} \qquad n = 0,1,...,M-1$$

The equation can be rearranged as

$$X_{\mathrm{F}n} = \sum_{r=0}^{M/2-1} b_{2r}\mathrm{W}_M^{2rn} + \sum_{r=0}^{M/2-1} b_{2r+1}\mathrm{W}_M^{(2r+1)n} = \sum_{r=0}^{M/2-1} b_{2r}\mathrm{W}_M^{2rn} + \mathrm{W}_M^n \sum_{r=0}^{M/2-1} b_{2r+1}\mathrm{W}_M^{2rn}$$

Since

$$\mathrm{W}_M^{2rn} = \mathrm{W}_{M/2}^{rn}$$

it can be derived

$$\begin{aligned} X_{\mathrm{F}n} &= \sum_{r=0}^{M/2-1} b_{2r}\mathrm{W}_{M/2}^{rn} + \mathrm{W}_M^n \sum_{r=0}^{M/2-1} b_{2r+1}\mathrm{W}_{M/2}^{rn} \\ &= G_n + \mathrm{W}_M^n H_n \qquad n = 0,1,...,\frac{M}{2} - 1 \end{aligned}$$

Hence, an M-point DFT is formulated in terms of $M/2$-point DFTs, G_n and H_n, which are DFTs of even samples and odd samples of $\boldsymbol{b}$ respectively.

Since G_n and H_n are periodic with period $\frac{M}{2}$, we have

$$G_{n+\frac{M}{2}} = G_n \quad H_{n+\frac{M}{2}} = H_n$$

Therefore, it can be written

$$X_{\mathrm{F}(n+M/2)} = G_n + \mathrm{W}_M^{n+M/2} H_n$$

Since

$$\mathrm{W}_M^{M/2} = -1$$

it follows that

$$X_{\mathrm{F}(n+M/2)} = G_n - \mathrm{W}_M^n H_n \qquad n = 0,1,...,\frac{M}{2} - 1$$

Hence, the computation of $X_{\mathrm{F}n}$ and $X_{\mathrm{F}(n+M/2)}$ can be performed by using a butterfly unit as depicted in figure 2.14.

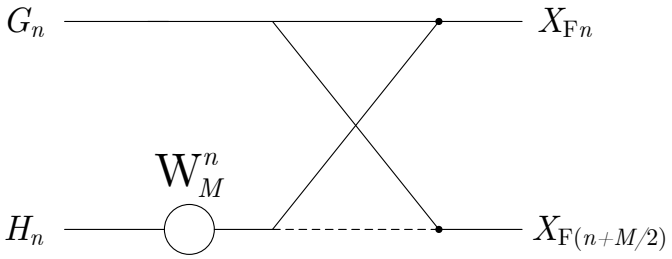


Figure 2.14: *Butterfly Unit*

We now consider $M = 8$. The scheme presented in figure 2.15 can therefore be derived from the description reported above.

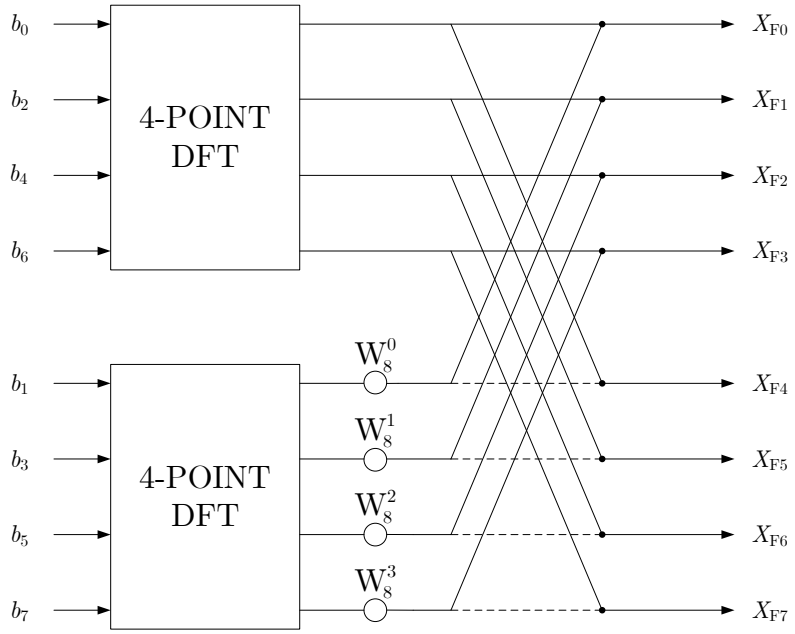


Figure 2.15: *8-Point DFT expressed in terms of two 4-point DFTs*

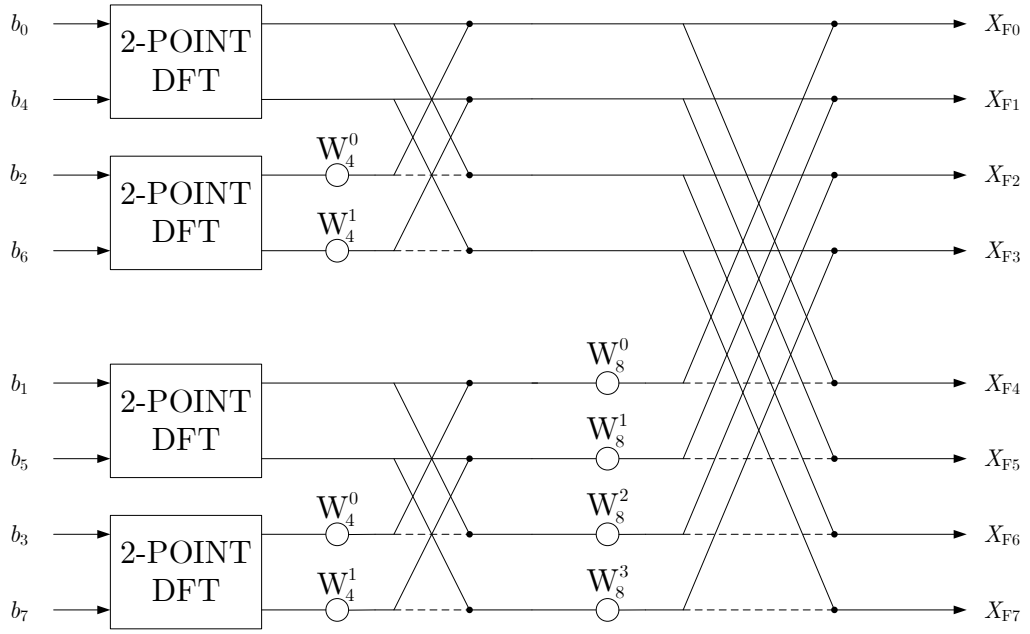


Figure 2.16: *8-Point DFT expressed in terms of four 2-point DFTs*

The process can be iteratively repeated until two-point DFTs are obtained. Thus, for $M = 8$, the scheme presented in figure 2.16 can be drawn.

The 2-point DFTs can be implemented by using butterfly units. The SFG presented

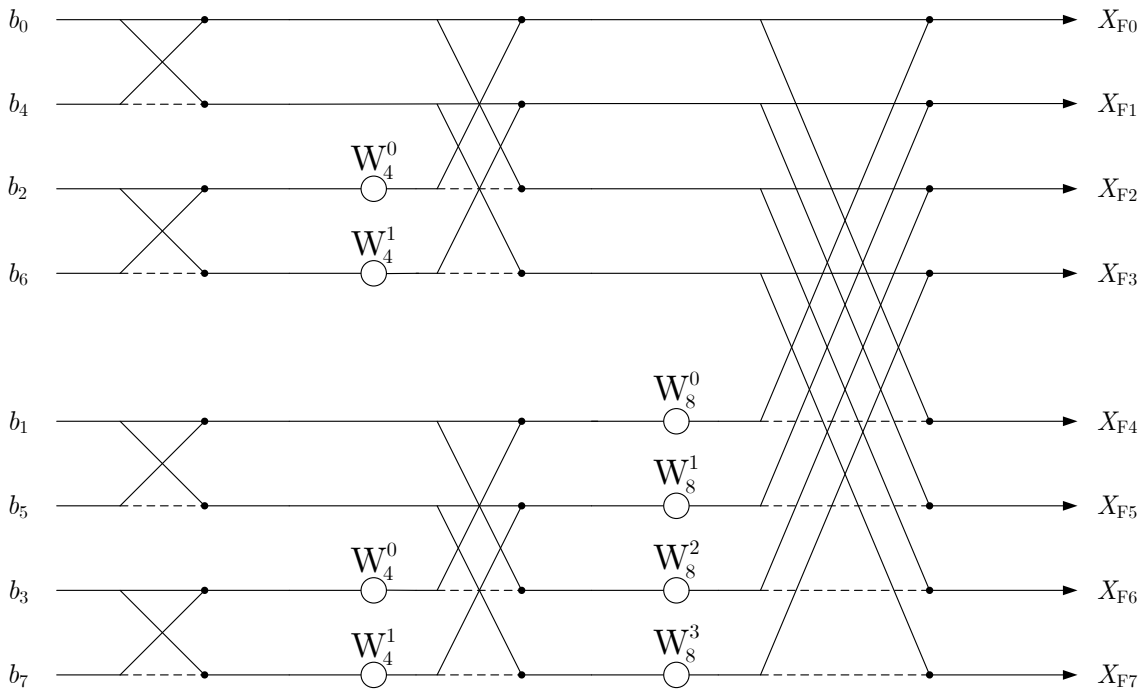
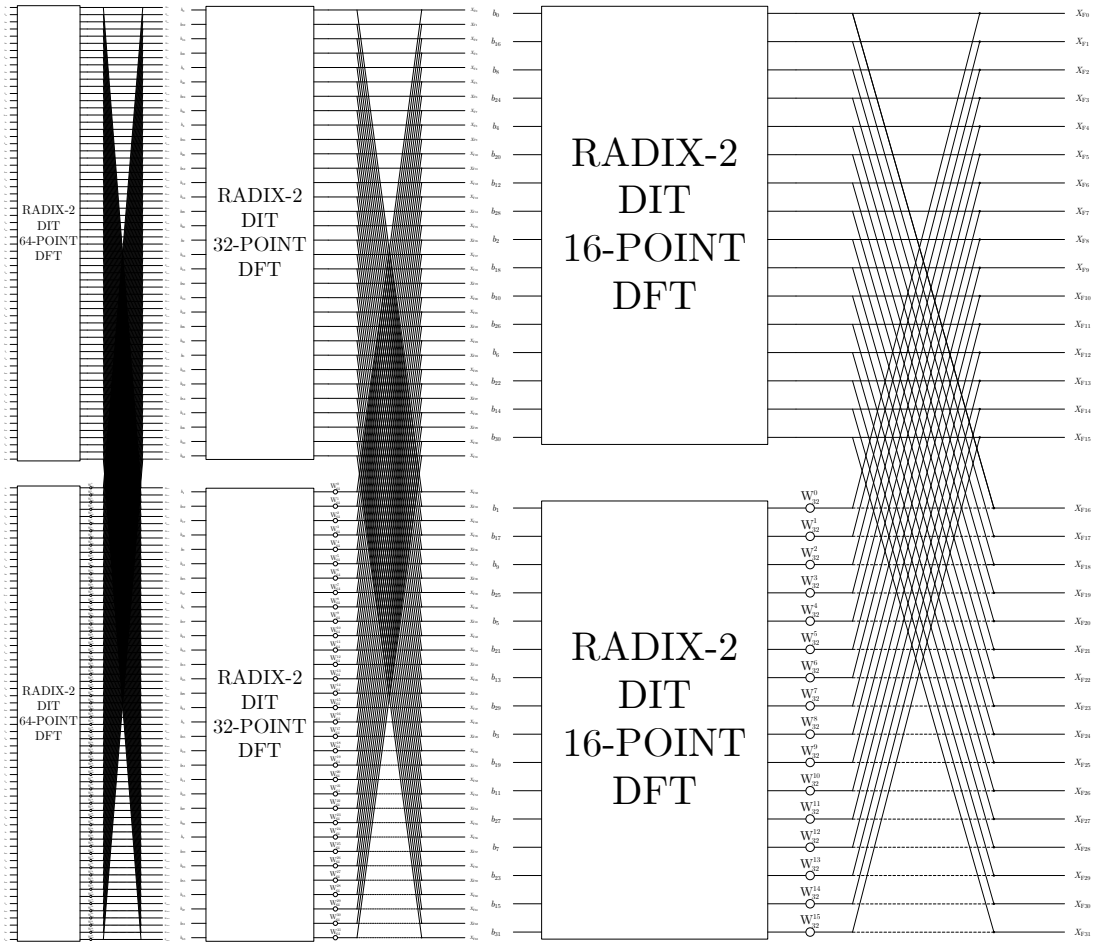


Figure 2.17: *Radix-2 DIT algorithm applied to an 8-Point DFT*

in figure 2.17 is therefore obtained for $M = 8$. It should be highlighted that the inputs are organized in bit reversed order.

Shared submodules

According to what is described in the previous paragraphs, a DCT5 of length 32 is computed by performing a DFT of length 128. Similarly, a DCT5 of length 16 can be expressed in terms of a DFT of length 64. Since the SFG presented in figure 2.18(a) shows that a DFT of length 128 can be computed by means of two DFTs of length 64, each of the modules, used to implement the 64-point DFTs in the computation of a length-32 DCT5, can be shared to obtain a DCT5 of length 16. Hence, the 128-point DFT, required for the computation of a 32-point DCT5, can also be used to compute two 16-point DCT5. In a similar way, the computation of a 64-point DFT can be performed by means of two 32-point DFTs as depicted in figure 2.18(b). Therefore, each of the modules, used to implement the 32-point DFTs in the computation of a length-32 DCT5, can also be employed to compute a DCT5 of length 8. Thus, the DFT module used to compute the 32-point DCT5 can also be useful for the computation of four 8-point DCT5. Finally, a DFT of length 32 can be obtained from two DFTs of length 16 (figure 2.18(c)). Hence, the hardware that is required for the computation of a 128-point DFT can also be employed for obtaining eight 16-point DFTs. Therefore, eight DCT5 of length 4 can be partially computed by using the hardware required for the computation of a length-32 DCT5.



(a) *128-Point* (b) *64-Point* (c) *32-Point*

Figure 2.18: *Submodules of the Radix-2 DIT algorithm for different lengths of the DFT*

IDFT via DFT

There are several possible techniques that can be adopted to compute the IDFT via a forward DFT ([10], [11]). Given an input vector $\boldsymbol{B}$, one possible method consists in reversing the order of the samples in the vector $[B_1,...,B_{N_{\mathrm{IDFT}}-1}]$ and appending the obtained vector to B_0. The vector

$$\hat{\boldsymbol{B}} = \begin{bmatrix} B_0 & B_{N_{\mathrm{IDFT}}-1} & B_{N_{\mathrm{IDFT}}-2} & B_{N_{\mathrm{IDFT}}-3} & \dots & B_3 & B_2 & B_1 \end{bmatrix}^{\mathrm{T}}$$

is therefore derived from this procedure. Hence, this is used as input for the DFT. The outputs of the DFT are then divided by N_{IDFT}. The method is presented in figure 2.19.

Since the IDFT can be computed via DFT, Bluestein's algorithm can be executed by an architecture similar to the one depicted in figure 2.20.

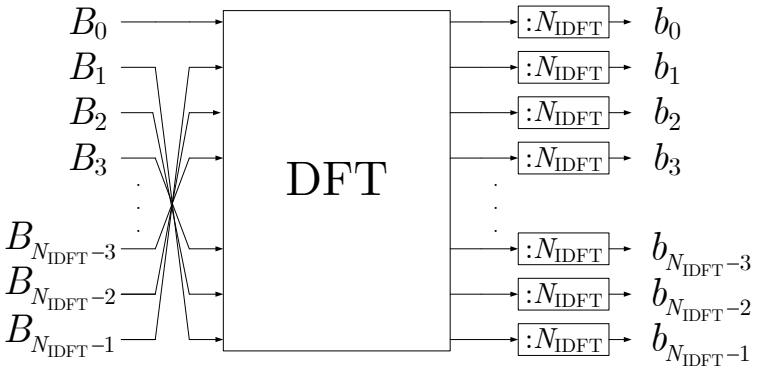


Figure 2.19: *IDFT via DFT*

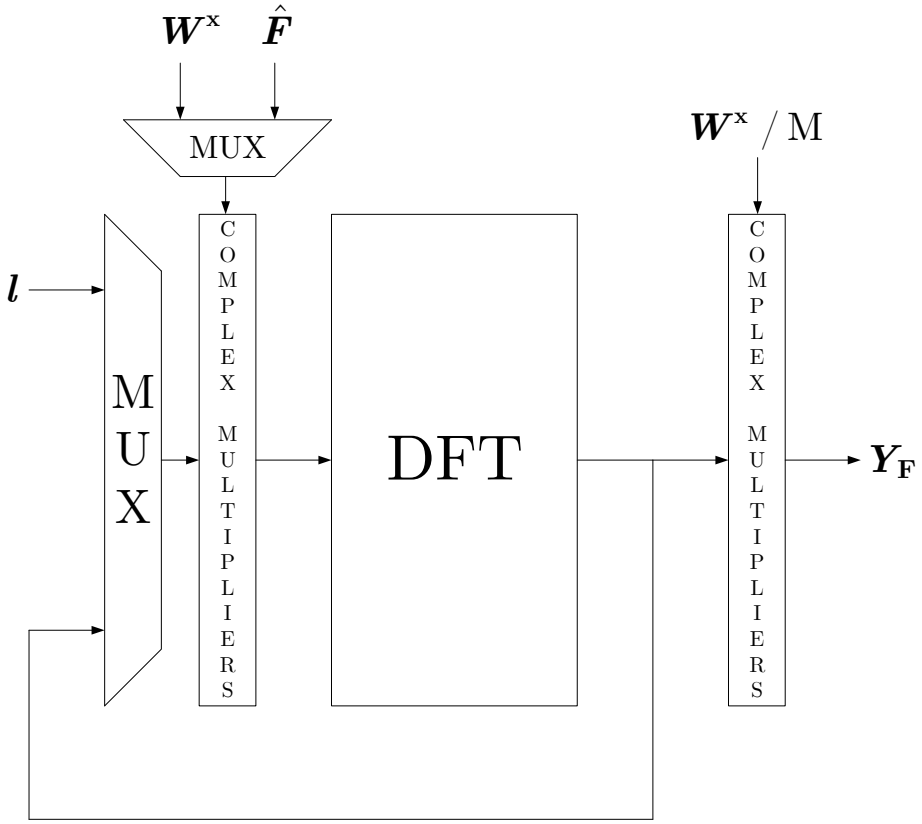


Figure 2.20: *Architecture for Bluestein's algorithm*

Computational complexity

The architecture presented in figure 2.20 requires N_{mult} complex multipliers. This number can be approximately computed according to the following formula:

$$N_{\mathrm{mult}} = M + \frac{M}{2}\log_2(M) + N$$

Each complex multiplication can be performed by means of 4 real multiplications and 2 real additions or 3 real multiplications and 5 real additions [6]. More in detail, the SFGs depicted in figures 2.21 and 2.22 can be obtained.

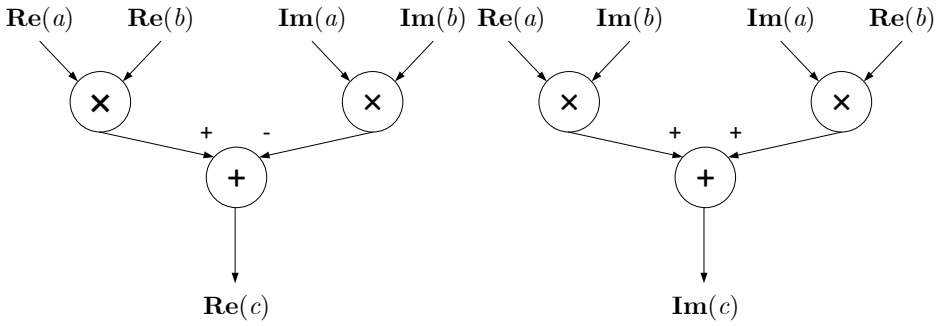


Figure 2.21: *Complex multiplication between a and b (4 real multiplications and 2 real additions)*

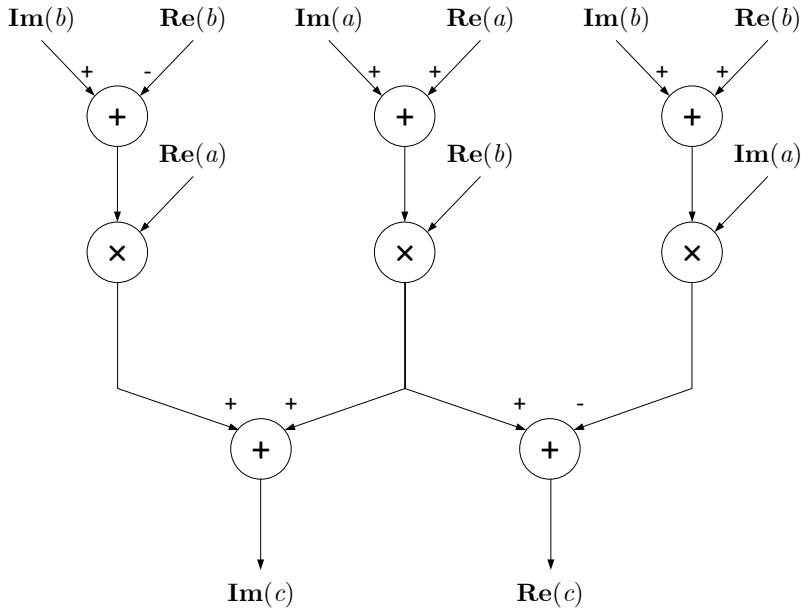


Figure 2.22: *Complex multiplication between a and b (3 real multiplications and 5 real additions)*

The numbers of real multipliers required for different values of N can therefore be analyzed by choosing the implementation represented in figure 2.22. Moreover, the number of real multipliers required by the direct implementation of the matrix-vector multiplication can be compared to the one needed by the algorithm. This comparison is made in table 2.17.

It should be highlighted that, for large values of N, Bluestein's algorithm requires a lower number of real multipliers than the direct implementation of the matrix-vector product.

Table 2.17: *Number of multipliers for several values of N (MVM = Matrix-Vector Multiplication)*

	Number of real multipliers	
N	*MVM*	*Bluestein's Algorithm*
4	16	156
8	64	360
16	256	816
32	1024	1824
64	4096	4032
128	16 384	8832

Remarks

The Radix 2 DIT algorithm is only one of the many possible algorithms that can be adopted to compute a DFT having length equal to a power of 2. Among the others, there are the Radix 2 DIF algorithm and the split-radix algorithm.

2.5 Rader's Algorithm

We have seen that any DFT can be translated into a convolution by the *chirp-z transform algorithm* at the cost of $2N_{\mathrm{DFT}}$ complex multiplications performed on the input and output data samples. We shall see now that DFTs can also be turned into circular convolutions by a completely different method ([4], [6], and [12]). This method is, in some cases, computationally more efficient than the *chirp z-transform algorithm* because premultiplications and postmultiplications are replaced by a simple rearrangement of input and output data samples, as depicted in figure 2.23. The steps of the algorithm (that can be adopted when N_{DFT} is an odd prime or a power of an odd prime) will be presented in the following.

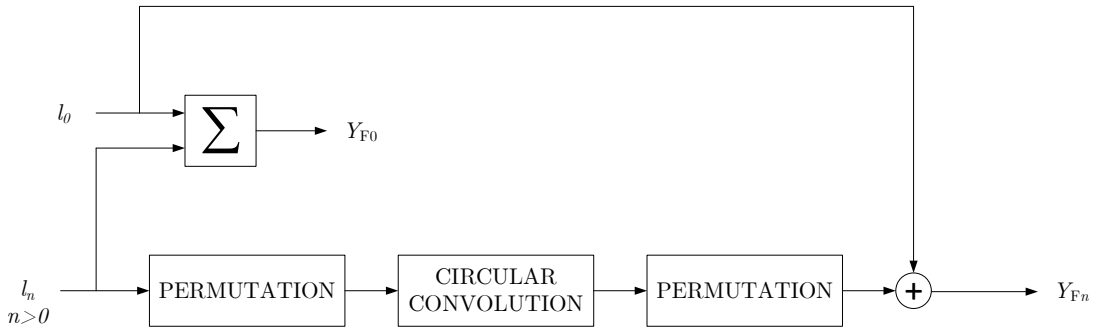


Figure 2.23: *Rader's Algorithm*

2.5.1 Rader's algorithm steps

The algorithm is composed of the following steps:

1. Computation of Y_{F0}: this can be computed according to the following formula

$$Y_{\mathrm{F0}} = \sum_{n=0}^{N_{\mathrm{DFT}}-1} l_n$$

2. Definition of a *primitive root* g of N_{DFT}: as stated in [8]

> In modular arithmetic, a branch of number theory, a number g is a primitive root modulo N_{DFT} if every number a coprime to N_{DFT} is congruent to a power of g modulo N_{DFT}. That is, for every integer a coprime to N_{DFT}, there is an integer k such that $g^k = a \bmod N_{\mathrm{DFT}}$.

Primitive roots can be easily found by using commercial softwares like *WolframAlpha*.

3. Definition of the vector $\hat{\boldsymbol{l}}$: the input permutation applied to the vector $\boldsymbol{l}$ generates the vector $\hat{\boldsymbol{l}}$. More in detail, we have

$$\hat{l}_n = l_{\left(g^{N_{\mathrm{DFT}}-1-n} \bmod N_{\mathrm{DFT}}\right)} \qquad \text{for } n = 0,1,\ldots,N_{\mathrm{DFT}}-2$$

4. Definition of the vector $\boldsymbol{T}$: by defining

$$W = \mathrm{e}^{-\mathrm{j}\frac{2\pi}{N_{\mathrm{DFT}}}}$$

the vector $\boldsymbol{T}$ can be obtained as

$$T_n = W^{(g^n \bmod N_{\mathrm{DFT}})} \qquad \text{for } n = 0,1,\ldots,N_{\mathrm{DFT}}-2$$

5. Computation of the circular convolution $\boldsymbol{c}$ of $\hat{\boldsymbol{l}}$ with $\boldsymbol{T}$: this can be computed according to the Circular Convolution Theorem. Hence, we have:

$$\boldsymbol{c} = \mathrm{IDFT}(\mathrm{DFT}(\hat{\boldsymbol{l}}) \circ \mathrm{DFT}(\boldsymbol{T}))$$

where $\circ$ is the symbol of the element-by-element vector product.

6. Computation of the vector $\hat{\boldsymbol{c}}$: this is computed according to the following formula

$$\hat{c}_n = c_n + l_0$$

7. Definition of the vector $\boldsymbol{Y_F}$: Y_{F0} is defined at point 1). The other elements of the vector are stored in the vector $\hat{\boldsymbol{c}}$ according to the following permutation:

$$Y_{\mathrm{F}(g^n \bmod N_{\mathrm{DFT}})} = \hat{c}_n \qquad \text{for } n = 0,1,\ldots,N_{\mathrm{DFT}}-2$$

2.5.2 Rader's algorithm for DCT5 ($N = 4$)

A DCT5 of length 4 can be mapped into a DFT of length 7 as described in section 2.1. In particular, the DCT5 equation can be written as

$$Y_n = \mathrm{c}_4 T_n \mathbf{Re}(Y_{\mathrm{F}n}) \qquad \text{for } n = 0,1,2,3 \tag{2.6}$$

where

- $\mathrm{c}_4 = \frac{2}{\sqrt{7}}$
- $\mathbf{Re}(Y_{\mathrm{F}n}) = \mathbf{Re}\left(\sum_{k=0}^{6} l_k \mathrm{e}^{-\mathrm{j}nk\frac{2\pi}{7}}\right)$

and

$$\boldsymbol{l} = \begin{bmatrix} l_0 \\ l_1 \\ l_2 \\ l_3 \\ l_4 \\ l_5 \\ l_6 \end{bmatrix} = \begin{bmatrix} \hat{x}_0 \\ 0 \\ \hat{x}_2 \\ 0 \\ \hat{x}_3 \\ 0 \\ \hat{x}_1 \end{bmatrix} = \begin{bmatrix} \frac{x_0}{\sqrt{2}} \\ 0 \\ x_2 \\ 0 \\ x_3 \\ 0 \\ x_1 \end{bmatrix}$$

The computation of the DCT5 is therefore translated into the computation of a DFT of length $N_{\mathrm{DFT}} = 7$. Since N_{DFT} is a prime number, Rader's algorithm can be adopted to compute the DFT. Hence, the steps described in the previous paragraph can be followed to compute the DCT5. In the following, each of these steps will be analyzed.

Computation of $Y_{\mathrm{F}0}$

The computation of $Y_{\mathrm{F}0}$ is performed according to the following formula:

$$Y_{\mathrm{F}0} = \sum_{n=0}^{N_{\mathrm{DFT}}-1} l_n = \frac{x_0}{\sqrt{2}} + x_1 + x_2 + x_3$$

Hence, Y_0 can be obtained from $Y_{\mathrm{F}0}$ as follows

$$Y_0 = \mathrm{c}_4 T_0 \mathbf{Re}(Y_{\mathrm{F}0}) = \frac{2}{\sqrt{7}} \frac{1}{\sqrt{2}} Y_{\mathrm{F}0}$$

Definition of a primitive root g of N_{DFT}

A primitive root of 7 is

$$g = 3$$

Definition of the vector $\hat{\boldsymbol{l}}$

The vector $\hat{\boldsymbol{l}}$ is defined according to the permutation:

$$\hat{l}_n = l_{(3^{6-n} \bmod 7)} \qquad \text{for } n = 0,1,\ldots,5$$

Hence, it can be written:

$$\begin{aligned}
\hat{l}_0 &= l_{3^{6-0} \bmod 7} = l_1 \\
\hat{l}_1 &= l_{3^{6-1} \bmod 7} = l_5 \\
\hat{l}_2 &= l_{3^{6-2} \bmod 7} = l_4 \\
\hat{l}_3 &= l_{3^{6-3} \bmod 7} = l_6 \\
\hat{l}_4 &= l_{3^{6-4} \bmod 7} = l_2 \\
\hat{l}_5 &= l_{3^{6-5} \bmod 7} = l_3
\end{aligned}$$

Therefore, we have

$$\hat{\boldsymbol{l}} = \begin{bmatrix} 0 \\ 0 \\ x_3 \\ x_1 \\ x_2 \\ 0 \end{bmatrix}$$

Definition of the vector $\boldsymbol{T}$

The vector $\boldsymbol{T}$ is defined according to the following formula:

$$T_n = W^{(3^n \bmod 7)} \qquad \text{for } n = 0,1,\ldots,5$$

Hence, it can be written

$$\begin{aligned}
T_0 &= W^{(3^0 \bmod 7)} = W^1 \\
T_1 &= W^{(3^1 \bmod 7)} = W^3 \\
T_2 &= W^{(3^2 \bmod 7)} = W^2 \\
T_3 &= W^{(3^3 \bmod 7)} = W^6 \\
T_4 &= W^{(3^4 \bmod 7)} = W^4 \\
T_5 &= W^{(3^5 \bmod 7)} = W^5
\end{aligned}$$

Therefore we have

$$\boldsymbol{T} = \begin{bmatrix} W^1 & W^3 & W^2 & W^6 & W^4 & W^5 \end{bmatrix}^{\mathrm{T}}$$

Computation of the circular convolution $\boldsymbol{c}$ of $\hat{\boldsymbol{l}}$ with $\boldsymbol{T}$

The circular convolution $\boldsymbol{c}$ can be computed according to the Circular Convolution Theorem. Hence, we have:

$$\boldsymbol{c} = \mathrm{IDFT}(\mathrm{DFT}(\hat{\boldsymbol{l}}) \circ \mathrm{DFT}(\boldsymbol{T}))$$

Since $\boldsymbol{T}$ contains only constants, $\boldsymbol{T_F} = \mathrm{DFT}(\boldsymbol{T})$ can be precomputed. On the contrary, the DFT of the vector $\hat{\boldsymbol{l}}$ can be computed according to the WFTA or the PFA. In fact, $\hat{\boldsymbol{l}}$ has length $M = 6 = 3 \times 2$. Therefore, the Winograd short-N DFT modules of length 2 and 3 can be used to compute the DFT. Moreover, the IDFT can be obtained by using a forward DFT as depicted in figure 2.19.

Computation of the vector $\hat{\boldsymbol{c}}$

The vector $\hat{\boldsymbol{c}}$ is computed according to the following formula:

$$\hat{c}_n = (\mathbf{Re}(c_n) + l_0)\, \mathrm{c}_4$$

Definition of the vector $\boldsymbol{Y}$

The elements $[Y_1,Y_2,Y_3]$ of the vector $\boldsymbol{Y}$ are stored in the vector $\hat{\boldsymbol{c}}$ according to the following permutation:

$$Y_{(3^n \bmod 7)} = \hat{c}_n \qquad \text{for } n = 0,1,2$$

Hence, we have:

$$\begin{aligned} Y_1 &= Y_{(3^0 \bmod 7)} = \hat{c}_0 \\ Y_2 &= Y_{(3^2 \bmod 7)} = \hat{c}_2 \\ Y_3 &= Y_{(3^1 \bmod 7)} = \hat{c}_1 \end{aligned}$$

Final algorithm

Unfortunately, the procedure described above generates an algorithm characterized by a high computational complexity. In fact, supposing that the PFA is adopted to compute the DFT, the architecture depicted in figure 2.24 can be derived. Since the number of real multiplications needed for each 3-point DFT is equal to 4 and the number of real multiplications needed for each complex multiplication is at least equal to 3, the number of real multipliers (N_{mult}) needed to implement this architecture is:

$$N_{\mathrm{mult}} = 6 + 4 \times 2 + 6 \times 3 = 32$$

This number is greater than the one obtained by directly implementing the matrix-vector multiplication. This makes the algorithm unpractical for most of applications. As we will see in appendix A, the results obtained about the computational complexity slightly improve when the algorithm is applied to the case $N = 16$.

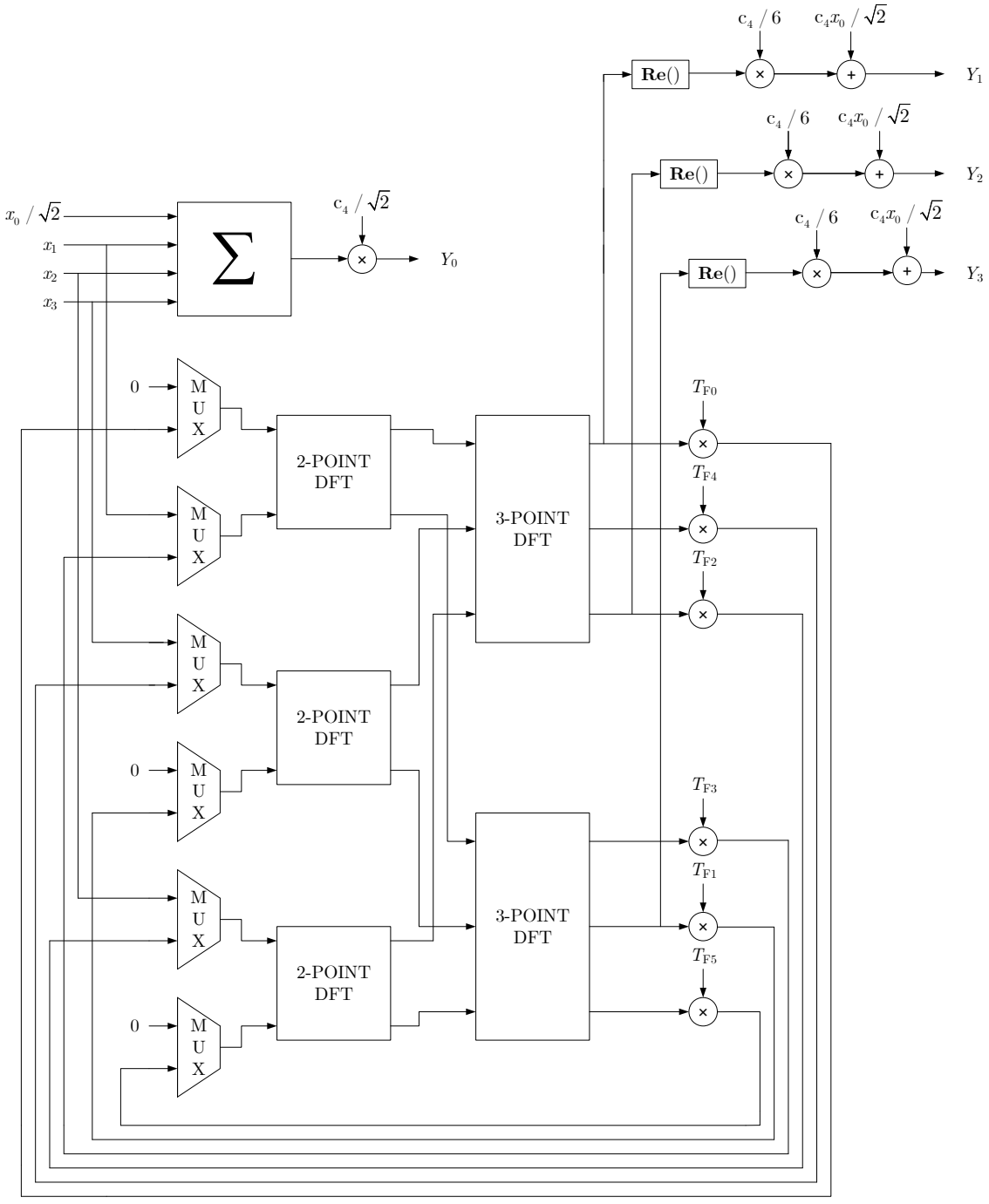


Figure 2.24: *Architecture for DCT5 ($N = 4$) based on Rader and Prime Factor algorithms*

Chapter 3

DCT5 via DCT2

This chapter presents the algorithms for the DCT5 that can be obtained from algorithms for the Discrete Cosine Transform Type 2 (DCT2). More in detail, the relationships between the DCT5 and the Discrete Cosine Transform Type 6 (DCT6) will be presented as derived in [13]. Successively, the DCT6 will be related to the DCT2 as reported in [14] and finally, the DCT2 will be calculated according to the algorithms presented in [15], [16] and [17].

3.1 Relationship between the DCT5 and the DCT6

As described in [13], the DCT5 can be derived from the DCT6 according to the following relationship

$$\left[C_N^{\mathrm{V}}\right] = [D_N]\left[C_N^{\mathrm{VI}}\right][J_N]$$

where:

- $\left[C_N^{\mathrm{V}}\right]$ is the N-point DCT5 matrix.
- $[D_N]$ is the diagonal matrix implementing the sign-alteration. More in detail, it can be written

$$[D_N] = \begin{bmatrix} 1 & 0 & \dots & 0 & 0 \\ 0 & -1 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 1 & 0 \\ 0 & 0 & \dots & 0 & -1 \end{bmatrix}$$

- $\left[C_N^{\mathrm{VI}}\right]$ is the N-point DCT6 matrix.

- $[J_N]$ is the backward identity matrix. Hence, it can be written

$$[J_N] = \begin{bmatrix} 0 & 0 & \dots & 0 & 1 \\ 0 & 0 & \dots & 1 & 0 \\ \vdots & \vdots & \iddots & \vdots & \vdots \\ 0 & 1 & \dots & 0 & 0 \\ 1 & 0 & \dots & 0 & 0 \end{bmatrix}$$

3.2 Relationship between the DCT6 and the DCT2

As stated in [14], the following relationship holds between the DCT2 and the DCT6:

$$\left[C_{2N+1}^{\text{II}}\right] = [Q_{2N+1}] \begin{bmatrix} \left[C_{N+1}^{\text{VI}}\right] & \\ & \left[S_N^{\text{VII}}\right] \end{bmatrix} \begin{bmatrix} [I_N] & & [J_N] \\ & 1 & \\ -[J_N] & & [I_N] \end{bmatrix} \tag{3.1}$$

where:

- $\left[C_{2N+1}^{\text{II}}\right]$ is the $2N+1$-point DCT2 matrix;
- $[Q_{2N+1}]$ is a sign alteration and reordering matrix, which acts as in the following:

$$\begin{aligned} \hat{x}_{2n} &= x_n \qquad n = [0,N] \\ \hat{x}_{2n+1} &= (-1)^{n+1} x_{N+1+n} \qquad n = [0,N-1] \end{aligned}$$

- $\left[C_{N+1}^{\text{VI}}\right]$ is the $N+1$-point DCT6 matrix;
- $\left[S_N^{\text{VII}}\right]$ is the N-pont Discrete Sine Transform Type 7 (DST7) matrix;
- $[I_N]$, $[J_N]$ are respectively the N-point identity and backward identity matrix.

3.2.1 Derivation of the 4-point DCT6 from the 7-point DCT2

The relationship expressed by equation 3.1 is better explained by an example. Consider, for instance, $N+1 = 4$. For this particular case, we have:

$$[Q_{2N+1}] = [Q_7] = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

According to what is described above, it can be written:

$$\begin{bmatrix} [C_4^{\mathrm{VI}}] & \\ & [S_3^{\mathrm{VII}}] \end{bmatrix} = [Q_7]^{-1} \left[C_7^{\mathrm{II}}\right] \begin{bmatrix} [I_3] & & [J_3] \\ & 1 & \\ -[J_3] & & [I_3] \end{bmatrix}^{-1}$$

The 4-point DCT6 can therefore be obtained by performing the matrix products

$$\hat{\boldsymbol{Y}}^{\mathbf{II}} = [Q_7]^{-1} \left[C_7^{\mathrm{II}}\right] \begin{bmatrix} [I_3] & & [J_3] \\ & 1 & \\ -[J_3] & & [I_3] \end{bmatrix}^{-1} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ -x_2 \\ -x_1 \\ -x_0 \end{bmatrix}$$

and taking the first four elements of the output vector $\hat{\boldsymbol{Y}}^{\mathbf{II}}$. In the following each of the steps needed to obtain the output vector will be analyzed.

Step 1

The following operation is performed:

$$\hat{\boldsymbol{x}} = \begin{bmatrix} [I_3] & & [J_3] \\ & 1 & \\ -[J_3] & & [I_3] \end{bmatrix}^{-1} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ -x_2 \\ -x_1 \\ -x_0 \end{bmatrix} = \begin{bmatrix} x_0 \\ 0 \\ x_2 \\ x_3 \\ 0 \\ x_1 \\ 0 \end{bmatrix}$$

Step 2

The following operation is performed:

$$\boldsymbol{Y}^{\mathbf{II}} = \left[C_7^{\mathrm{II}}\right] \begin{bmatrix} x_0 \\ 0 \\ x_2 \\ x_3 \\ 0 \\ x_1 \\ 0 \end{bmatrix}$$

This operation is the computation of the DCT2 of the vector obtained at the end of step 1.

Step 3

The following operation is performed:

$$\hat{\boldsymbol{Y}}^{\mathbf{II}} = [Q_7]^{-1}\, \boldsymbol{Y}^{\mathbf{II}} = \begin{bmatrix} Y_0^{\mathrm{II}} \\ Y_2^{\mathrm{II}} \\ Y_4^{\mathrm{II}} \\ Y_6^{\mathrm{II}} \\ -Y_1^{\mathrm{II}} \\ Y_3^{\mathrm{II}} \\ -Y_5^{\mathrm{II}} \end{bmatrix}$$

Steps for the computation of the DCT6 via DCT2

According to what is described in the previous paragraphs, the 4-point DCT6 of the input vector

$$\boldsymbol{x} = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

can be obtained by performing the following operations:

1. definition of the vector
$$\hat{\boldsymbol{x}} = \begin{bmatrix} x_0 \\ 0 \\ x_2 \\ x_3 \\ 0 \\ x_1 \\ 0 \end{bmatrix}$$
2. computation of the vector $\boldsymbol{Y}^{\mathbf{II}}$ obtained by computing the DCT2 of the vector $\hat{\boldsymbol{x}}$;
3. definition of the outputs of the DCT6, which are the even-indexed elements of the vector $\boldsymbol{Y}^{\mathbf{II}}$.

3.3 DCT5 via DCT2 ($N = 4$)

According to what is described above, an algorithm for the 4-point DCT5 can be derived from algorithms for the 7-point DCT2. More in detail, the computation of a 4-point DCT5 can be translated into the computation of a 4-point DCT6 as described in section 3.1 and the 4-point DCT6 can be derived from the 7-point DCT2 as illustrated in section 3.2. The following paragraphs will present the steps involved in the derivation of the algorithm.

3.3.1 Reordering of the input vector

According to what is described in section 3.1, the input vector $\boldsymbol{x}$ is reordered so that the vector

$$\boldsymbol{x_r} = [J_4]\,\boldsymbol{x} = \begin{bmatrix} x_3 \\ x_2 \\ x_1 \\ x_0 \end{bmatrix}$$

is obtained.

3.3.2 Definition of the vector $\boldsymbol{x_R}$

According to what is described in section 3.2, the vector $\boldsymbol{x_R}$ is obtained as

$$\boldsymbol{x_R} = \begin{bmatrix} [I_3] & & [J_3] \\ & 1 & \\ -[J_3] & & [I_3] \end{bmatrix}^{-1} \begin{bmatrix} x_3 \\ x_2 \\ x_1 \\ x_0 \\ -x_1 \\ -x_2 \\ -x_3 \end{bmatrix} = \begin{bmatrix} x_3 \\ 0 \\ x_1 \\ x_0 \\ 0 \\ x_2 \\ 0 \end{bmatrix}$$

3.3.3 Computation of the 7-point DCT2

The vector $\boldsymbol{Y^{II}}$ is obtained by computing the DCT2 of the vector $\boldsymbol{x_R}$. A possible algorithm for the 7-point DCT2 can be derived by relating the DCT2 to a 7-point DFT as described in [15] and using the algorithm for the 7-point DFT reported in [17]. Several simplifications can be made to this algorithm. In fact:

1. 3 out of 7 inputs are equal to zero;

2. only the even-indexed outputs of the DCT2 are of interest.

Hence, the computation of the DCT2 leads to the vector:

$$\boldsymbol{Y^{II}} = \begin{bmatrix} Y_0^{\mathrm{II}} \\ U \\ Y_2^{\mathrm{II}} \\ U \\ Y_4^{\mathrm{II}} \\ U \\ Y_6^{\mathrm{II}} \end{bmatrix}$$

where the elements labeled as U are not of interest for the computation of the 4-point DCT5. More in detail, the output vector produced by the computation of the DCT6 is

$$\boldsymbol{Y}^{\mathbf{VI}} = \begin{bmatrix} Y_0^{\mathrm{VI}} \\ Y_1^{\mathrm{VI}} \\ Y_2^{\mathrm{VI}} \\ Y_3^{\mathrm{VI}} \end{bmatrix} = \begin{bmatrix} Y_0^{\mathrm{II}} \\ Y_2^{\mathrm{II}} \\ Y_4^{\mathrm{II}} \\ Y_6^{\mathrm{II}} \end{bmatrix}$$

3.3.4 Definition of the output vector $\boldsymbol{Y}$

According to what is described in section 3.1, the output vector $\boldsymbol{Y}$ can be obtained from the vector $\boldsymbol{Y}^{\mathbf{VI}}$ as follows:

$$\boldsymbol{Y} = [D_4]\,\boldsymbol{Y}^{\mathbf{VI}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} Y_0^{\mathrm{VI}} \\ Y_1^{\mathrm{VI}} \\ Y_2^{\mathrm{VI}} \\ Y_3^{\mathrm{VI}} \end{bmatrix} = \begin{bmatrix} Y_0^{\mathrm{VI}} \\ -Y_1^{\mathrm{VI}} \\ Y_2^{\mathrm{VI}} \\ -Y_3^{\mathrm{VI}} \end{bmatrix}$$

3.3.5 Final Algorithm

The SFG of the 4-point DCT5 obtained from the SFG of the 7-point DCT2 is reported in figure 3.1 while the values of the constants and the list of the operations needed are respectively reported in tables 3.1 and 3.2.

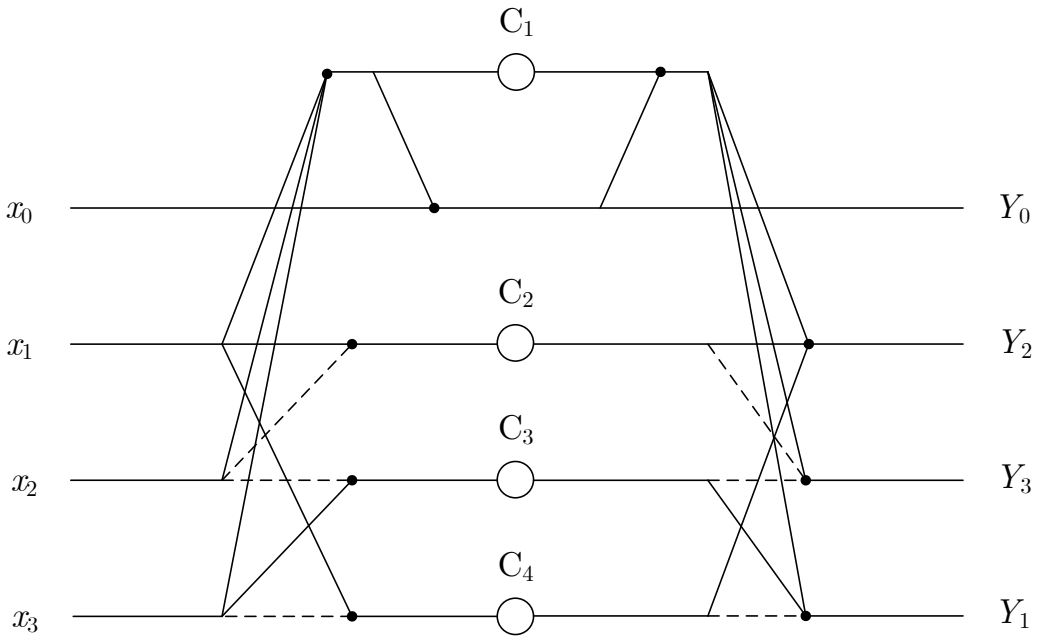


Figure 3.1: *Non-normalized 4-point DCT5 SFG obtained from the 7-point DCT2 SFG*

Table 3.1: *Constants for the 4-point DCT5 algorithm derived from the 7-point DCT2 algorithm*

Constant	*Value*	*Constant*	*Value*
C_1	−1.166 666 67	C_3	0.055 854 27
C_2	0.734 302 20	C_4	−0.790 156 47

Table 3.2: *Algorithm for the 4-point DCT5 derived from the 7-point DCT2*

Op. Name	*Operation*	*Op. Name*	*Operation*	*Op. Name*	*Operation*
$M_{\mathrm{n}1}$	$\mathrm{C}_{\mathrm{norm}} x_0$	a_4	$m_0 + m_1$	a_9	$a_8 + m_4$
a_1	$x_1 + x_2$	a_5	$x_1 - x_2$	Y_2	a_9
a_2	$x_3 + a_1$	m_2	$\hat{\mathrm{C}}_2 a_5$	a_{10}	$-m_3 - m_2$
a_3	$a_2 + M_{\mathrm{n}1}$	a_6	$x_3 - x_2$	a_{11}	$a_{10} + a_4$
m_0	$\mathrm{c}_4 a_3$	m_3	$\hat{\mathrm{C}}_3 a_6$	Y_3	a_{11}
$M_{\mathrm{n}2}$	$\mathrm{C}_{\mathrm{norm}} m_0$	a_7	$x_1 - x_3$	a_{12}	$m_3 - m_4$
Y_0	$M_{\mathrm{n}2}$	m_4	$\hat{\mathrm{C}}_4 a_7$	a_{13}	$a_{12} + a_4$
m_1	$\hat{\mathrm{C}}_1 a_2$	a_8	$a_4 + m_2$	Y_1	a_{13}

More in detail, the constants $\hat{\mathrm{C}}_i$, that are present in the algorithm reported in table 3.2, can be obtained as

$$\hat{\mathrm{C}}_i = \mathrm{c}_4 \mathrm{C}_i \qquad \mathrm{c}_4 = \frac{2}{\sqrt{7}}$$

while the constant $\mathrm{C}_{\mathrm{norm}}$ is equal to $\frac{1}{\sqrt{2}}$.

3.4 DCT5 via DCT2 ($N = 8$)

An algorithm for the 8-point DCT5 can be derived from algorithms for the 15-point DCT2. More in detail, the computation of an 8-point DCT5 can be translated into the computation of an 8-point DCT6 as described in section 3.1 and the 8-point DCT6 can be derived from the 15-point DCT2 as illustrated in section 3.2. The following paragraphs will present the steps involved in the derivation of the algorithm.

3.4.1 Reordering of the input vector

According to what is described in section 3.1, the input vector $\boldsymbol{x}$ is reordered so that the vector

$$\boldsymbol{x}_{\mathbf{r}} = [J_8]\, \boldsymbol{x} = \begin{bmatrix} x_7 & x_6 & x_5 & x_4 & x_3 & x_2 & x_1 & x_0 \end{bmatrix}^{\mathrm{T}}$$

is obtained.

3.4.2 Definition of the vector $\boldsymbol{x}_\mathbf{R}$

According to what is described in section 3.2, the vector $\boldsymbol{x}_\mathbf{R}$ is obtained as

$$\boldsymbol{x}_\mathbf{R} = \begin{bmatrix} [I_7] & & [J_7] \\ & 1 & \\ -[J_7] & & [I_7] \end{bmatrix}^{-1} \begin{bmatrix} x_7 \\ x_6 \\ x_5 \\ x_4 \\ x_3 \\ x_2 \\ x_1 \\ x_0 \\ -x_1 \\ -x_2 \\ -x_3 \\ -x_4 \\ -x_5 \\ -x_6 \\ -x_7 \end{bmatrix} = \begin{bmatrix} x_7 \\ 0 \\ x_5 \\ 0 \\ x_3 \\ 0 \\ x_1 \\ x_0 \\ 0 \\ x_2 \\ 0 \\ x_4 \\ 0 \\ x_6 \\ 0 \end{bmatrix}$$

3.4.3 Computation of the 15-point DCT2

The vector $\boldsymbol{Y}^\mathbf{II}$ is derived by computing the DCT2 of the vector $\boldsymbol{x}_\mathbf{R}$. A possible algorithm for the 15-point DCT2 is the Prime Factor Algorithm illustrated in [16] and [18].

More in detail, since the length of the DCT2 is $N_{\text{DCT2}} = 15$ and can be expressed as a product of two numbers ($N1$,$N2$) that are coprime, the PFA can be adopted to solve this DCT2. In particular, the DCT2 can be computed according to the following steps:

1. index transform of the input sequence
2. computation of $N1$ DCTs of length $N2$
3. computation of $N2$ DCTs of length $N1$
4. computation of the output additions and index transform of the output sequence

In the following, each of these steps will be described in detail, supposing that $N1 = 5$ and $N2 = 3$ is chosen. The same procedure can be followed to derive a different algorithm if $N1 = 3$ and $N2 = 5$ is chosen.

Input mapping

The input vector is mapped into the matrix $[z]$ according to the following procedure. Let n_1, $0 \leq n_1 < N1$, and n_2, $0 \leq n_2 < N2$ be the 2D array indices, and n, $0 \leq n < N_{\text{DCT2}}$, be the index of the original input sequence. In general, the mapping process can be performed according to the following steps. For each n, $0 \leq n < N_{\text{DCT2}}$:

1. starting from 0, the index n_1 of the 2D array is incremented by one until $N1 - 1$, then from $N1-1$, index n_1 is decremented by one until to 0; this process is repeated.

2. index n_2 is generated in the same way as n_1 except the range is from 0 to $N2-1$ for an increment and from $N2-1$ to 0 for a decrement.

Therefore, each column in table 3.3 specifies the relation between (n_1,n_2) and n.

Table 3.3: *Mapping of 1D input sequence into 2D array*

Index name	*Index values*														
n_1	0	1	2	3	4	4	3	2	1	0	0	1	2	3	4
n_2	0	1	2	2	1	0	0	1	2	2	1	0	0	1	2
n	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

Hence, the matrix z can be written as:

$$[z] = \begin{bmatrix} x_{\mathrm{R0}} & x_{\mathrm{R10}} & x_{\mathrm{R9}} \\ x_{\mathrm{R11}} & x_{\mathrm{R1}} & x_{\mathrm{R8}} \\ x_{\mathrm{R12}} & x_{\mathrm{R7}} & x_{\mathrm{R2}} \\ x_{\mathrm{R6}} & x_{\mathrm{R13}} & x_{\mathrm{R3}} \\ x_{\mathrm{R5}} & x_{\mathrm{R4}} & x_{\mathrm{R14}} \end{bmatrix}$$

Computation of $N1$ DCTs of length $N2$

As depicted in figure 3.2, each row of the matrix $[z]$ represent the input vector of a DCT2 of length $N2$.

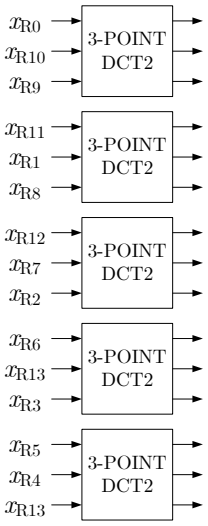


Figure 3.2: *First stage of the 15-point DCT2 PFA*

Computation of $N2$ DCTs of length $N1$

As depicted in figure 3.3, the outputs of the $N1$ DCTs of length $N2$ are used as inputs for the $N2$ DCTs of length $N1$.

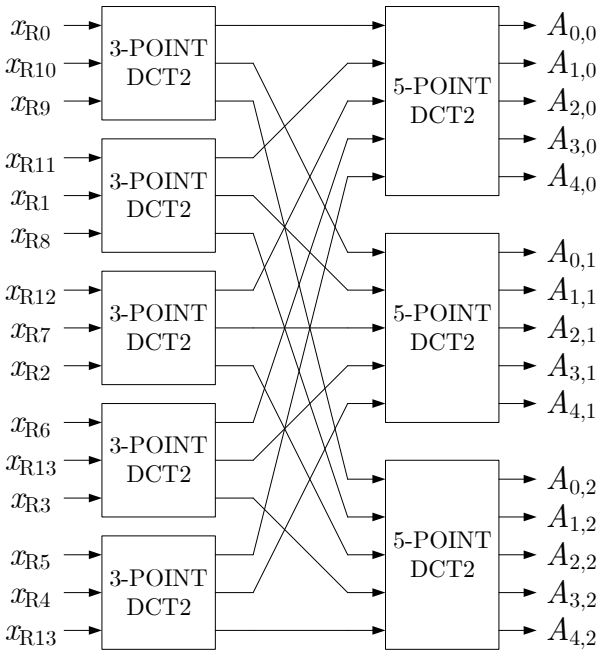


Figure 3.3: *First and second stage of the 15-point DCT2 PFA*

Output additions and index transform of the output sequence

The vector $\boldsymbol{Y}^{\mathbf{II}}$ can be obtained according to the following equation:

$$Y^{\mathrm{II}}_{|N2\mu k+N1m|} = A_{k,m} - B_{\mu k,m} \qquad \begin{cases} 0 \leq k < N1 \\ 0 \leq m < N2 \end{cases}$$

where

$$\begin{cases} B_{k,0} = 0 \\ B_{\mu k,m} = \mu A_{N1-k,N2-m} \end{cases}$$

and

$$\mu = \begin{cases} 1 & \text{if } N2k + N1m < N_{\mathrm{DCT2}} \\ -1 & \text{if } N2k + N1m > N_{\mathrm{DCT2}} \end{cases}$$

Hence, for $N1 = 5$ and $N2 = 3$, the equation is:

$$Y^{\mathrm{II}}_{|3\mu k+5m|} = A_{k,m} - B_{\mu k,m} \qquad \begin{cases} 0 \leq k < 5 \\ 0 \leq m < 3 \end{cases}$$

where

$$\begin{cases} B_{k,0} = 0 \\ B_{\mu k,m} = \mu A_{5-k,3-m} \end{cases}$$

and

$$\mu = \begin{cases} 1 & \text{if } 3k + 5m < 15 \\ -1 & \text{if } 3k + 5m > 15 \end{cases}$$

The equations reported in table 3.4 can therefore be derived.

Table 3.4: *Output additions and index transform of the output sequence for* $N1 = 5$ *and* $N2 = 3$

(m,k)	$3k + 5m$	μ	$Y^{\mathrm{II}}_{\lvert 3\mu k+5m\rvert}$	$A_{k,m} - B_{\mu k,m}$
(0,0)	0	1	Y^{II}_{0}	$A_{0,0}$
(0,1)	3	1	Y^{II}_{3}	$A_{1,0}$
(0,2)	6	1	Y^{II}_{6}	$A_{2,0}$
(0,3)	9	1	Y^{II}_{9}	$A_{3,0}$
(0,4)	12	1	Y^{II}_{12}	$A_{4,0}$
(1,0)	5	1	Y^{II}_{5}	$A_{0,1}$
(1,1)	8	1	Y^{II}_{8}	$A_{1,1} - A_{4,2}$
(1,2)	11	1	Y^{II}_{11}	$A_{2,1} - A_{3,2}$
(1,3)	14	1	Y^{II}_{14}	$A_{3,1} - A_{2,2}$
(1,4)	17	−1	Y^{II}_{7}	$A_{4,1} + A_{1,2}$
(2,0)	10	1	Y^{II}_{10}	$A_{0,2}$
(2,1)	13	1	Y^{II}_{13}	$A_{1,2} - A_{4,1}$
(2,2)	16	−1	Y^{II}_{4}	$A_{2,2} + A_{3,1}$
(2,3)	19	−1	Y^{II}_{1}	$A_{3,2} + A_{2,1}$
(2,4)	22	−1	Y^{II}_{2}	$A_{4,2} + A_{1,1}$

Hence, the scheme depicted in figure 3.4 can be obtained from the description reported above.

Derivation of the output vector of the 8-point DCT6

The computation of the DCT2 leads to the vector:

$$\boldsymbol{Y}^{\mathbf{II}} = \begin{bmatrix} Y^{\mathrm{II}}_{0} & U & Y^{\mathrm{II}}_{2} & U & Y^{\mathrm{II}}_{4} & U & Y^{\mathrm{II}}_{6} & 0 & Y^{\mathrm{II}}_{8} & U & Y^{\mathrm{II}}_{10} & U & Y^{\mathrm{II}}_{12} & U & Y^{\mathrm{II}}_{14} \end{bmatrix}^{\mathrm{T}}$$

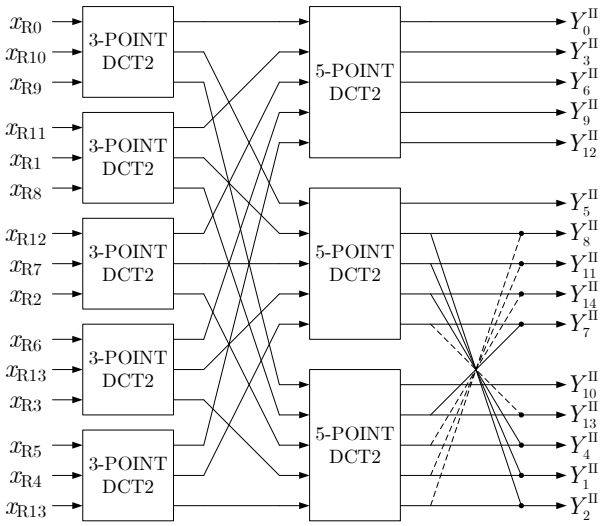


Figure 3.4: *15-point DCT2 PFA*

where the elements labeled as U are not of interest for the computation of the 8-point DCT5. More in detail, the output vector produced by the computation of the DCT6 is

$$\boldsymbol{Y}^{\mathbf{VI}} = \begin{bmatrix} Y_0^{\mathrm{VI}} \\ Y_1^{\mathrm{VI}} \\ Y_2^{\mathrm{VI}} \\ Y_3^{\mathrm{VI}} \\ Y_4^{\mathrm{VI}} \\ Y_5^{\mathrm{VI}} \\ Y_6^{\mathrm{VI}} \\ Y_7^{\mathrm{VI}} \end{bmatrix} = \begin{bmatrix} Y_0^{\mathrm{II}} \\ Y_2^{\mathrm{II}} \\ Y_4^{\mathrm{II}} \\ Y_6^{\mathrm{II}} \\ Y_8^{\mathrm{II}} \\ Y_{10}^{\mathrm{II}} \\ Y_{12}^{\mathrm{II}} \\ Y_{14}^{\mathrm{II}} \end{bmatrix}$$

3.4.4 Definition of the output vector $\boldsymbol{Y}$

According to what is described in section 3.1, the output vector $\boldsymbol{Y}$ can be obtained from the vector $\boldsymbol{Y}^{\mathbf{VI}}$ as follows:

$$\boldsymbol{Y} = [D_4]\,\boldsymbol{Y}^{\mathbf{VI}} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} Y_0^{\mathrm{VI}} \\ Y_1^{\mathrm{VI}} \\ Y_2^{\mathrm{VI}} \\ Y_3^{\mathrm{VI}} \\ Y_4^{\mathrm{VI}} \\ Y_5^{\mathrm{VI}} \\ Y_6^{\mathrm{VI}} \\ Y_7^{\mathrm{VI}} \end{bmatrix} = \begin{bmatrix} Y_0^{\mathrm{VI}} \\ -Y_1^{\mathrm{VI}} \\ Y_2^{\mathrm{VI}} \\ -Y_3^{\mathrm{VI}} \\ Y_4^{\mathrm{VI}} \\ -Y_5^{\mathrm{VI}} \\ Y_6^{\mathrm{VI}} \\ -Y_7^{\mathrm{VI}} \end{bmatrix}$$

3.4.5 Final Algorithm

The SFG of the 8-point DCT5 obtained from the SFG of the 15-point DCT2 is reported in figure 3.5 while the values of the constants and the list of the operations needed are respectively reported in tables 3.5 and 3.6.

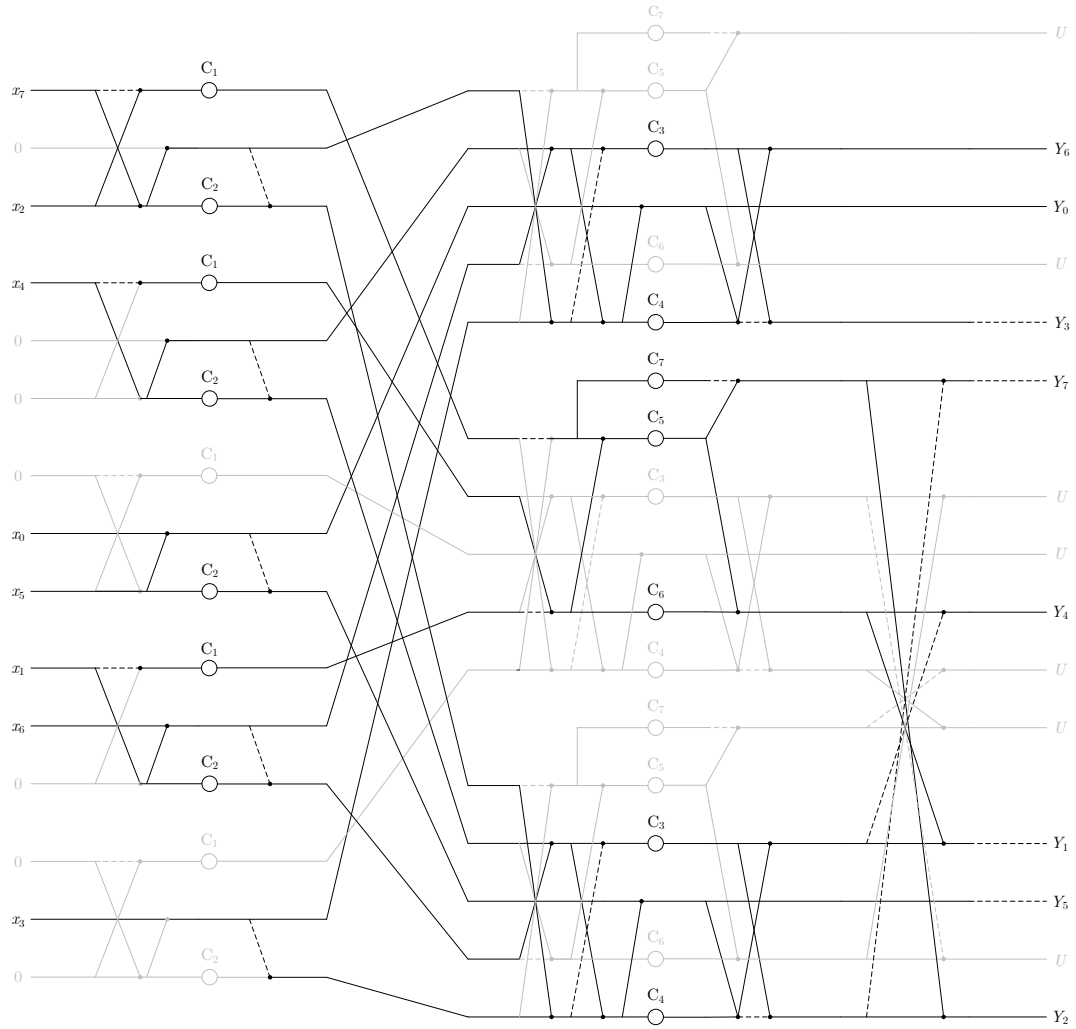


Figure 3.5: *Non-normalized 8-point DCT5 SFG derived from the 15-point DCT2 SFG*

Table 3.5: *Constants for the 8-point DCT5 algorithm derived from the 15-point DCT2 algorithm*

Constant	*Value*
C_1	$-0.866\,025$
C_2	$1.500\,000$

Constant	*Value*
C_3	$-0.559\,017$
C_4	$-1.250\,000$

Constant	*Value*
C_5	$-0.951\,057$
C_6	$1.538\,842$
C_7	$-0.363\,271$

Table 3.6: *Algorithm for the 8-point DCT5 derived from the 15-point DCT2*

Op. Name	*Operation*
M_{n1}	$\mathrm{C}_{\mathrm{norm}} x_0$
a_1	$x_2 - x_7$
a_2	$x_2 + x_7$
m_1	$\mathrm{C}_1 a_1$
m_2	$\mathrm{C}_2 a_2$
a_3	$m_2 - a_2$
m_3	$-\mathrm{C}_1 x_4$
m_4	$\mathrm{C}_2 x_4$
a_4	$m_4 - x_4$
a_5	$M_{\mathrm{n1}} + x_5$
m_5	$\mathrm{C}_2 x_5$
a_6	$m_5 - a_5$
m_6	$-\mathrm{C}_1 x_1$
a_7	$x_1 + x_6$
m_7	$\mathrm{C}_2 x_1$
a_8	$m_7 - a_7$
a_9	$x_4 + a_7$
a_{10}	$a_2 + x_3$
a_{11}	$a_9 - a_{10}$

Op. Name	*Operation*
a_{12}	$a_9 + a_{10}$
a_{13}	$a_{12} + a_5$
m_8	$\mathrm{c}_8 a_{13}$
m_9	$\hat{\mathrm{C}}_3 a_{11}$
m_{10}	$\hat{\mathrm{C}}_4 a_{12}$
a_{14}	$m_{10} + m_8$
a_{15}	$m_9 + a_{14}$
a_{16}	$m_9 - a_{14}$
Y_6	a_{15}
M_{n2}	$\mathrm{C}_{\mathrm{norm}} m_8$
Y_0	M_{n2}
Y_3	$-a_{16}$
a_{17}	$m_3 - m_6$
a_{18}	$a_{17} - m_1$
m_{11}	$-m_1 \hat{\mathrm{C}}_7$
m_{12}	$a_{18} \hat{\mathrm{C}}_5$
m_{13}	$a_{17} \hat{\mathrm{C}}_6$
a_{19}	$m_{12} - m_{11}$
a_{20}	$m_{12} + m_{13}$

Op. Name	*Operation*
a_{21}	$a_4 + a_8$
a_{22}	$a_3 - x_3$
a_{23}	$a_{21} - a_{22}$
a_{24}	$a_{21} + a_{22}$
a_{25}	$a_{24} + a_6$
m_{14}	$\mathrm{c}_8 a_{25}$
m_{15}	$\hat{\mathrm{C}}_3 a_{23}$
m_{16}	$\hat{\mathrm{C}}_4 a_{24}$
a_{26}	$m_{14} + m_{16}$
a_{27}	$m_{15} + a_{26}$
a_{28}	$m_{15} - a_{26}$
a_{29}	$a_{19} - a_{28}$
Y_7	$-a_{29}$
a_{30}	$a_{20} - a_{27}$
Y_4	a_{30}
a_{31}	$a_{20} + a_{27}$
Y_1	$-a_{31}$
a_{32}	$a_{19} + a_{28}$
Y_2	a_{32}
Y_5	$-m_{14}$

More in detail, the constants $\hat{\mathrm{C}}_i$, that are present in the algorithm reported in table 3.6, can be obtained as

$$\hat{\mathrm{C}}_i = \mathrm{c}_8 \mathrm{C}_i \qquad \mathrm{c}_8 = \frac{2}{\sqrt{15}}$$

while the constant $\mathrm{C}_{\mathrm{norm}}$ is equal to $\frac{1}{\sqrt{2}}$.

Chapter 4

DCT5 via Givens Rotations

This chapter presents a possible algorithm that can be adopted to compute the DCT5 by using only rotations. The algorithm, which will be developed in the following, does not reach levels of computational complexity as low as the ones achieved by the WFTA or the PFA. Nevertheless, it is worth analyzing it since it minimizes the variety of required processing units [19].

4.1 Givens Rotations

Any matrix [A] can be factored so that

$$[A] = [Q][R] \tag{4.1}$$

where [Q] is an orthogonal matrix and [R] is an upper triangular matrix. Moreover, if [A] is orthogonal then [R] is a diagonal matrix so that

$$[R]^{\mathrm{T}}[R] = [I]$$

Hence, the diagonal elements of [R] can only be +1 or -1. Furthermore, the columns of [A] and [Q] are identical except for a possible minus sign.

Such a decomposition of the matrix [A] is called QR-decomposition and can be achieved by using the so-called Givens rotations. The essential point of this method is to null the lower off-diagonal elements of $[A]$ by performing a rotation of the right angle. This is better illustrated by an example. Consider, for instance, the vector

$$\boldsymbol{b} = \begin{bmatrix} b_0 \\ b_1 \end{bmatrix}$$

A matrix $[T(\theta)]$ exists so that

$$[T(\theta)]\boldsymbol{b} = \begin{bmatrix} \hat{b}_0 \\ 0 \end{bmatrix} \tag{4.2}$$

If

$$[T(\theta)] = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix}$$

then

$$\cos(\theta) = \frac{b_0}{\sqrt{b_0^2 + b_1^2}} \qquad \sin(\theta) = \frac{b_1}{\sqrt{b_0^2 + b_1^2}}$$

This procedure can also be followed to null the ij-element of the orthogonal matrix $[A]$. Hence, to null the ij-element of $[A]$, the matrix $[T_{i,j}(\theta)]$ is needed. The angle θ is such that

$$\cos(\theta) = \frac{A_{i-1,j}}{\sqrt{A_{i-1,j}^2 + A_{i,j}^2}} \qquad \sin(\theta) = \frac{A_{i,j}}{\sqrt{A_{i-1,j}^2 + A_{i,j}^2}}$$

and $[T_{i,j}(\theta)]$ is an identity matrix except for the elements in positions $(i-1,\, i-1)$, $(i-1,\, i)$, $(i,\, i-1)$ and $(i,\, i)$ where $\cos(\theta)$, $\sin(\theta)$, $-\sin(\theta)$ and $\cos(\theta)$ are respectively located.

By systematically applying this type of rotations, the matrix $[A]$ is reduced to a diagonal matrix with $+1$ or -1 as diagonal elements. The sequence of rotations on the matrix $[A]$ can be represented by

$$\left\{ \prod_{j=1}^{N} \prod_{i=j+1}^{N} [T_{i,j}(\theta)] \right\} [A] = [R]$$

Hence, it can be written

$$[A] = \left\{ \prod_{j=1}^{N} \prod_{i=j+1}^{N} [T_{i,j}(\theta)] \right\}^{\mathrm{T}} [R]$$

If we compare this expression to equation 4.1, we can notice that

$$[Q] = \left\{ \prod_{j=1}^{N} \prod_{i=j+1}^{N} [T_{i,j}(\theta)] \right\}^{\mathrm{T}} = \left\{ \prod_{i=1}^{N} \prod_{j=i+1}^{N} [T_{i,j}(\theta)] \right\}$$

We have therefore obtained a decomposition of the matrix $[A]$ into planar rotations.

4.2 DCT5 via Givens Rotations ($N = 4$)

Since $\left[C_4^{\mathrm{V}}\right]$ is an orthonormal matrix, it can be factored into a sequence of planar rotations. More in detail, the matrix is

$$\left[C_4^{\mathrm{V}}\right] = \begin{bmatrix} 0.3780 & 0.5345 & 0.5345 & 0.5345 \\ 0.5345 & 0.4713 & -0.1682 & -0.6811 \\ 0.5345 & -0.1682 & -0.6811 & 0.4713 \\ 0.5345 & -0.6811 & 0.4713 & -0.1682 \end{bmatrix}$$

We can start by nulling the element in position (3, 0). Hence, we can define

$$\cos(\theta) = \frac{0.5345}{\sqrt{0.5345^2 + 0.5345^2}} = 0.7071 \qquad \sin(\theta) = \frac{0.5345}{\sqrt{0.5345^2 + 0.5345^2}} = 0.7071$$

and derive the rotation matrix as

$$[T_{3,0}(\theta)] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \cos(\theta) & \sin(\theta) \\ 0 & 0 & -\sin(\theta) & \cos(\theta) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0.7071 & 0.7071 \\ 0 & 0 & -0.7071 & 0.7071 \end{bmatrix}$$

By performing the matrix product

$$[C_1] = [T_{3,0}(\theta)] \left[C_4^{\mathrm{V}}\right] = \begin{bmatrix} 0.3780 & 0.5345 & 0.5345 & 0.5345 \\ 0.5345 & 0.4713 & -0.1682 & -0.6811 \\ 0.7559 & -0.6005 & -0.1483 & 0.2143 \\ 0 & -0.3626 & 0.8149 & -0.4522 \end{bmatrix}$$

we obtain the matrix whose element in position (2, 0) has to be nulled. Thus, we redefine

$$\cos(\theta) = \frac{0.5345}{\sqrt{0.5345^2 + 0.7559^2}} = 0.5774 \qquad \sin(\theta) = \frac{0.7559}{\sqrt{0.5345^2 + 0.7559^2}} = 0.8165$$

and build the second rotation matrix as

$$[T_{2,0}(\theta)] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\theta) & \sin(\theta) & 0 \\ 0 & -\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0.5774 & 0.8165 & 0 \\ 0 & -0.8165 & 0.5774 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

We can therefore perform the matrix product

$$[C_2] = [T_{2,0}(\theta)] [C_1] = \begin{bmatrix} 0.3780 & 0.5345 & 0.5345 & 0.5345 \\ 0.9258 & -0.2182 & -0.2182 & -0.2182 \\ 0 & -0.7315 & 0.0517 & 0.6798 \\ 0 & -0.3626 & 0.8149 & -0.4522 \end{bmatrix}$$

and obtain the matrix whose element in position (1, 0) has to be nulled. Hence, we write

$$\cos(\theta) = \frac{0.3780}{\sqrt{0.3780^2 + 0.9258^2}} = 0.3780 \qquad \sin(\theta) = \frac{0.9258}{\sqrt{0.3780^2 + 0.9258^2}} = 0.9258$$

and construct the third rotation matrix as

$$[T_{1,0}(\theta)] = \begin{bmatrix} \cos(\theta) & \sin(\theta) & 0 & 0 \\ -\sin(\theta) & \cos(\theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0.3780 & 0.9258 & 0 & 0 \\ -0.9258 & 0.3780 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

We can now perform the matrix product

$$[C_3] = [T_{1,0}(\theta)]\,[C_2] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -0.5774 & -0.5774 & -0.5774 \\ 0 & -0.7315 & 0.0517 & 0.6798 \\ 0 & -0.3626 & 0.8149 & -0.4522 \end{bmatrix}$$

and obtain the matrix whose element in position (3, 1) has to be nulled. Therefore, we impose

$$\cos(\theta) = \frac{-0.7315}{\sqrt{(-0.7315)^2 + (-0.3626)^2}} = -0.8960$$
$$\sin(\theta) = \frac{-0.3626}{\sqrt{(-0.7315)^2 + (-0.3626)^2}} = -0.4441$$

and get the fourth rotation matrix as

$$[T_{3,1}(\theta)] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \cos(\theta) & \sin(\theta) \\ 0 & 0 & -\sin(\theta) & \cos(\theta) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -0.8960 & -0.4441 \\ 0 & 0 & 0.4441 & -0.8960 \end{bmatrix}$$

Hence, we perform the matrix product

$$[C_4] = [T_{3,1}(\theta)]\,[C_3] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -0.5774 & -0.5774 & -0.5774 \\ 0 & 0.8165 & -0.4082 & -0.4082 \\ 0 & 0 & -0.7071 & 0.7071 \end{bmatrix}$$

and derive the matrix whose element in position (2, 1) has to be nulled. Thus, we redefine

$$\cos(\theta) = \frac{-0.5774}{\sqrt{(-0.5774)^2 + 0.8165^2}} = -0.5774$$
$$\sin(\theta) = \frac{0.8165}{\sqrt{(-0.5774)^2 + 0.8165^2}} = 0.8165$$

and build the fifth rotation matrix as

$$[T_{2,1}(\theta)] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\theta) & \sin(\theta) & 0 \\ 0 & -\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -0.5774 & 0.8165 & 0 \\ 0 & -0.8165 & -0.5774 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Hence, we can perform the matrix product

$$[C_5] = [T_{2,1}(\theta)]\,[C_4] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0.7071 & 0.7071 \\ 0 & 0 & -0.7071 & 0.7071 \end{bmatrix}$$

and obtain the matrix whose element in position (3, 2) has to be nulled. Thus, we set

$$\cos(\theta) = \frac{0.7071}{\sqrt{0.7071^2 + (-0.7071)^2}} = 0.7071$$
$$\sin(\theta) = \frac{-0.7071}{\sqrt{0.7071^2 + (-0.7071)^2}} = -0.7071$$

and construct the last rotation matrix as

$$[T_{3,2}(\theta)] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \cos(\theta) & \sin(\theta) \\ 0 & 0 & -\sin(\theta) & \cos(\theta) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0.7071 & -0.7071 \\ 0 & 0 & 0.7071 & 0.7071 \end{bmatrix}$$

The matrix $[R]$ is then obtained by performing the matrix product

$$[R] = [T_{3,2}(\theta)]\,[C_5] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = [I_4]$$

Hence, we have

$$[T_{3,2}]\,[T_{2,1}]\,[T_{3,1}]\,[T_{1,0}]\,[T_{2,0}]\,[T_{3,0}]\,\left[C_4^{\mathrm{V}}\right] = [I_4]$$

As a consequence, we can write

$$\begin{aligned} \left[C_4^{\mathrm{V}}\right] &= [T_{3,0}]^{\mathrm{T}}\,[T_{2,0}]^{\mathrm{T}}\,[T_{1,0}]^{\mathrm{T}}\,[T_{3,1}]^{\mathrm{T}}\,[T_{2,1}]^{\mathrm{T}}\,[T_{3,2}]^{\mathrm{T}} \\ &= \left[\hat{T}_{3,0}\right]\left[\hat{T}_{2,0}\right]\left[\hat{T}_{1,0}\right]\left[\hat{T}_{3,1}\right]\left[\hat{T}_{2,1}\right]\left[\hat{T}_{3,2}\right] \end{aligned}$$

We have therefore found the factorization of $\left[C_4^V\right]$ into planar rotations. In particular, the needed rotation matrices are:

$$\left[\hat{T}_{3,0}\right] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0.7071 & -0.7071 \\ 0 & 0 & 0.7071 & 0.7071 \end{bmatrix} \qquad \left[\hat{T}_{2,0}\right] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0.5774 & -0.8165 & 0 \\ 0 & 0.8165 & 0.5774 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\left[\hat{T}_{1,0}\right] = \begin{bmatrix} 0.3780 & -0.9258 & 0 & 0 \\ 0.9258 & 0.3780 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \left[\hat{T}_{3,1}\right] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -0.8960 & 0.4441 \\ 0 & 0 & -0.4441 & -0.8960 \end{bmatrix}$$

$$\left[\hat{T}_{2,1}\right] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -0.5774 & -0.8165 & 0 \\ 0 & 0.8165 & -0.5774 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \left[\hat{T}_{3,2}\right] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0.7071 & 0.7071 \\ 0 & 0 & -0.7071 & 0.7071 \end{bmatrix}$$

It should be highlighted that six rotations are required to compute a 4-point DCT5. In general, an N-point DCT5 needs $\frac{N^2}{2} - \frac{N}{2}$ rotations to be computed. Moreover, each rotation can be performed by means of four multiplications and two additions or three multiplications and three additions.

Chapter 5

DCT5 via Direct Factorization

The aim of this chapter is to present algorithms based on a direct factorization of the DCT5. In particular, the direct factorization obtained in [20] will be firstly analyzed. Hence, by making use of this result, an algorithm will be derived for the 8-point DCT5. Moreover, an algorithm for the 32-point DCT5 will be presented in appendix C by referring to the results reported in [13], [21] and [14].

5.1 Direct Factorization of the DCT5

The factorization proposed in [20] is the following one:

$$\left[C^{\mathrm{V}}_{3m+2}\right] = \left[Q^{3m+2}_{m}\right] \left(\left[C^{\mathrm{V}}_{m+1}\right] \oplus \left[C^{\mathrm{III}}_{2m+1} \left(\frac{2}{3}\right) \right] \right) \left[B^{(C5)}_{3m+2}\right] \tag{5.1}$$

where

- $\left[C^{\mathrm{V}}_{3m+2}\right]$ is the non-normalized $(3m+2)$-point DCT5 matrix;
- $\left[Q^{3m+2}_{m}\right]$ is a permutation matrix;
- $\left[C^{\mathrm{V}}_{m+1}\right]$ is the non-normalized $(m+1)$-point DCT5 matrix;
- $\oplus$ is the direct sum operator;
- $\left[C^{\mathrm{III}}_{2m+1}\left(\frac{2}{3}\right)\right]$ is a non-normalized $(2m+1)$-point skew-DCT3 matrix;
- $\left[B^{(C5)}_{3m+2}\right]$ is a base change matrix.

In the following, each of the elements present in the equation will be analyzed.

5.1.1 Permutation matrix

The permutation matrix $\left[Q_m^{3m+2}\right]$ is such that:

$$\left[Q_m^{3m+2}\right] = i_1 + 3i_2 \mapsto \begin{cases} i_2 & \text{for } i_1 = 0 \\ 2i_2 + m + 1 & \text{for } i_1 = 1 \\ 2i_2 + m + 2 & \text{for } i_1 = 2 \end{cases}$$

This means that the row whose index is $i_1 + 3i_2$ has a one in position:

- $i_2 \quad \text{if } i_1 = 0$
- $2i_2 + m + 1 \quad \text{if } i_1 = 1$
- $2i_2 + m + 2 \quad \text{if } i_1 = 2$

We can, for instance, consider $N = 3m + 2 = 8$. The position of the ones of the permutation matrix is reported in table 5.1.

Table 5.1: *Rows and columns of the ones of the permutation matrix for $N = 8$*

Row	(i_1,i_2)	*Column computation*	*Column*
$0 = 0 + 3 \times 0$	(0,0)	$i_2 = 0$	0
$1 = 1 + 3 \times 0$	(1,0)	$2 \times 0 + 2 + 1$	3
$2 = 2 + 3 \times 0$	(2,0)	$2 \times 0 + 2 + 2$	4
$0 = 0 + 3 \times 1$	(0,1)	$i_2 = 1$	1
$4 = 1 + 3 \times 1$	(1,1)	$2 \times 1 + 2 + 1$	5
$5 = 2 + 3 \times 1$	(2,1)	$2 \times 1 + 2 + 2$	6
$6 = 0 + 3 \times 2$	(0,2)	$i_2 = 2$	2
$7 = 1 + 3 \times 2$	(1,2)	$2 \times 2 + 2 + 1$	7

Hence, for this particular case, the permutation matrix is

$$\left[Q_2^8\right] = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

5.1.2 Non-normalized DCT5 matrix

The non-normalized DCT5 matrix is equal to the normalized DCT5 matrix except for the following facts:

- the multiplicative constant equal to $\frac{2}{\sqrt{2N-1}}$ in the normalized form is equal to one in the non-normalized matrix;

- all the elements of the column 0 are equal to one in the non-normalized form;
- all the elements of the row 0 are equal to one in the non-normalized form.

Hence, the non-normalized DCT5 matrix is:

$$\left[C_N^{\mathrm{V}}\right] = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \cos\left(\frac{2\pi}{2N-1}\right) & \cos\left(\frac{4\pi}{2N-1}\right) & \cdots & \cos\left((N-1)\frac{2\pi}{2N-1}\right) \\ 1 & \cos\left(\frac{4\pi}{2N-1}\right) & \cos\left(\frac{8\pi}{2N-1}\right) & \cdots & \cos\left(2(N-1)\frac{2\pi}{2N-1}\right) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \cos\left((N-1)\frac{2\pi}{2N-1}\right) & \cos\left(2(N-1)\frac{2\pi}{2N-1}\right) & \cdots & \cos\left((N-1)^2\frac{2\pi}{2N-1}\right) \end{bmatrix}$$

5.1.3 Direct sum operator

The direct sum operator is such that, given the matrices $[A]$ and $[B]$:

$$[A] \oplus [B] = \begin{bmatrix} [A] & \\ & [B] \end{bmatrix}$$

5.1.4 Non-normalized skew-DCT3 matrix

The non-normalized skew-DCT3 matrix can be obtained from the DCT3 matrix as follows:

$$\left[C^{\mathrm{III}}_{N_{\mathrm{DCT3}}}(r)\right] = \left[C^{\mathrm{III}}_{N_{\mathrm{DCT3}}}\right]\left[X^{(C3)}_{N_{\mathrm{DCT3}}}(r)\right]$$

where:

- $\left[C^{\mathrm{III}}_{N_{\mathrm{DCT3}}}\right]$ is the non-normalized $N_{\mathrm{DCT3}} \times N_{\mathrm{DCT3}}$ DCT3 matrix. More in detail, the non-normalized DCT3 of a given sequence $\{x_k\}$ of length N_{DCT3} is defined as:

$$Y_n^{\mathrm{III}} = \sum_{k=0}^{N_{\mathrm{DCT3}}-1} x_k \cos\left((n+1/2)k\frac{\pi}{N_{\mathrm{DCT3}}}\right) \qquad \text{for } n = 0,1,\ldots,N_{\mathrm{DCT3}}-1$$

Hence, the non-normalized DCT3 matrix is:

$$\left[C^{\mathrm{III}}_{N_{\mathrm{DCT3}}}\right] = \begin{bmatrix} 1 & \cos\left(\frac{1}{2}\frac{\pi}{N_{\mathrm{DCT3}}}\right) & \ldots & \cos\left(\frac{N_{\mathrm{DCT3}}-1}{2}\frac{\pi}{N_{\mathrm{DCT3}}}\right) \\ 1 & \cos\left(\frac{3}{2}\frac{\pi}{N_{\mathrm{DCT3}}}\right) & \ldots & \cos\left(\frac{3(N_{\mathrm{DCT3}}-1)}{2}\frac{\pi}{N_{\mathrm{DCT3}}}\right) \\ 1 & \cos\left(\frac{5}{2}\frac{\pi}{N_{\mathrm{DCT3}}}\right) & \ldots & \cos\left(\frac{5(N_{\mathrm{DCT3}}-1)}{2}\frac{\pi}{N_{\mathrm{DCT3}}}\right) \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \cos\left(\frac{2N_{\mathrm{DCT3}}-1}{2}\frac{\pi}{N_{\mathrm{DCT3}}}\right) & \ldots & \cos\left(\frac{(2N_{\mathrm{DCT3}}-1)(N_{\mathrm{DCT3}}-1)}{2}\frac{\pi}{N_{\mathrm{DCT3}}}\right) \end{bmatrix}$$

- $\left[X^{(C3)}_{N_{\text{DCT3}}}(r)\right]$ is a matrix defined as follows:

$$\left[X^{(C3)}_{N_{\text{DCT3}}}(r)\right] = \begin{bmatrix} 1 & 0 & \dots & \dots & 0 \\ 0 & c_1 & & & s_{N_{\text{DCT3}}-1} \\ \vdots & & \ddots & \iddots & \\ \vdots & & \iddots & \ddots & \\ 0 & s_1 & & & c_{N_{\text{DCT3}}-1} \end{bmatrix}$$

  where:
  - $c_l = \cos\left(\frac{(1/2-r)l\pi}{N_{\text{DCT3}}}\right)$
  - $s_l = \sin\left(\frac{(1/2-r)l\pi}{N_{\text{DCT3}}}\right)$

5.1.5 Base change matrix

The base change matrix is defined as:

$$\left[B^{(C5)}_{3m+2}\right] = \left[\begin{array}{ccc|cc} 1 & & & & 1 \\ & [I_m] & [J_m] & & & [I_m] \\ \hline & & & & -1/2 & \\ & [I_{2m+1}] & & & & -[I_m] \\ & & & & & -[J_m] \end{array}\right]$$

where

- $[I_n]$ is the n-point identity matrix:

$$[I_n] = \begin{bmatrix} 1 & 0 & \dots & 0 & 0 \\ 0 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 1 & 0 \\ 0 & 0 & \dots & 0 & 1 \end{bmatrix}$$

- $[J_n]$ is the n-point backward identity matrix:

$$[J_n] = \begin{bmatrix} 0 & 0 & \dots & 0 & 1 \\ 0 & 0 & \dots & 1 & 0 \\ \vdots & \vdots & \iddots & \vdots & \vdots \\ 0 & 1 & \dots & 0 & 0 \\ 1 & 0 & \dots & 0 & 0 \end{bmatrix}$$

5.2 Direct Factorization of the DCT5 ($N = 8$)

The factorization described in section 5.1 can be applied to the 8-point DCT5. More in detail, it can be written:

$$\left[C_8^{\mathrm{V}}\right] = \left[Q_2^8\right] \left(\left[C_3^{\mathrm{V}}\right] \oplus \left[C_5^{\mathrm{III}} \left(\frac{2}{3} \right) \right] \right) \left[B_8^{(C5)}\right]$$

In the following each of the terms present in the right-side of the equation will be described.

5.2.1 Eight-point permutation matrix

The permutation matrix can be obtained by following the procedure illustrated in section 5.1.1. Hence, the matrix is:

$$\left[Q_2^8\right] = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

5.2.2 Three-point non-normalized DCT5 matrix

The 3-point DCT5 matrix is defined as:

$$\left[C_3^{\mathrm{V}}\right] = \begin{bmatrix} 1 & 1 & 1 \\ 1 & \cos\left(\frac{2\pi}{5}\right) & \cos\left(\frac{4\pi}{5}\right) \\ 1 & \cos\left(\frac{4\pi}{5}\right) & \cos\left(\frac{8\pi}{5}\right) \end{bmatrix}$$

According to what is described in the previous chapters, several low-complexity algorithms can be adopted for the computation of the 3-point DCT5. One algorithm is also found in [21]. In the following, the algorithm will be presented.

Low-complexity algorithm for 3-point DCT5

The algorithm is shown in the SFG depicted in figure 5.1. The values of the constants are instead reported in table 5.2. Finally, the operations needed are listed in table 5.3.

Table 5.2: *Constants for the 3-point DCT5 algorithm*

Constant	*Value*
C_1	−0.250 00
C_2	0.559 02

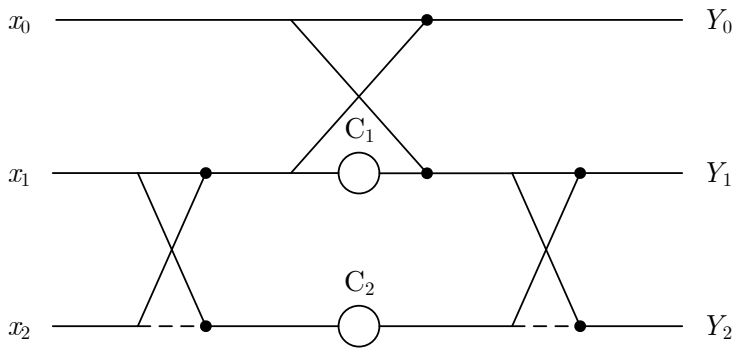


Figure 5.1: *3-Point DCT5 SFG*

Table 5.3: *Algorithm for the 3-point DCT5*

Op. Name	*Operation*
a_0	$x_1 + x_2$
a_1	$x_1 - x_2$
Y_0	$a_0 + x_0$

Op. Name	*Operation*
M_1	$a_0 \mathrm{C}_1$
M_2	$a_1 \mathrm{C}_2$
a_2	$M_1 + x_0$

Op. Name	*Operation*
Y_1	$a_2 + M_2$
Y_2	$a_2 - M_2$

5.2.3 Five-point non-normalized skew-DCT3 matrix

The 5-point non-normalized skew-DCT3 matrix can be obtained as

$$\left[C_5^{\mathrm{III}}\left(\frac{2}{3}\right)\right] = \left[C_5^{\mathrm{III}}\right]\left[X_5^{(C3)}\left(\frac{2}{3}\right)\right]$$

where

- $\left[C_5^{\mathrm{III}}\right] = \begin{bmatrix} 1 & \cos\left(\frac{\pi}{10}\right) & \cos\left(\frac{2\pi}{10}\right) & \cos\left(\frac{3\pi}{10}\right) & \cos\left(\frac{4\pi}{10}\right) \\ 1 & \cos\left(\frac{3\pi}{10}\right) & \cos\left(\frac{6\pi}{10}\right) & \cos\left(\frac{9\pi}{10}\right) & \cos\left(\frac{12\pi}{10}\right) \\ 1 & \cos\left(\frac{5\pi}{10}\right) & \cos\left(\frac{10\pi}{10}\right) & \cos\left(\frac{15\pi}{10}\right) & \cos\left(\frac{20\pi}{10}\right) \\ 1 & \cos\left(\frac{7\pi}{10}\right) & \cos\left(\frac{14\pi}{10}\right) & \cos\left(\frac{21\pi}{10}\right) & \cos\left(\frac{28\pi}{10}\right) \\ 1 & \cos\left(\frac{9\pi}{10}\right) & \cos\left(\frac{18\pi}{10}\right) & \cos\left(\frac{27\pi}{10}\right) & \cos\left(\frac{36\pi}{10}\right) \end{bmatrix}$
- $\left[X_5^{(C3)}\left(\frac{2}{3}\right)\right] = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & \cos\left(-\frac{\pi}{30}\right) & 0 & 0 & \sin\left(-\frac{4\pi}{30}\right) \\ 0 & 0 & \cos\left(-\frac{2\pi}{30}\right) & \sin\left(-\frac{3\pi}{30}\right) & 0 \\ 0 & 0 & \sin\left(-\frac{2\pi}{30}\right) & \cos\left(-\frac{3\pi}{30}\right) & 0 \\ 0 & \sin\left(-\frac{\pi}{30}\right) & 0 & 0 & \cos\left(-\frac{4\pi}{30}\right) \end{bmatrix}$

A possible low-complexity algorithm for the 5-point DCT3 is reported in [21]. In the following, the algorithm will be presented.

Low-complexity algorithm for 5-point DCT3

The algorithm is shown in the SFG depicted in figure 5.2. Moreover, the values of the constants are reported in table 5.4 and the list of operations needed is in table 5.5.

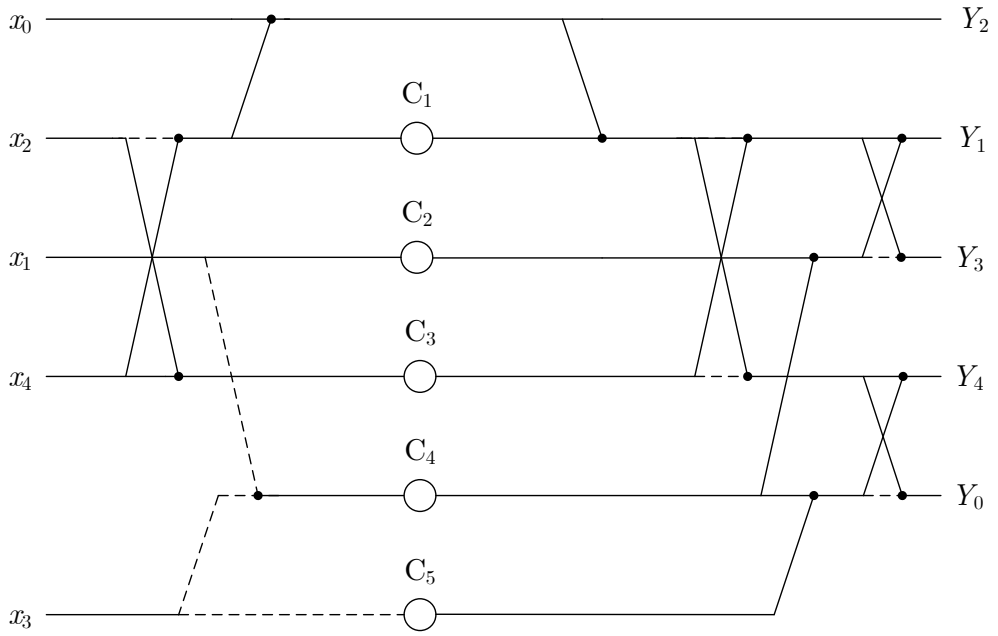


Figure 5.2: *Non-normalized 5-Point DCT3 SFG*

Table 5.4: *Constants for the 5-point DCT3 algorithm*

Constant	*Value*
C_1	$-1.250\,00$
C_2	$1.538\,84$
C_3	$-0.559\,02$
C_4	$0.951\,06$
C_5	$-0.363\,27$

Table 5.5: *Algorithm for the non-normalized 5-point DCT3*

Op. Name	*Operation*	*Op. Name*	*Operation*	*Op. Name*	*Operation*
a_0	$x_4 - x_2$	M_3	$C_3 a_2$	a_7	$M_2 + M_4$
Y_2	$a_0 + x_0$	M_4	$C_4 a_3$	Y_1	$a_5 + a_7$
a_2	$x_2 + x_4$	M_5	$-C_5 x_3$	Y_3	$a_5 - a_7$
a_3	$-x_1 - x_3$	a_4	$M_1 + Y_2$	a_{10}	$M_4 + M_5$
M_1	$C_1 a_0$	a_5	$a_4 + M_3$	Y_4	$a_6 + a_{10}$
M_2	$C_2 x_1$	a_6	$a_4 - M_3$	Y_0	$a_6 - a_{10}$

5.2.4 Eight-point base change matrix

The 8-point base change matrix is

$$\left[B_8^{(C5)}\right] = \left[\begin{array}{ccc|cc} 1 & & & 1 & \\ & [I_2] & [J_2] & & [I_2] \\ \hline & & & -1/2 & \\ & [I_5] & & & -[I_2] \\ & & & & -[J_2] \end{array}\right] = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & -\frac{1}{2} & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 \end{bmatrix}$$

5.2.5 Computational Complexity

The computational complexity is reduced to 15 multiplications and 33 additions.

Chapter 6

Comparison of the algorithms

This chapter presents a comparison of the algorithms considered until now. More in detail, the algorithms will be compared as far as the computational complexity, the regularity and modularity and the number of cascaded multipliers are concerned. The computational complexity is calculated for the "unfolded form"of the algorithms.

6.1 Algorithms for the 4-point DCT5: a comparison

The computational complexity of the algorithms for the 4-point DCT5 is reported in table 6.1.

Table 6.1: *Number of additions and multiplications needed by the algorithms for the normalized 4-point DCT5*

Algorithm	*Number of multiplications*	*Number of additions*
Matrix-Vector Multiplication	16	12
Circular Convolution	6	14
DFT (WFTA)	7	13
DFT (Bluestein+Radix 2 DIT)	128	466
DFT (Rader+PFA)	36	87
DCT2	7	13
Givens Rotations	18	18

The algorithm that requires the lowest number of multiplications is the one derived from the circular convolution (reported in table 2.6 at page 13). The algorithms derived from the WFTA and the DCT2 only need 13 additions but 7 multiplications. Moreover, the algorithm which makes use of the Givens rotations is characterized by a computational complexity higher than the direct implementation of the matrix-vector multiplication. Furthermore, Bluestein's algorithm and Rader's algorithm have high levels of computational

complexity even though they are the only ones which exhibit a particularly regular structure. Finally, it should be considered that Bluestein's algorithm and Rader's algorithm are characterized by the presence of several cascaded multipliers. This is important because the presence of cascaded multipliers determine an increase of the parallelism when the fixed-point algorithm is generated.

6.2 Algorithms for the 8-point DCT5: a comparison

The computational complexity of the algorithms for the 8-point DCT5 is reported in table 6.2.

Table 6.2: *Number of additions and multiplications needed by the algorithms for the normalized 8-point DCT5*

Algorithm	*Number of multiplications*	*Number of additions*
Matrix-Vector Multiplication	64	56
DFT (WFTA)	11	29
DFT (PFA $N1 = 3$, $N2 = 5$)	21	36
DFT (PFA $N1 = 5$, $N2 = 3$)	18	32
DFT (Bluestein+Radix 2 DIT)	344	1210
DCT2 (PFA $N1 = 3$, $N2 = 5$)	21	36
DCT2 (PFA $N1 = 5$, $N2 = 3$)	18	32
Direct Factorization	19	33

The lowest computational complexity is reached by the algorithm derived from the WFTA (presented in table 2.11 at page 22). An increase in the regularity of the structure determine an increase of the number of needed multiplications and additions. This is particularly true for the algorithms derived from PFA and DCT2. Moreover these algorithms are characterized by the presence of cascaded multipliers that cause an increase of the parallelism in the fixed-point implementation. Cascaded multipliers are also present in the algorithm obtained from the direct factorization of the DCT5. Finally, even though Bluestein's algorithm is the one characterized by the highest level of modularity, its computational complexity is too high to make the algorithm appropriate for any reasonable application.

6.3 Algorithms for the 16-point DCT5: a comparison

The computational complexity of the algorithms for the 16-point DCT5 is reported in table 6.3.

The lowest computational complexity is reached by the algorithms described in sections A.1.1 and B.1. Also in this case, even if Bluestein's and Rader's algorithms are characterized by a good level of modularity that, for instance, allows the computation of

Table 6.3: *Number of additions and multiplications needed by the algorithms for the normalized 16-point DCT5*

Algorithm	*Number of multiplications*	*Number of additions*
Matrix-Vector Multiplication	256	240
DFT (Selesnick and Burrus)	43	165
DFT (Bluestein+Radix 2 DIT)	872	2986
DFT (Rader+PFA+WFTA)	236	904
DCT2 (Spiral)	72	144

the 8-point DCT5, their computational complexity is too high to take them into consideration. Finally, it should be underlined that the algorithm described in section A.1.1 is the only one that is not characterized by the presence of cascaded multipliers (except for the ones needed for the pre-and post-normalization).

6.4 Algorithms for the 32-point DCT5: a comparison

The computational complexity of the algorithms for the 32-point DCT5 is reported in table 6.4.

Table 6.4: *Number of additions and multiplications needed by the algorithms for the normalized 32-point DCT5*

Algorithm	*Number of multiplications*	*Number of additions*
Matrix-Vector Multiplication	1024	992
DFT (WFTA)	52	304
DFT (PFA $N1 = 9$, $N2 = 7$)	105	330
DFT (PFA $N1 = 7$, $N2 = 9$)	99	326
DFT (Bluestein+Radix 2 DIT)	2120	6602
DCT2 (PFA $N1 = 7$, $N2 = 9$)	93	302
DCT2 (PFA $N1 = 9$, $N2 = 7$)	103	320
Direct Factorization	149	296

The lowest computational complexity is reached by the algorithm based on the WFTA. Also in this case, an increase in the regularity of the structure determine an increase of the computational complexity as demonstrated by the algorithms derived from the DFT and the DCT2 via PFA. A regular structure is also present in the algorithm obtained from the direct factorization of the DCT5. Moreover, this algorithm, as well as the ones obtained from the PFA, allows also the computation of the 4-point DCT5. Finally, even if Bluestein's algorithm is characterized by a high level of regularity and by the presence of submodules that can be shared for the computation of the DCT5 of other lengths, its complexity is higher than the one required by the direct implementation of the matrix-vector multiplication. Furthermore, except for the one derived from the WFTA whose

cascaded multiplications are only due to the pre- and post-normalization, all the algorithms are characterized by the presence of several cascaded multipliers.

Chapter 7

16-Point DCT5 Implementation

This chapter is devoted to describing the development of an integrated circuit, which can be used to compute the 16-point DCT5. First of all, an algorithm is selected among those presented in the previous chapters. Starting from the selected algorithm, a fixed-point version is obtained and implemented in C language. The C-model is therefore included in the JEM software and the performances are evaluated by performing simulations. The architecture is then developed and tested. Finally, the circuit is synthesized and its timing performances are analyzed as well as the occupied area and the power cosumption.

7.1 Algorithm Selection

Taking into account the computational complexities reported in table 6.3, the algorithm that requires the lowest number of operations is the one obtained by following the procedure presented in section A.1.1. Moreover, apart from the multiplications needed for the pre- and post-normalization, this does not require cascaded multipliers. Hence, this algorithm is selected among the possible ones. A MatLab implementation of the floating-point version of the algorithm is reported in appendix D.

7.2 Fixed-Point Algorithm

The fixed-point version of the algorithm is derived by converting the fractional constants into integer constants. In particular, the generic integer constant $\widehat{\mathrm{C}}_i$ is obtained from the fractional constant C_i by using the following equation:

$$\widehat{\mathrm{C}}_i = \lfloor \mathrm{C}_i \times 2^9 + 0.5 \rfloor$$

A MatLab implementation of the fixed-point version of the algorithm is also reported in appendix D.

7.3 C-Model

The C-Model is obtained by converting the fixed-point version of the algorithm (presented in section D.2) in C language. In particular, the type of each variable must be defined so that no overflow occurs when inputs of type *int* are supplied.

7.4 JEM Simulations

The JEM software provides an implementation of the 16-point DCT5 in the function "fastForwardDCT5_B16"described in the file "TComTrQuant.cpp". This is the direct implementation of the matrix-vector multiplication. The C-Model of the new algorithm can therefore be substituted to the proposed one and the JEM simulation can be performed with both the original model and the C-Model obtained as described in section 7.3. In particular, the encoder behavior can be simulated by using the configuration files "encoder_intra_jvet10.cfg"and "RaceHorsesC.cfg"and considering the test sequence "RaceHorses_832x480_30.yuv". The configuration files can be specified when the executable "TAppEncoderStatic"is launched. Another parameter is also needed. This parameter is QP and must be set equal to one of the following values: 22, 27, 32, 37. The obtained values of PSNR are reported in table 7.1.

Table 7.1: *PSNR and Bit Rate for different values of QP obtained by applying the direct implementation of the matrix-vector multiplication (MVM) and the new algorithm*

QP	*MVM*		*New Algorithm*	
	BitRate	PSNR	BitRate	PSNR
22	1758.3734	43.1979	1758.8384	43.1973
27	1056.7232	39.9631	1056.6789	39.9637
32	611.4884	36.8599	611.4947	36.8592
37	315.3521	33.6566	315.4666	33.6598

The Bjøntegaard delta (table 7.2) can therefore be computed from the values of PSNR shown in table 7.1.

Table 7.2: *Bjøntegaard Delta*

$\Delta_{\text{Bjøntegaard}}$	
DSNR	Rate
-1.6300×10^{-4}	-6.0479×10^{-5}

7.5 Architecture Development

In order to optimize the circuit, the RAGn technique is used to implement each multiplication. By adopting this technique, the circuit requires no multipliers and only 249 adders/subtractors (except for the ones used to implement the sign changes).

Moreover, the number of bits needed at each node is analyzed by performing a simulation. According to the JEM specifications, the inputs are represented by using 16 bits. Hence, in order to obtain the number of bits required at each node, the following inputs are supplied:

- 10000 vectors whose elements belong to the range [−32768, 32767];
- vectors obtained by considering all the possible combinations of elements equal to -32768 or 32767.

Finally, input and output registers are inserted for each input and output signal.

7.6 HDL Description and Simulation

The architecture presented in section 7.5 is described in VHDL and simulated by using the software ModelSim. More in detail, the inputs used in the test-bench are obtained by modifying the JEM software so that the values inputted to the function "fastForwardDCT5_B16"are printed to a file. The file containing the input values is therefore generated by running the executable "TAppEncoderStatic"(as described in section 7.4) for a limited amount of time.

7.7 Logic Synthesis

The circuit is synthesized with Synopsys Design Compiler using the *NanGate FreePDK45 Open Cell Library*. The results concerning the area and the timing performances are reported in tables 7.3 and 7.4.

Table 7.3: *Synthesis Results*

Minimum Clock Period (ns)	*Equivalent number of gates*
4.20	56 745

Table 7.4: *Area Estimation*

Combinational Area	*Buf/Inv Area*	*Noncombinational Area*	*Total Cell Area*
45 282.2	6825.6	3664.9	48 947.2

7.8 Post-synthesis simulation and power estimation

The synthesized circuit is simulated in order to also estimate the switching activity at each node and the obtained results are printed in a *saif file*. This is used to evaluate the power consumption and the obtained results are reported in table 7.5.

Table 7.5: *Power Consumption Estimation @ 59.5* MHz

	mW	%
Cell Internal Power	2.5262	37.0
Net Switching Power	3.0854	45.1
Total Dynamic Power	5.6116	82.1
Cell Leakage Power	1.2255	17.9
Total Power	6.8372	

Appendix A

DCT5 via DFT for longer lengths

This appendix presents how the DCT5 can be computed via DFT for $N = 16$ and $N = 32$.

A.1 WFTA for longer lengths

This section shows how the WFTA can be applied to the DCT5 for $N = 16$ and $N = 32$.

A.1.1 WFTA for DCT5 ($N = 16$)

A DCT5 of length 16 can be mapped into a DFT of length 31 as described in section 2.1. In particular, the DCT5 equation can be written as

$$Y_n = \mathrm{c}_{16} T_n \mathbf{Re}(Y_{\mathrm{F}n}) \qquad \text{for } n = 0,1,...,15 \tag{A.1}$$

where

- $\mathrm{c}_{16} = \frac{2}{\sqrt{31}}$
- $\mathbf{Re}(Y_{\mathrm{F}n}) = \mathbf{Re}\left(\sum_{k=0}^{30} l_k \mathrm{e}^{-\mathrm{j}nk\frac{2\pi}{31}}\right)$

and

$$\begin{aligned}
\boldsymbol{l} &= \begin{bmatrix} l_0 & l_1 & l_2 & l_3 & \dots & l_{14} & l_{15} & l_{16} & l_{17} & l_{18} & \dots & l_{27} & l_{28} & l_{29} & l_{30} \end{bmatrix}^{\mathrm{T}} \\
&= \begin{bmatrix} \hat{x}_0 & 0 & \hat{x}_2 & 0 & \dots & \hat{x}_{14} & 0 & \hat{x}_{15} & 0 & \hat{x}_{13} & \dots & 0 & \hat{x}_3 & 0 & \hat{x}_1 \end{bmatrix}^{\mathrm{T}} \\
&= \begin{bmatrix} \frac{x_0}{\sqrt{2}} & 0 & x_2 & 0 & \dots & x_{14} & 0 & x_{15} & 0 & x_{13} & \dots & 0 & x_3 & 0 & x_1 \end{bmatrix}^{\mathrm{T}}
\end{aligned}$$

Equation A.1 can be further rearranged to give

$$Y_n = T_n \mathbf{Re}(\mathrm{c}_{16} Y_{\mathrm{F}n}) = T_n \mathbf{Re}(\hat{Y}_{\mathrm{F}n})$$

where

$$\hat{\boldsymbol{Y}}_{\mathbf{F}} = \begin{bmatrix} \hat{Y}_{\mathrm{F}_0} \\ \vdots \\ \hat{Y}_{\mathrm{F}_{30}} \end{bmatrix} = \mathrm{c}_{16}\,[S_{31}]\,[C_{31}]\,[T_{31}]\,\boldsymbol{l} = [S_{31}]\left[\hat{C}_{31}\right][T_{31}]\,\boldsymbol{l}$$

More in detail, the matrices $[S_{31}]$, $\left[\hat{C}_{31}\right]$ and $[T_{31}]$ can be found according to the following steps:

1. the DFT is expressed as a function of a circular convolution by using Rader's theorem;
2. the circular convolution is solved by making use of the Winograd short convolution algorithms.

Nevertheless, this procedure is cumbersome and leads to an algorithm that requires a large number of sums. A better method to solve this DFT was proposed by *Selesnick* and *Burrus* ([22], [23]). This can be efficiently used to compute the DCT5. In fact, several simplifications can be made. These are due to the following facts:

1. the input sequence is real;
2. 15 out of 31 inputs are equal to zero;
3. the computation of only the real part of the DFT is needed.

By taking these aspects into account, the method yields an algorithm that only requires 42 multiplications and 165 sums.

A.1.2 WFTA for DCT5 ($N = 32$)

A DCT5 of length 32 can be mapped into a DFT of length 63 as described in section 2.1. In particular, the DCT5 equation can be written as

$$Y_n = \mathrm{c}_{32} T_n \mathbf{Re}(Y_{\mathrm{F}n}) \qquad \text{for } n = 0,1,...,31 \tag{A.2}$$

where

- $\mathrm{c}_{32} = \frac{2}{\sqrt{63}}$
- $\mathbf{Re}(Y_{\mathrm{F}n}) = \mathbf{Re}\left(\sum_{k=0}^{62} l_k \mathrm{e}^{-\mathrm{j}nk\frac{2\pi}{63}}\right)$

and

$$\begin{aligned} \boldsymbol{l} &= \begin{bmatrix} l_0 & l_1 & l_2 & l_3 & l_4 & \dots & l_{29} & l_{30} & l_{31} & l_{32} & l_{33} & l_{34} & l_{35} & \dots & l_{60} & l_{61} & l_{62} \end{bmatrix}^{\mathrm{T}} \\ &= \begin{bmatrix} \hat{x}_0 & 0 & \hat{x}_2 & 0 & \hat{x}_4 & \dots & 0 & \hat{x}_{30} & 0 & \hat{x}_{31} & 0 & \hat{x}_{29} & 0 & \dots & \hat{x}_3 & 0 & \hat{x}_1 \end{bmatrix}^{\mathrm{T}} \\ &= \begin{bmatrix} \frac{x_0}{\sqrt{2}} & 0 & x_2 & 0 & x_4 & \dots & 0 & x_{30} & 0 & x_{31} & 0 & x_{29} & 0 & \dots & x_3 & 0 & x_1 \end{bmatrix}^{\mathrm{T}} \end{aligned}$$

Equation A.2 can be further rearranged to give

$$Y_n = T_n \mathbf{Re}(\mathrm{c}_{32} Y_{\mathrm{F}n}) \; = T_n \mathbf{Re}(\hat{Y}_{\mathrm{F}n})$$

The elements of $\hat{\mathbf{Y}}_{\mathbf{F}}$ are contained in the matrix $[Z]$ which is calculated by applying the equation

$$[Z] = \mathrm{c}_{32}\,[S_7]\,([S_9]\,[C_{9\times 7}] \circ [T_9]\,([T_7]\,[z])^{\mathrm{T}})^{\mathrm{T}} = [S_7]\left([S_9]\left[\hat{C}_{9\times 7}\right] \circ [T_9]\,([T_7]\,[z])^{\mathrm{T}}\right)^{\mathrm{T}}$$

where

$$\left[\hat{C}_{9\times 7}\right] = \mathrm{c}_{32}\,[C_{9\times 7}]$$

This equation yields an algorithm which is composed of five main steps:

- Pre-Normalization
- Pre-Additions
- Multiplications
- Post-Additions
- Post-Normalization

The algorithm can be derived by following the same procedure presented for the case $N = 8$. More in detail, the matrices $[T_7]$, $[S_7]$ and $[C_7]$ are the same adopted for the case $N = 4$. On the other hand,

$$[T_9] = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & -1 & 0 & 0 & 0 & 0 & -1 & 1 \\ 0 & 0 & 1 & 0 & -1 & -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 & 1 & 1 & 0 & 0 & -1 \\ 0 & 1 & -1 & 0 & 1 & -1 & 0 & 1 & -1 \\ 0 & 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & -1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & -1 & 0 & -1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & -1 & 1 & 0 & 0 & -1 \end{bmatrix}$$

$$[S_9] = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 2 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 1 & -1 & 2 & 0 & -1 & 1 & 0 & -1 & 0 & 1 & -1 \\ 1 & 0 & 3 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & -1 & 2 & -1 & 0 & -1 & 0 & 1 & -1 & 0 & -1 \\ 1 & -1 & 2 & -1 & 0 & -1 & 0 & -1 & 1 & 0 & 1 \\ 1 & 0 & 3 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 1 & -1 & 2 & 0 & -1 & 1 & 0 & 1 & 0 & -1 & 1 \\ 1 & -1 & 2 & 1 & 1 & 0 & 0 & -1 & -1 & -1 & 0 \end{bmatrix}$$

and $[C_9]$ is a diagonal matrix whose main diagonal contains the elements of the vector

$$\boldsymbol{C_9} = \begin{bmatrix} 1 \\ 1.5 \\ -0.5 \\ \frac{1}{3}\left[2\cos(\theta) - \cos(2\theta) - \cos(4\theta)\right] \\ \frac{1}{3}\left[\cos(\theta) + \cos(2\theta) - 2\cos(4\theta)\right] \\ \frac{1}{3}\left[\cos(\theta) - 2\cos(2\theta) + \cos(4\theta)\right] \\ \mathrm{j}\sin(3\theta) \\ \mathrm{j}\sin(3\theta) \\ -\mathrm{j}\sin(\theta) \\ -\mathrm{j}\sin(4\theta) \\ -\mathrm{j}\sin(2\theta) \end{bmatrix} \qquad \theta = -\frac{2\pi}{9}$$

It should be highlighted that the matrix $[S_9]$ contains elements that are not equal to 0, 1 or -1. This form of the matrix can be used to minimize the number of multiplications by introducing shift operations.

Pre-Normalization

The multiplication

$$M_{\mathrm{n1}} = \hat{x}_0 = \mathrm{C}_{\mathrm{norm}} x_0 = \frac{1}{\sqrt{2}} x_0$$

is performed.

Pre-Additions

The additions related to this step are obtained by performing the matrix products

$$[A] = [T_9]\left([T_7]\,[z]\right)^{\mathrm{T}}$$

The matrix $[A]$ is composed of 99 elements. Each element $A_{i,j}$ of the matrix $[A]$ needs additions to be performed. More in detail:

$$[A] = \begin{bmatrix} A_{0,0} & A_{0,1} & A_{0,2} & A_{0,3} & A_{0,4} & U & U & U & U \\ A_{1,0} & A_{1,1} & A_{1,2} & A_{1,3} & A_{1,4} & U & U & U & U \\ A_{2,0} & A_{2,1} & A_{2,2} & A_{2,3} & A_{2,4} & U & U & U & U \\ A_{3,0} & A_{3,1} & A_{3,2} & A_{3,3} & A_{3,4} & U & U & U & U \\ A_{4,0} & A_{4,1} & A_{4,2} & A_{4,3} & A_{4,4} & U & U & U & U \\ A_{5,0} & A_{5,1} & A_{5,2} & A_{5,3} & A_{5,4} & U & U & U & U \\ U & U & U & U & U & A_{6,5} & A_{6,6} & A_{6,7} & A_{6,8} \\ U & U & U & U & U & A_{7,5} & A_{7,6} & A_{7,7} & A_{7,8} \\ U & U & U & U & U & A_{8,5} & A_{8,6} & A_{8,7} & A_{8,8} \\ U & U & U & U & U & A_{9,5} & A_{9,6} & A_{9,7} & A_{9,8} \\ U & U & U & U & U & A_{10,5} & A_{10,6} & A_{10,7} & a_{10,8} \end{bmatrix}$$

where the elements labeled as "U"are only needed for the computation of the imaginary part of the DFT.

Since the number of required additions is much larger than the one analyzed for the case $N = 8$, it cannot be reduced by inspection method. In the following, a possible algorithm that can be adopted to minimize the number of pre-additions is therefore presented.

Minimization of the number of pre-additions

Each element of the matrix $[A]$ can be described by the equation:

$$A_{i,j} = \sum_{k=0}^{31} u_{i,j,k}\hat{x}_k$$

where $u_{i,j,k}$ can be equal to 1, 0 or -1. Hence a vector

$$\boldsymbol{u_{i,j}} = \begin{bmatrix} u_{i,j,0} & u_{i,j,1} & u_{i,j,2} & \dots & u_{i,j,30} & u_{i,j,31} \end{bmatrix}$$

can be found for each element of the matrix $[A]$.

The vectors $\boldsymbol{u_{i,j}}$ can be gathered into a matrix $[w]$. Thus, this matrix can be defined as

$$[w] = \begin{bmatrix} \boldsymbol{u_{0,0}} \\ \boldsymbol{u_{0,1}} \\ \vdots \\ \boldsymbol{u_{0,4}} \\ \boldsymbol{u_{1,0}} \\ \vdots \\ \boldsymbol{u_{10,8}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{w_0} \\ \boldsymbol{w_1} \\ \vdots \\ \boldsymbol{w_4} \\ \boldsymbol{w_5} \\ \vdots \\ \boldsymbol{w_{49}} \end{bmatrix} = \begin{bmatrix} w_{0,0} & w_{0,1} & \dots & w_{0,31} \\ w_{1,0} & w_{1,1} & \dots & w_{1,31} \\ \vdots & \vdots & & \vdots \\ w_{4,0} & w_{4,1} & \dots & w_{4,31} \\ w_{5,0} & w_{5,1} & \dots & w_{5,31} \\ \vdots & \vdots & & \vdots \\ w_{49,0} & w_{49,1} & \dots & w_{49,31} \end{bmatrix}$$

The elements related to the computation of the imaginary part of the DFT are not included in this matrix.

Each couple ($w_{i,j}$,$w_{i,k}$) is therefore analyzed by varying i. Hence, the values assumed by the couples ($w_{0,0}$,$w_{0,1}$), ($w_{1,0}$,$w_{1,1}$), …, ($w_{49,0}$,$w_{49,1}$) are firstly analyzed. Secondly, the values assumed by the couples ($w_{0,0}$,$w_{0,2}$), ($w_{1,0}$,$w_{1,2}$), …, ($w_{49,0}$,$w_{49,2}$) are considered and the process is iterated for all the possible values of j and k. In the end, the couple ($\hat{j}$, $\hat{k}$) is selected, that is the one that most frequently satisfies one of the following two conditions:

1. $(w_{i,j},w_{i,k}) = (1, \quad 1) \quad \text{OR} \quad (w_{i,j},w_{i,k}) = (-1, -1)$
2. $(w_{i,j},w_{i,k}) = (1, -1) \quad \text{OR} \quad (w_{i,j},w_{i,k}) = (-1, \quad 1)$

In case more than one couple satisfies one of these conditions with the maximum number of occurrences, a random number can be generated to select one of those couples.

If the selected couple satisfies condition 1):

- The sum

  $$s_1 = \hat{x}_{\hat{j}} + \hat{x}_{\hat{k}}$$

  is performed.

- A new matrix $[\hat{w}]$ is defined starting from the matrix $[w]$. More in detail, the matrix $[\hat{w}]$ is equal to the matrix $[w]$ except for the fact that the couples $(w_{i,\hat{j}},w_{i,\hat{k}})$, which are equal to (1,1) or (-1,-1) in the matrix $[w]$ are set equal to (0,0) in the matrix $[\hat{w}]$. Moreover, a column is added to represent the sum s_1. The i-th element of this column is such that:

$$\hat{w}_{i,32} = \begin{cases} 0 & \text{if } (w_{i,\hat{j}},w_{i,\hat{k}}) \neq (\ 1,\ \ 1) \quad \text{AND} \quad (w_{i,\hat{j}},w_{i,\hat{k}}) \neq (-1,-1) \\ 1 & \text{if } (w_{i,\hat{j}},w_{i,\hat{k}}) = (\ 1,\ \ 1) \\ -1 & \text{if } (w_{i,\hat{j}},w_{i,\hat{k}}) = (-1,-1) \end{cases}$$

On the other hand, if the selected couple satisfies condition 2):

- The sum

$$s_1 = \hat{x}_{\hat{j}} - \hat{x}_{\hat{k}}$$

is performed.

- A new matrix $[\hat{w}]$ is defined starting from the matrix $[w]$. More in detail, the matrix $[\hat{w}]$ is equal to the matrix $[w]$ except for the fact that the couples $(w_{i,\hat{j}},w_{i,\hat{k}})$, which are equal to (1,-1) or (-1,1) in the matrix $[w]$, are set equal to (0,0) in the matrix $[\hat{w}]$. Moreover, a column is added to represent the sum s_1. The i-th element of this column is such that:

$$\hat{w}_{i,32} = \begin{cases} 0 & \text{if } (w_{i,\hat{j}},w_{i,\hat{k}}) \neq (\ 1,-1) \quad \text{AND} \quad (w_{i,\hat{j}},w_{i,\hat{k}}) \neq (-1,\ \ 1) \\ 1 & \text{if } (w_{i,\hat{j}},w_{i,\hat{k}}) = (\ 1,-1) \\ -1 & \text{if } (w_{i,\hat{j}},w_{i,\hat{k}}) = (-1,\ \ 1) \end{cases}$$

The process described above can be reiterated by considering the matrix $[\hat{w}]$. This obviously implies that the sum s_1 has to be considered among the possible elements. Hence, if $\hat{k} = 32$, the sum to be performed is

$$s_2 = \hat{x}_{\hat{j}} + s_1$$

in case condition 1) is satisfied and

$$s_2 = \hat{x}_{\hat{j}} - s_1$$

in case condition 2) holds.

In particular, this procedure can be repeated until each line of the matrix contains only one element different from zero. When this happens, all the additions, needed to compute all the significant elements of the matrix $[A]$, have been performed.

Moreover, the whole process can be reiterated so that different random numbers can be generated to select one of the couples that, at each step, satisfy one of the two conditions with the maximum number of occurrences. The obtained number of pre-additions can therefore be compared to the one achieved at the end of the previous iterations. Hence, the set of equations, derived from the iteration that guarantees the lowest number of pre-additions, can be selected.

Multiplications

The multiplications derive from the element-by-element matrix product

$$[M] = \left[\hat{C}_{9\times 7}\right] \circ [A]$$

The derivation of the needed multiplications is not reported here since it is analogous to the one described for the case $N = 8$.

Post-Additions

The additions related to this step are obtained by performing the matrix products

$$[Z] = [S_7]\left([S_9]\,[M]\right)^{\mathrm{T}}$$

The matrix $[Z]$ is composed of 63 elements, which need additions to be performed. More in detail:

$$[Z] = \mathbf{Re}\left(\begin{bmatrix} \hat{Y}_{\mathrm{F0}} & \hat{Y}_{\mathrm{F28}} & U & \hat{Y}_{\mathrm{F21}} & U & \hat{Y}_{\mathrm{F14}} & U & \hat{Y}_{\mathrm{F7}} & U \\ U & \hat{Y}_{\mathrm{F1}} & \hat{Y}_{\mathrm{F29}} & U & \hat{Y}_{\mathrm{F22}} & U & \hat{Y}_{\mathrm{F15}} & U & \hat{Y}_{\mathrm{F8}} \\ \hat{Y}_{\mathrm{F9}} & U & \hat{Y}_{\mathrm{F2}} & \hat{Y}_{\mathrm{F30}} & U & \hat{Y}_{\mathrm{F23}} & U & \hat{Y}_{\mathrm{F16}} & U \\ U & \hat{Y}_{\mathrm{F10}} & U & \hat{Y}_{\mathrm{F3}} & \hat{Y}_{\mathrm{F31}} & U & \hat{Y}_{\mathrm{F24}} & U & \hat{Y}_{\mathrm{F17}} \\ \hat{Y}_{\mathrm{F18}} & U & \hat{Y}_{\mathrm{F11}} & U & \hat{Y}_{\mathrm{F4}} & U & U & \hat{Y}_{\mathrm{F25}} & U \\ U & \hat{Y}_{\mathrm{F19}} & U & \hat{Y}_{\mathrm{F12}} & U & \hat{Y}_{\mathrm{F5}} & U & U & \hat{Y}_{\mathrm{F26}} \\ \hat{Y}_{\mathrm{F27}} & U & \hat{Y}_{\mathrm{F20}} & U & \hat{Y}_{\mathrm{F13}} & U & \hat{Y}_{\mathrm{F6}} & U & U \end{bmatrix}\right)$$

where the elements labeled as "U"are not needed for the computation of the DCT5. Nevertheless, since the inputs of the DFT are real, these elements are equal to those which are not labeled as "U". Moreover, also in this case, the number of required additions is much larger than the one analyzed for the case $N = 8$ and consequently it cannot be reduced by inspection method. In the following, a possible algorithm, which can be adopted to minimize the number of post-additions, is therefore presented.

Minimization of the number of post-additions

Since

$$[M] = \begin{bmatrix} M_0 & M_1 & M_2 & M_3 & M_4 & U & U & U & U \\ M_5 & M_6 & M_7 & M_8 & M_9 & U & U & U & U \\ M_{10} & M_{11} & M_{12} & M_{13} & M_{14} & U & U & U & U \\ M_{15} & M_{16} & M_{17} & M_{18} & M_{19} & U & U & U & U \\ M_{20} & M_{21} & M_{22} & M_{23} & M_{24} & U & U & U & U \\ M_{25} & M_{26} & M_{27} & M_{28} & M_{29} & U & U & U & U \\ U & U & U & U & U & M_{30} & M_{31} & M_{32} & M_{33} \\ U & U & U & U & U & M_{34} & M_{35} & M_{36} & M_{37} \\ U & U & U & U & U & M_{38} & M_{39} & M_{40} & M_{41} \\ U & U & U & U & U & M_{42} & M_{43} & M_{44} & M_{45} \\ U & U & U & U & U & M_{46} & M_{47} & M_{48} & M_{49} \end{bmatrix}$$

each element of the matrix $[Z]$ can be expressed as

$$Z_{i,j} = \sum_{k=0}^{49} u_{i,j,k} M_k$$

where $u_{i,j,k}$ can be equal to -3, -2, -1, 0, 1, 2 or 3. Hence a vector

$$\boldsymbol{u_{i,j}} = \begin{bmatrix} u_{i,j,0} & u_{i,j,1} & u_{i,j,2} & \dots & u_{i,j,48} & u_{i,j,49} \end{bmatrix}$$

can be found for each element of the matrix $[Z]$.

Starting from the vector $\boldsymbol{u_{i,j}}$, a new vector

$$\boldsymbol{n_{i,j}} = \begin{bmatrix} n_{i,j,0} & n_{i,j,1} & \dots & n_{i,j,98} & n_{i,j,99} \end{bmatrix}$$

can be defined. The vector $\boldsymbol{n_{i,j}}$ is such that

$$(n_{i,j,2k},n_{i,j,2k+1}) = \begin{cases} (-1\ ,-1) & \text{if } u_{i,j,k} = -3 \\ (-1\ ,\ 0\ ) & \text{if } u_{i,j,k} = -2 \\ (\ 0\ \ ,-1) & \text{if } u_{i,j,k} = -1 \\ (\ 0\ \ ,\ 0\ ) & \text{if } u_{i,j,k} = 0 \\ (\ 0\ \ ,\ 1\ ) & \text{if } u_{i,j,k} = 1 \\ (\ 1\ \ ,\ 0\ ) & \text{if } u_{i,j,k} = 2 \\ (\ 1\ \ ,\ 1\ ) & \text{if } u_{i,j,k} = 3 \end{cases}$$

A matrix

$$[w] = \begin{bmatrix} \boldsymbol{n_{0,0}} \\ \boldsymbol{n_{1,1}} \\ \vdots \\ \boldsymbol{n_{2,3}} \\ \boldsymbol{n_{3,4}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{w_0} \\ \boldsymbol{w_1} \\ \vdots \\ \boldsymbol{w_{30}} \\ \boldsymbol{w_{31}} \end{bmatrix} = \begin{bmatrix} w_{0,0} & w_{0,1} & \dots & w_{0,99} \\ w_{1,0} & w_{1,1} & \dots & w_{1,99} \\ \vdots & \vdots & & \vdots \\ w_{30,0} & w_{30,1} & \dots & w_{30,99} \\ w_{31,0} & w_{31,1} & \dots & w_{31,99} \end{bmatrix}$$

is then defined.

Each couple $(w_{i,j},w_{i,k})$ is therefore analyzed by varying i. Hence the values assumed by the couples $(w_{0,0},w_{0,1})$, $(w_{1,0},w_{1,1})$, ..., $(w_{31,0},w_{31,1})$ are firstly analyzed. Secondly, the values assumed by the couples $(w_{0,0},w_{0,2})$, $(w_{1,0},w_{1,2})$, ..., $(w_{31,0},w_{31,2})$ are considered and the process is iterated for all the possible values of j and k. In the end, the couple $(\hat{j},\ \hat{k})$ is selected, that is the one that most frequently satisfies one of the following two conditions:

1. $(w_{i,j},w_{i,k}) = (1,\ \ 1)\ \ \text{OR}\ \ (w_{i,j},w_{i,k}) = (-1,-1)$
2. $(w_{i,j},w_{i,k}) = (1,-1)\ \ \text{OR}\ \ (w_{i,j},w_{i,k}) = (-1,\ \ 1)$

In case more than one couple satisfies one of these conditions with the maximum number of occurrences, a random number can be generated to select one of those couples.

If the selected couple satisfies condition 1):

- The sum s_1 is performed according to the following rule:

$$s_1 = \begin{cases} 2M_{\frac{\hat{j}}{2}} + 2M_{\frac{\hat{k}}{2}} & \text{if } (\hat{j} \bmod 2) = 0 \quad \text{AND} \quad (\hat{k} \bmod 2) = 0 \\ 2M_{\frac{\hat{j}}{2}} + M_{\lfloor \frac{\hat{k}}{2} \rfloor} & \text{if } (\hat{j} \bmod 2) = 0 \quad \text{AND} \quad (\hat{k} \bmod 2) = 1 \\ M_{\lfloor \frac{\hat{j}}{2} \rfloor} + 2M_{\frac{\hat{k}}{2}} & \text{if } (\hat{j} \bmod 2) = 1 \quad \text{AND} \quad (\hat{k} \bmod 2) = 0 \\ M_{\lfloor \frac{\hat{j}}{2} \rfloor} + M_{\lfloor \frac{\hat{k}}{2} \rfloor} & \text{if } (\hat{j} \bmod 2) = 1 \quad \text{AND} \quad (\hat{k} \bmod 2) = 1 \end{cases}$$

- A new matrix $[\hat{w}]$ is defined starting from the matrix $[w]$. More in detail, the matrix $[\hat{w}]$ is equal to the matrix $[w]$ except for the fact that the couples $(w_{i,\hat{j}},w_{i,\hat{k}})$, which are equal to (1,1) or (-1,-1) in the matrix $[w]$ are set equal to (0,0) in the matrix $[\hat{w}]$. Moreover, a column is added to represent the sum s_1. The i-th element of this column is such that:

$$\hat{w}_{i,100} = \begin{cases} 0 & \text{if } (w_{i,\hat{j}},w_{i,\hat{k}}) \neq (\ \ 1, \ \ 1) \quad \text{AND} \quad (w_{i,\hat{j}},w_{i,\hat{k}}) \neq (-1,-1) \\ 1 & \text{if } (w_{i,\hat{j}},w_{i,\hat{k}}) = (\ \ 1, \ \ 1) \\ -1 & \text{if } (w_{i,\hat{j}},w_{i,\hat{k}}) = (-1,-1) \end{cases}$$

On the other hand, if the selected couple satisfies condition 2):

- The sum s_1 is performed according to the following rule:

$$s_1 = \begin{cases} 2M_{\frac{\hat{j}}{2}} - 2M_{\frac{\hat{k}}{2}} & \text{if } (\hat{j} \bmod 2) = 0 \quad \text{AND} \quad (\hat{k} \bmod 2) = 0 \\ 2M_{\frac{\hat{j}}{2}} - M_{\lfloor \frac{\hat{k}}{2} \rfloor} & \text{if } (\hat{j} \bmod 2) = 0 \quad \text{AND} \quad (\hat{k} \bmod 2) = 1 \\ M_{\lfloor \frac{\hat{j}}{2} \rfloor} - 2M_{\frac{\hat{k}}{2}} & \text{if } (\hat{j} \bmod 2) = 1 \quad \text{AND} \quad (\hat{k} \bmod 2) = 0 \\ M_{\lfloor \frac{\hat{j}}{2} \rfloor} - M_{\lfloor \frac{\hat{k}}{2} \rfloor} & \text{if } (\hat{j} \bmod 2) = 1 \quad \text{AND} \quad (\hat{k} \bmod 2) = 1 \end{cases}$$

- A new matrix $[\hat{w}]$ is defined starting from the matrix $[w]$. More in detail, the matrix $[\hat{w}]$ is equal to the matrix $[w]$ except for the fact that the couples $(w_{i,\hat{j}},w_{i,\hat{k}})$, which are equal to (1,-1) or (-1,1) in the matrix $[w]$, are set equal to (0,0) in the matrix $[\hat{w}]$. Moreover, a column is added to represent the sum s_1. The i-th element of this column is such that:

$$\hat{w}_{i,100} = \begin{cases} 0 & \text{if } (w_{i,\hat{j}},w_{i,\hat{k}}) \neq (\ \ 1, -1) \quad \text{AND} \quad (w_{i,\hat{j}},w_{i,\hat{k}}) \neq (-1, \ \ 1) \\ 1 & \text{if } (w_{i,\hat{j}},w_{i,\hat{k}}) = (\ \ 1, -1) \\ -1 & \text{if } (w_{i,\hat{j}},w_{i,\hat{k}}) = (-1, \ \ 1) \end{cases}$$

The process described above can be reiterated by considering the matrix $[\hat{w}]$. This obviously implies that the sum s_1 has to be considered among the possible elements. In particular, this procedure can be repeated until each line of the matrix contains only one

element different from zero. When this happens, all the additions needed to compute all the significant elements of the matrix $[Z]$, have been performed.

Moreover, the whole process can be reiterated so that different random numbers can be generated to select one of the couples which, at each step, satisfy one of the two conditions with the maximum number of occurrences. The obtained number of post-additions can therefore be compared to the one achieved at the end of the previous iterations. Hence, the set of equations derived from the iteration, which guarantees the lowest number of post-additions, can be selected.

Post-Normalization

The multiplication

$$M_{\mathrm{n2}} = \mathrm{C}_{\mathrm{norm}} M_0 = \frac{1}{\sqrt{2}} M_0$$

is performed.

Final Algorithm

The procedure described above yields an algorithm, which requires 52 multiplications, 304 sums and 5 shifts.

A.2 PFA for longer lengths

This section presents how the DCT5 can be computed via PFA for $N = 32$.

A.2.1 PFA for DCT5 ($N = 32$)

A DCT5 of length 32 can be mapped into a DFT of length 63 as described in section 2.1. In particular, the DCT5 equation can be written as

$$Y_n = \mathrm{c}_{32} T_n \mathbf{Re}(Y_{\mathrm{F}n}) \qquad \text{for } n = 0,1,...,31$$

where

- $\mathrm{c}_{32} = \frac{2}{\sqrt{63}}$
- $\mathbf{Re}(Y_{\mathrm{F}n}) = \mathbf{Re}\left(\sum_{k=0}^{62} l_k \mathrm{e}^{-\mathrm{j}nk\frac{2\pi}{63}} \right)$

and

$$\begin{aligned}
\boldsymbol{l} &= \begin{bmatrix} l_0 & l_1 & l_2 & l_3 & \dots & l_{29} & l_{30} & l_{31} & l_{32} & l_{33} & l_{34} & l_{35} & \dots & l_{60} & l_{61} & l_{62} \end{bmatrix}^{\mathrm{T}} \\
&= \begin{bmatrix} \hat{x}_0 & 0 & \hat{x}_2 & 0 & \dots & 0 & \hat{x}_{30} & 0 & \hat{x}_{31} & 0 & \hat{x}_{29} & 0 & \dots & \hat{x}_3 & 0 & \hat{x}_1 \end{bmatrix}^{\mathrm{T}} \\
&= \begin{bmatrix} x_0 T_k|_{k=0} & 0 & x_2 & 0 & \dots & 0 & x_{30} & 0 & x_{31} & 0 & x_{29} & 0 & \dots & x_3 & 0 & x_1 \end{bmatrix}^{\mathrm{T}} \\
&= \begin{bmatrix} \frac{x_0}{\sqrt{2}} & 0 & x_2 & 0 & \dots & 0 & x_{30} & 0 & x_{31} & 0 & x_{29} & 0 & \dots & x_3 & 0 & x_1 \end{bmatrix}^{\mathrm{T}}
\end{aligned}$$

For the sake of simplicity, we can neglect c_{32} and consider:

$$T_n = 1 \qquad T_k = 1 \qquad \forall n,k$$

As a consequence, the vector $\boldsymbol{l}$ can be redefined as

$$\begin{aligned}\boldsymbol{l} &= \begin{bmatrix} l_0 & l_1 & l_2 & l_3 & \dots & l_{28} & l_{29} & l_{30} & l_{31} & l_{32} & l_{33} & l_{34} & l_{35} & \dots & l_{60} & l_{61} & l_{62} \end{bmatrix}^{\mathrm{T}} \\ &= \begin{bmatrix} x_0 & 0 & x_2 & 0 & \dots & x_{28} & 0 & x_{30} & 0 & x_{31} & 0 & x_{29} & 0 & \dots & x_3 & 0 & x_1 \end{bmatrix}^{\mathrm{T}}\end{aligned}$$

The input permutation matrix is therefore applied to this vector in order to obtain the vector $\hat{\boldsymbol{l}}$, which is used as input for the first stage of the PFA. Hence, the scheme shown in figure A.1(a) is derived. The real part of the elements $Y_{\mathrm{F}n}$ represented in this figure, are the outputs generated by the DCT5 (considered in non-normalized form). Alternatively, the scheme presented in figure A.1(b) can be considered.

The simplifications made for the case $N = 8$ can be adopted also in this case. Moreover, it should be noticed that 7-point DFTs are required in the algorithms presented in figures A.1(a) and A.1(b). Since a 4-point DCT5 can be mapped into a 7-point DFT, the modules used to implement the 7-point DFTs can also be employed for the computation of the 4-point DCT5. Finally, it should be highlighted that the 9-point DFTs can be implemented by following the Winograd algorithm as well as the Radix-3 algorithm.

A.3 Rader's algorithm for longer lengths

This section presents how Rader's algorithm can be adopted to compute the DCT5 for $N = 16$.

A.3.1 Rader's algorithm for DCT5 ($N = 16$)

A DCT5 of length 16 can be mapped into a DFT of length 31 as described in section 2.1. In particular, the DCT5 equation can be written as

$$Y_n = c_{16} T_n \mathbf{Re}(Y_{\mathrm{F}n}) \qquad \text{for } n = 0,1,\dots,15$$

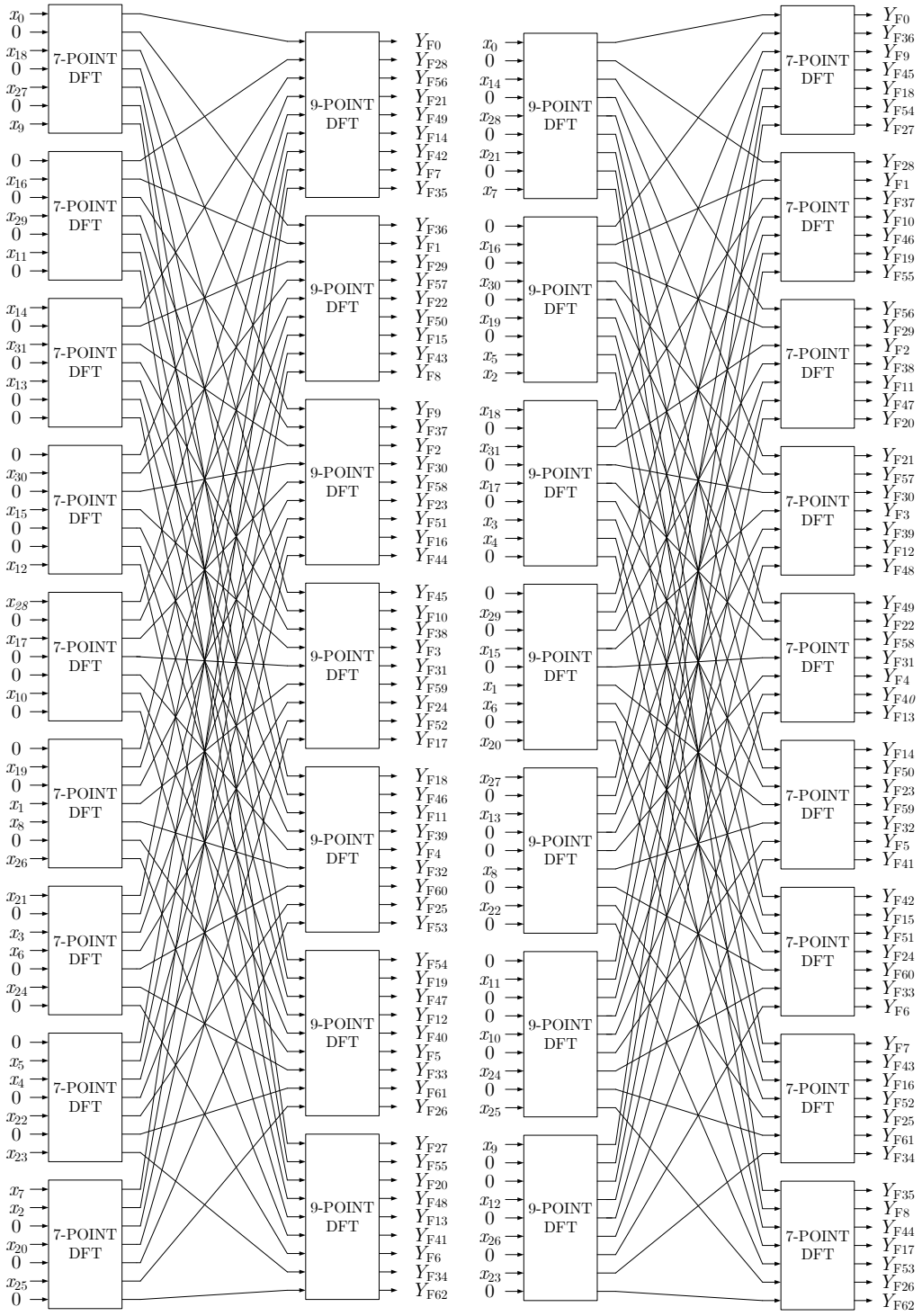
where

- $c_{16} = \frac{2}{\sqrt{31}}$
- $\mathbf{Re}(Y_{\mathrm{F}n}) = \mathbf{Re}\left(\sum_{k=0}^{30} l_k \mathrm{e}^{-\mathrm{j}nk\frac{2\pi}{31}} \right)$

and

$$\begin{aligned}\boldsymbol{l} &= \begin{bmatrix} l_0 & l_1 & l_2 & l_3 & \dots & l_{14} & l_{15} & l_{16} & l_{17} & l_{18} & \dots & l_{28} & l_{29} & l_{30} \end{bmatrix}^{\mathrm{T}} \\ &= \begin{bmatrix} \hat{x}_0 & 0 & \hat{x}_2 & 0 & \dots & \hat{x}_{14} & 0 & \hat{x}_{15} & 0 & \hat{x}_{13} & \dots & \hat{x}_3 & 0 & \hat{x}_1 \end{bmatrix}^{\mathrm{T}} \\ &= \begin{bmatrix} \frac{x_0}{\sqrt{2}} & 0 & x_2 & 0 & \dots & x_{14} & 0 & x_{15} & 0 & x_{13} & \dots & x_3 & 0 & x_1 \end{bmatrix}^{\mathrm{T}}\end{aligned}$$

The computation of the DCT5 is therefore translated into the computation of a DFT of length $N_{\mathrm{DFT}} = 31$. Since N_{DFT} is a prime number, Rader's algorithm can be adopted to compute the DFT. Hence, the steps described at page 47 can be followed to compute the DCT5. In the following, each of these steps will be analyzed.



(a) $N1 = 9$, $N2 = 7$ (b) $N1 = 7$, $N2 = 9$

Figure A.1: *PFA for DCT5 with* $N = 32$

Computation of Y_{F0}

The computation of Y_{F0} is performed according to the following formula:

$$Y_{\mathrm{F0}} = \sum_{n=0}^{N_{\mathrm{DFT}}-1} l_n = \frac{x_0}{\sqrt{2}} + \sum_{n=1}^{15} x_n$$

Hence, Y_0 can be obtained from Y_{F0} as follows

$$Y_0 = \mathrm{c}_{16} T_0 \mathbf{Re}(Y_{\mathrm{F0}}) = \frac{2}{\sqrt{31}} \frac{1}{\sqrt{2}} Y_{\mathrm{F0}}$$

Definition of a primitive root g of N_{DFT}

A primitive root of 31 is

$$g = 3$$

Definition of the vector $\hat{\boldsymbol{l}}$

The vector $\hat{\boldsymbol{l}}$ is defined according to the permutation:

$$\hat{l}_n = l_{(3^{30-n} \bmod 31)} \qquad \text{for } n = 0,1,\ldots,29$$

Definition of the vector $\boldsymbol{T}$

The vector $\boldsymbol{T}$ is defined according to the following formula:

$$T_n = W^{(3^n \bmod 31)} \qquad \text{for } n = 0,1,\ldots,29$$

Computation of the circular convolution $\boldsymbol{c}$ of $\hat{\boldsymbol{l}}$ with $\boldsymbol{T}$

The circular convolution $\boldsymbol{c}$ can be computed according to the Circular Convolution Theorem. Hence, we have:

$$\boldsymbol{c} = \mathrm{IDFT}(\mathrm{DFT}(\hat{\boldsymbol{l}}) \circ \mathrm{DFT}(\boldsymbol{T}))$$

Since $\boldsymbol{T}$ contains only constants, $\boldsymbol{T}_{\mathbf{F}} = \mathrm{DFT}(\boldsymbol{T})$ can be precomputed. On the contrary, the DFT of the vector $\hat{\boldsymbol{l}}$ can be computed according to the WFTA or the PFA. In fact, $\hat{\boldsymbol{l}}$ has length $M = 30 = 2 \times 3 \times 5$. Therefore, the Winograd short-N DFT modules of length 2, 3 and 5 can be used to compute the DFT. Moreover, the IDFT can be obtained by using a forward DFT as depicted in figure 2.19.

Computation of the vector $\hat{\boldsymbol{c}}$

The vector $\hat{\boldsymbol{c}}$ is computed according to the following formula:

$$\hat{c}_n = (\mathbf{Re}(c_n) + l_0)\, \mathrm{c}_{16}$$

Definition of the vector $\boldsymbol{Y}$

The elements $[Y_1, \ldots, Y_{15}]$ of the vector $\boldsymbol{Y}$ are stored in the vector $\hat{\boldsymbol{c}}$ according to the following permutation:

$$Y_{(3^n \bmod 31)} = \hat{c}_n$$

Final Algorithm

The procedure described above leads to the architecture depicted in figure A.3. This architecture is characterized by the presence of fifteen 2-point DFTs and two 15-point DFTs. Each of the 15-point DFTs is also used to obtain an 8-point DCT5.

The 2-point DFTs can be implemented by using butterfly units with unitary twiddle factor. On the other hand, the 15-point DFTs can be implemented by adopting the PFA or the WFTA. In order to minimize the complexity, we suppose that the WFTA is chosen and implemented according to the scheme reported in figure A.2 [24].

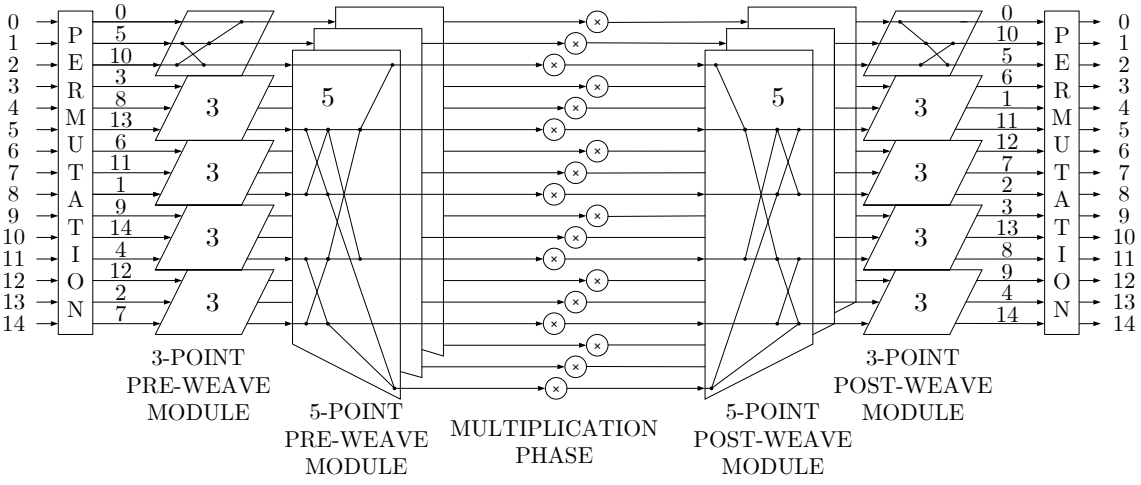


Figure A.2: *WFTA for 15-point DFT*

The computational complexity can therefore be analyzed by making the assumptions reported above. In particular, each 15-point DFT module requires 81 complex additions and 18 complex multiplications. Each complex addition is performed by means of two real additions. Moreover, since the multiplicative constants are purely real or purely imaginary, each complex multiplication needs 2 real multiplications. Hence, each 15-point DFT requires 162 real additions and 36 real multiplications. On the other hand, each 2-point DFT requires 4 real additions. Furthermore, 50 multipliers and 16 adders are needed to process the outputs. Finally, 30 complex multiplications must be performed by the constants stored in the vector $\boldsymbol{T_F}$. Each of these multiplications needs at least 3 real multiplications and 5 real sums or 4 real multiplications and 2 real sums. The number of multipliers (N_{mult}) and adders (N_{add}) can therefore be computed as follows:

$$N_{\text{mult}} = 36 \times 2 + 50 + 30 \times 3 = 212$$

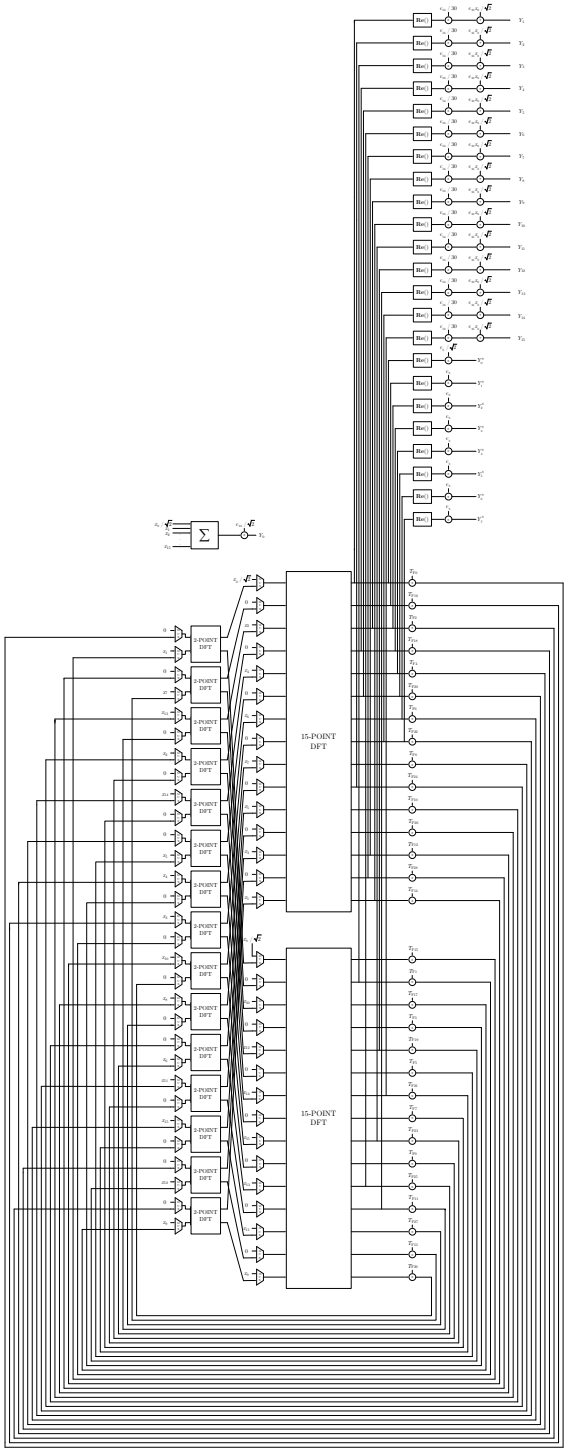


Figure A.3: *Architecture for DCT5 ($N = 16$) based on Rader and Prime Factor algorithms. Two 8-point DCT5 are also obtained by exploiting the 15-point DFTs. The outputs of only one of these are reported in the figure.*

$$N_{\text{add}} = 162 \times 2 + 4 \times 15 + 16 + 30 \times 5 = 550$$

Unfortunately, even if the number of multipliers is lower than the one needed by the direct implementation of the matrix-vector multiplication, the number of required adders makes this architecture unpractical for most of applications.

Appendix B

DCT5 via DCT2 for longer lengths

This appendix presents how the DCT5 can be computed via DCT2 for $N = 16$ and $N = 32$.

B.1 DCT5 via DCT2 ($N = 16$)

An algorithm for the 16-point DCT5 can be derived from algorithms for the 31-point DCT2. More in detail, the computation of a 16-point DCT5 can be translated into the computation of a 16-point DCT6 as described in section 3.1 and the 16-point DCT6 can be derived from the 31-point DCT2 as illustrated in section 3.2. The following paragraphs will illustrate the steps involved in the derivation of the algorithm.

B.1.1 Reordering of the input vector

According to what is described in section 3.1, the input vector $\boldsymbol{x}$ is reordered so that the vector

$$\boldsymbol{x}_{\mathbf{r}} = [J_{16}]\,\boldsymbol{x} = \begin{bmatrix} x_{15} \\ x_{14} \\ \vdots \\ x_1 \\ x_0 \end{bmatrix}$$

is obtained.

B.1.2 Definition of the vector $\boldsymbol{x}_{\mathbf{R}}$

According to what is described in section 3.2, the vector $\boldsymbol{x}_{\mathbf{R}}$ is obtained as

$$\boldsymbol{x}_{\mathbf{R}} = \begin{bmatrix} [I_{15}] & & [J_{15}] \\ & 1 & \\ -[J_{15}] & & [I_{15}] \end{bmatrix}^{-1} \begin{bmatrix} x_{15} \\ x_{14} \\ \vdots \\ x_1 \\ x_0 \\ -x_1 \\ \vdots \\ -x_{14} \\ -x_{15} \end{bmatrix} = \begin{bmatrix} x_{15} \\ 0 \\ x_{13} \\ 0 \\ \vdots \\ x_1 \\ x_0 \\ 0 \\ x_2 \\ 0 \\ \vdots \\ x_{14} \\ 0 \end{bmatrix}$$

B.1.3 Computation of the 31-point DCT2

The vector $\boldsymbol{Y}^{\mathbf{II}}$ is obtained by computing the DCT2 of the vector $\boldsymbol{x}_{\mathbf{R}}$. A possible algorithm for the 31-point DCT2 can be produced by using the software Spiral [25]. Several simplifications can be made to this algorithm. In fact:

1. 15 out of 31 inputs are equal to zero;

2. only the even-indexed outputs of the DCT2 are of interest.

Hence, the computation of the DCT2 leads to the vector:

$$\boldsymbol{Y}^{\mathbf{II}} = \begin{bmatrix} Y_0^{\mathrm{II}} \\ U \\ Y_2^{\mathrm{II}} \\ U \\ \vdots \\ U \\ Y_{28}^{\mathrm{II}} \\ U \\ Y_{30}^{\mathrm{II}} \end{bmatrix}$$

where the elements labeled as U are not of interest for the computation of the 16-point DCT5. More in detail, the output vector produced by the computation of the DCT6 is

$$\boldsymbol{Y}^{\mathbf{VI}} = \begin{bmatrix} Y_0^{\mathrm{VI}} \\ Y_1^{\mathrm{VI}} \\ \vdots \\ Y_{14}^{\mathrm{VI}} \\ Y_{15}^{\mathrm{VI}} \end{bmatrix} = \begin{bmatrix} Y_0^{\mathrm{II}} \\ Y_2^{\mathrm{II}} \\ \vdots \\ Y_{28}^{\mathrm{II}} \\ Y_{30}^{\mathrm{II}} \end{bmatrix}$$

B.1.4 Definition of the output vector $\boldsymbol{Y}$

According to what is described in section 3.1, the output vector $\boldsymbol{Y}$ can be obtained from the vector $\boldsymbol{Y}^{\mathbf{VI}}$ as follows:

$$\boldsymbol{Y} = [D_{16}]\, \boldsymbol{Y}^{\mathbf{VI}}$$

B.1.5 Final Algorithm

The procedure reported above leads to an algorithm that requires 72 multiplications and 144 sums.

B.2 DCT5 via DCT2 ($N = 32$)

An algorithm for the 32-point DCT5 can be derived from algorithms for the 63-point DCT2. More in detail, the computation of a 32-point DCT5 can be translated into the computation of a 32-point DCT6 as described in section 3.1 and the 32-point DCT6 can be derived from the 63-point DCT2 as illustrated in section 3.2. The following paragraphs will illustrate the steps involved in the derivation of the algorithm.

B.2.1 Reordering of the input vector

According to what is described in section 3.1, the input vector $\boldsymbol{x}$ is reordered so that the vector

$$\boldsymbol{x}_{\mathbf{r}} = [J_{32}]\, \boldsymbol{x} = \begin{bmatrix} x_{31} \\ x_{30} \\ \vdots \\ x_1 \\ x_0 \end{bmatrix}$$

is obtained.

B.2.2 Definition of the vector $\boldsymbol{x}_{\mathbf{R}}$

According to what is described in section 3.2, the vector $\boldsymbol{x}_{\mathbf{R}}$ is obtained as

$$\boldsymbol{x}_{\mathbf{R}} = \begin{bmatrix} [I_{31}] & & [J_{31}] \\ & 1 & \\ -[J_{31}] & & [I_{31}] \end{bmatrix}^{-1} \begin{bmatrix} x_{31} \\ x_{30} \\ \vdots \\ x_1 \\ x_0 \\ -x_1 \\ \vdots \\ -x_{30} \\ -x_{31} \end{bmatrix} = \begin{bmatrix} x_{31} \\ 0 \\ x_{29} \\ 0 \\ \vdots \\ x_1 \\ x_0 \\ 0 \\ x_2 \\ 0 \\ \vdots \\ x_{30} \\ 0 \end{bmatrix}$$

B.2.3 Computation of the 31-point DCT2

The vector $\boldsymbol{Y}^{\mathbf{II}}$ is obtained by computing the DCT2 of the vector $\boldsymbol{x}_{\mathbf{R}}$. A possible algorithm for the 63-point DCT2 can be derived by adopting the PFA as illustrated for the case $N = 8$ in section 3.4.3. Several simplifications can be made to this algorithm. In fact:

1. 31 out of 63 inputs are equal to zero;

2. only the even-indexed outputs of the DCT2 are of interest.

Hence, the computation of the DCT2 leads to the vector:

$$\boldsymbol{Y}^{\mathbf{II}} = \begin{bmatrix} Y_0^{\mathrm{II}} \\ U \\ Y_2^{\mathrm{II}} \\ U \\ \vdots \\ U \\ Y_{60}^{\mathrm{II}} \\ U \\ Y_{62}^{\mathrm{II}} \end{bmatrix}$$

where the elements labeled as U are not of interest for the computation of the 32-point DCT5. More in detail, the output vector produced by the computation of the DCT6 is

$$\boldsymbol{Y}^{\mathbf{VI}} = \begin{bmatrix} Y_0^{\mathrm{VI}} \\ Y_1^{\mathrm{VI}} \\ \vdots \\ Y_{30}^{\mathrm{VI}} \\ Y_{31}^{\mathrm{VI}} \end{bmatrix} = \begin{bmatrix} Y_0^{\mathrm{II}} \\ Y_2^{\mathrm{II}} \\ \vdots \\ Y_{60}^{\mathrm{II}} \\ Y_{62}^{\mathrm{II}} \end{bmatrix}$$

B.2.4 Definition of the output vector $\boldsymbol{Y}$

According to what is described in section 3.1, the output vector $\boldsymbol{Y}$ can be obtained from the vector $\boldsymbol{Y}^{\mathbf{VI}}$ as follows:

$$\boldsymbol{Y} = [D_{32}]\, \boldsymbol{Y}^{\mathbf{VI}}$$

B.2.5 Final Algorithm

The procedure reported above leads to an algorithm that requires 93 multiplications and 302 sums if $N1 = 7$ and $N2 = 9$ are chosen or 103 multiplications and 320 sums in case $N1 = 9$ and $N2 = 7$ are selected.

Appendix C

Direct Factorization for $N = 32$

This appendix presents the derivation of the algorithm obtained starting from the direct factorization of the 32-point DCT5.

C.1 Direct Factorization of the DCT5 ($N = 32$)

The factorization described in section 5.1 can be applied to the 32-point DCT5. More in detail, it can be written:

$$\left[C_{32}^{\mathrm{V}}\right] = \left[Q_{10}^{32}\right] \left(\left[C_{11}^{\mathrm{V}}\right] \oplus \left[C_{21}^{\mathrm{III}} \left(\frac{2}{3} \right) \right] \right) \left[B_{32}^{(C5)}\right]$$

In the following each of the terms present in the right-side of the equation will be described.

C.1.1 Thirty-two-point permutation matrix

The permutation matrix $\left[Q_{10}^{32}\right]$ can be obtained by following the procedure illustrated in section 5.1.1.

C.1.2 Eleven-point non-normalized DCT5 matrix

The eleven-point non-normalized DCT5 can be, for instance, computed by recursively adopting the factorization proposed in equation 5.1. In particular, it can be written:

$$\left[C_{11}^{\mathrm{V}}\right] = \left[Q_{3}^{11}\right] \left(\left[C_{4}^{\mathrm{V}}\right] \oplus \left[C_{7}^{\mathrm{III}} \left(\frac{2}{3} \right) \right] \right) \left[B_{11}^{(C5)}\right]$$

where:

- $\left[Q_{3}^{11}\right]$ and $\left[B_{11}^{(C5)}\right]$ can be obtained by respectively referring to sections 5.1.1 and 5.1.5.
- $\left[C_{4}^{\mathrm{V}}\right]$ is the 4-point non-normalized DCT5 matrix. The 4-point DCT5 can therefore be computed according to one of the algorithms presented in the previous chapters.

- $\left[C_7^{\mathrm{III}}\left(\frac{2}{3}\right)\right]$ can be computed according to the following formula:

$$\left[C_7^{\mathrm{III}}\left(\frac{2}{3}\right)\right] = \left[C_7^{\mathrm{III}}\right]\left[X_7^{(C3)}\left(\frac{2}{3}\right)\right]$$

  where:

  - $\left[C_7^{\mathrm{III}}\right]$ is the seven-point DCT3 matrix;
  - $\left[X_7^{(C3)}\left(\frac{2}{3}\right)\right]$ can be obtained according to what is described in section 5.1.4.

  A possible algorithm for the seven-point DCT3 can be found in [21]. This will be presented in the following.

Algorithm for the 7-point DCT3

The seven-point DCT3 can be computed according to the following factorization:

$$\left[C_7^{\mathrm{III}}\right] = \begin{bmatrix} [I_3] & & -[J_3] \\ & 1 & \\ [J_3] & & [I_3] \end{bmatrix} \begin{bmatrix} [J_4]\cdot\left[C_4^{\mathrm{V}}\right]\cdot[D_4] & \\ & \left[S_3^{\mathrm{VII}}\right]^{\mathrm{T}} \end{bmatrix} [R_7]^{\mathrm{T}}$$

where:

- $[I_n]$ and $[J_n]$ are respectively the n-point identity matrix and backward identity matrix;
- $\left[C_4^{\mathrm{V}}\right]$ is the 4-point non-normalized DCT5 matrix;
- $[D_4] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}$
- $\left[S_3^{\mathrm{VII}}\right]$ is the non-normalized 3-point DST7 matrix. This is defined as:

$$\left[S_3^{\mathrm{VII}}\right] = \begin{bmatrix} \sin\left(\frac{\pi}{7}\right) & \sin\left(\frac{2\pi}{7}\right) & \sin\left(\frac{3\pi}{7}\right) \\ \sin\left(\frac{3\pi}{7}\right) & \sin\left(\frac{6\pi}{7}\right) & \sin\left(\frac{9\pi}{7}\right) \\ \sin\left(\frac{5\pi}{7}\right) & \sin\left(\frac{10\pi}{7}\right) & \sin\left(\frac{15\pi}{7}\right) \end{bmatrix}$$

- $[R_7] = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$

According to what is described in the previous chapters, several algorithms can be adopted to compute the 4-point DCT5. On the other hand, a possible algorithm for the transposed 3-point DST7 is presented in [21]. The SFG of this algorithm is shown in figure C.1 while the value of the constants and the list of operations needed are respectively reported in tables C.1 and C.2.

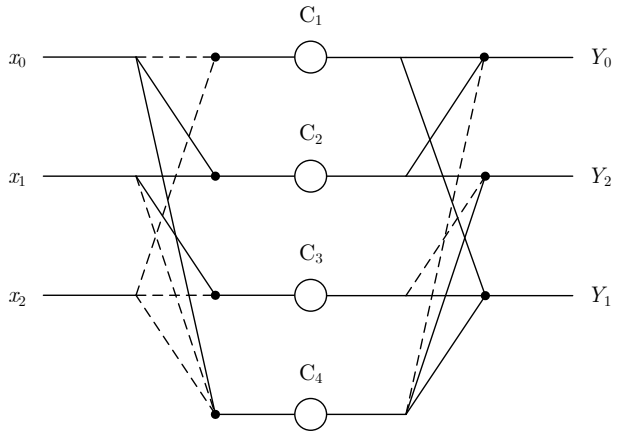


Figure C.1: *Non-normalized transposed 3-Point DST7 SFG*

Table C.1: *Constants for the transposed 3-point DST7 algorithm*

Constant	*Value*
C_1	$-0.340\,872\,93$
C_2	$0.533\,969\,36$
C_3	$0.874\,842\,29$
C_4	$0.440\,958\,55$

Table C.2: *Algorithm for the non-normalized transposed 3-point DST7*

Op. Name	*Operation*	*Op. Name*	*Operation*	*Op. Name*	*Operation*
a_0	$-x_0 - x_2$	M_2	$C_2 a_1$	a_7	$M_2 - M_3$
a_1	$x_0 + x_1$	M_3	$C_3 a_2$	a_8	$a_7 + M_4$
a_2	$x_1 - x_2$	M_4	$C_4 a_4$	Y_2	a_8
a_3	$x_0 - x_1$	a_5	$M_1 + M_2$	a_9	$M_1 + M_3$
a_4	$a_3 - x_2$	a_6	$a_5 - M_4$	a_{10}	$a_9 + M_4$
M_1	$C_1 a_0$	Y_0	a_6	Y_1	a_{10}

C.1.3 Twenty-one-point non-normalized skew-DCT3 matrix

According to what is described in [20], the 21-point non-normalized skew-DCT3 matrix ($\left[C_{21}^{\mathrm{III}}\left(\frac{2}{3}\right)\right]$) can be expressed as:

$$\left[K_7^{21}\right]\left(\left[C_7^{\mathrm{III}}\left(\frac{2}{9}\right)\right]\oplus\left[C_7^{\mathrm{III}}\left(\frac{4}{9}\right)\right]\oplus\left[C_7^{\mathrm{III}}\left(\frac{8}{9}\right)\right]\right)\left(\left[C_3^{\mathrm{III}}\left(\frac{2}{3}\right)\right]\otimes\left[I_7\right]\right)\left[B_{3,7}^{(C3)}\right]$$

where:

- $\left[K_7^{21}\right]$ is a permutation matrix;
- $\left[C_7^{\mathrm{III}}\left(\frac{2}{9}\right)\right]$, $\left[C_7^{\mathrm{III}}\left(\frac{4}{9}\right)\right]$ and $\left[C_7^{\mathrm{III}}\left(\frac{8}{9}\right)\right]$ are non-normalized 7-point skew-DCT3 matrices;
- $\oplus$ is the direct sum operator;
- $\left[C_3^{\mathrm{III}}\left(\frac{2}{3}\right)\right]$ is a non-normalized 3-point skew-DCT3 matrix;
- $\otimes$ is the symbol of the kronecker product;
- $[I_7]$ is the 7-point identity matrix;
- $\left[B_{3,7}^{(C3)}\right]$ is a base change matrix.

Permutation matrix $\left[K_7^{21}\right]$

The permutation matrix $\left[K_7^{21}\right]$ is such that:

$$\left[K_7^{21}\right]=\left([I_3]\oplus[J_3]\oplus[I_3]\oplus[J_3]\oplus[I_3]\oplus[J_3]\oplus[I_3]\right)\left[L_7^{21}\right]$$

where $\left[L_7^{21}\right]$ is defined as

$$\left[L_7^{21}\right]:\quad\begin{array}{rll} i & \mapsto 7i \bmod 20 & \text{for } 0\le i<20\\ 20 & \mapsto 20 & \end{array}$$

Non-normalized 7-point skew-DCT3 matrices

The considered non-normalized 7-point skew-DCT3 matrices are:

$$\left[C_7^{\mathrm{III}}\left(\frac{2}{9}\right)\right]=\left[C_7^{\mathrm{III}}\right]\left[X_7^{(C3)}\left(\frac{2}{9}\right)\right]$$
$$\left[C_7^{\mathrm{III}}\left(\frac{4}{4}\right)\right]=\left[C_7^{\mathrm{III}}\right]\left[X_7^{(C3)}\left(\frac{4}{9}\right)\right]$$
$$\left[C_7^{\mathrm{III}}\left(\frac{8}{9}\right)\right]=\left[C_7^{\mathrm{III}}\right]\left[X_7^{(C3)}\left(\frac{8}{9}\right)\right]$$

where the 7-point DCT3 can be computed according to the algorithm described in section C.1.2 and $\left[X_7^{(C3)}\left(\frac{2}{9}\right)\right]$, $\left[X_7^{(C3)}\left(\frac{4}{9}\right)\right]$, $\left[X_7^{(C3)}\left(\frac{8}{9}\right)\right]$ can be found by following what is illustrated in section 5.1.4.

Non-normalized 3-point skew-DCT3 matrix

The non-normalized 3-point skew-DCT3 matrix can be obtained as:

$$\left[C_3^{\mathrm{III}}\left(\frac{2}{3}\right)\right] = \left[C_3^{\mathrm{III}}\right]\left[X_3^{(C3)}\left(\frac{2}{3}\right)\right]$$

The 3-point DCT3 can be calculated by adopting the algorithm described in [21]. The SFG of this algorithm is shown in figure C.2 while the value of the constants and the list of operations needed are respectively reported in tables C.3 and C.4.

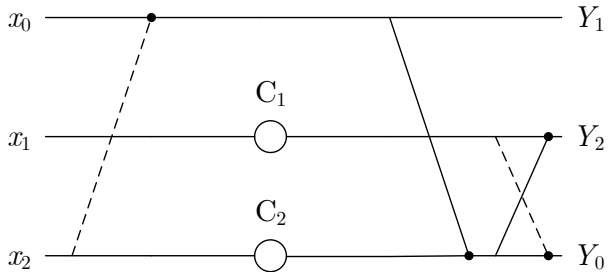


Figure C.2: *Non-normalized 3-Point DCT3 SFG*

Table C.3: *Constants for the 3-point DCT3 algorithm*

Constant	*Value*
C_1	$-0.866\,025\,40$
C_2	$1.500\,000\,00$

Table C.4: *Algorithm for the non-normalized 3-point DCT3*

Op. Name	*Operation*	*Op. Name*	*Operation*	*Op. Name*	*Operation*
a_0	$x_0 - x_2$	M_2	$C_2 x_2$	a_3	$a_1 - M_1$
Y_1	a_0	a_1	$a_0 + M_2$	Y_2	a_2
M_1	$C_1 x_1$	a_2	$a_1 + M_1$	Y_0	a_3

The matrix $\left[X_3^{(C3)}\left(\frac{2}{3}\right)\right]$ can be derived according to what is described in section 5.1.4.

Base change matrix $\left[B_{3,7}^{(C3)}\right]$

The base change matrix is defined as:

$$\left[B_{3,7}^{(C3)}\right] = \left([I_7] \oplus \left([I_2] \otimes \mathrm{diag}(1,2,\ldots,2)\right)\right) \begin{bmatrix} [I_7] & -[Z_7] & \begin{bmatrix} I_7' \end{bmatrix} \\ & [I_7] & -[Z_7] \\ & & [I_7] \end{bmatrix}$$

where $\left[I_m^{'}\right] = \text{diag}(0,1,\ldots,1)$ and

$$[Z_m] = \begin{bmatrix} & & & 0 \\ & & 0 & 1 \\ & \iddots & \iddots & \\ 0 & 1 & & \end{bmatrix}$$

C.1.4 Thirty-two-point base change matrix

The 32-point base change matrix is:

$$\left[B_{32}^{(C5)}\right] = \left[\begin{array}{ccc|cc} 1 & & & 1 & \\ & [I_{10}] & [J_{10}] & & [I_{10}] \\ \hline & & & -1/2 & \\ & [I_{21}] & & & -[I_{10}] \\ & & & & -[J_{10}] \end{array}\right]$$

Appendix D

16-Point DCT5 Algorithm

This appendix presents a MatLab implementation of both the floating point and fixed-point versions of the algorithm obtained by following the procedure described in section A.1.1. More in detail, the algorithm is obtained by following that procedure and making some manipulations in order to avoid cascading the multiplier that is needed for the pre-normalization.

D.1 MatLab Implementation (Floating-Point)

```
function [OUTPUT]=Burrus_DCT5_N16_simpl_floating(INPUT)


OUTPUT=zeros(16,1);
s1 = INPUT(2)+INPUT(7);
s2 = INPUT(2)-INPUT(6);
s3 = s1+INPUT(6);
s4 = INPUT(7)-INPUT(6);
s5 = INPUT(16)+INPUT(4);
s6 = INPUT(16)-INPUT(14);
s7 = s5+INPUT(14);
s8 = INPUT(4)-INPUT(14);
s9 = INPUT(9)+INPUT(15);
s10 = INPUT(9)-INPUT(10);
s11 = s9+INPUT(10);
s12 = INPUT(15)-INPUT(10);
s13 = INPUT(5)+INPUT(8);
s14=INPUT(5)-INPUT(12);
s15 = s13+INPUT(12);
s16 =INPUT(8)-INPUT(12);
s17 =INPUT(3)+INPUT(13);
s18 = INPUT(3)-INPUT(11);
s19 = s17+INPUT(11);
s20 = INPUT(13)-INPUT(11);
```

```
s21 =s3+s7;
s22 = s3-s19;
s23=s21+s11;
s24= s7-s19;
s25=s23+s15;
s26=s11-s19;
s27=s25+s19;
s28=s15-s19;
s29=s2+s6;
s30=s2-s18;
s31=s29+s10;
s32=s6-s18;
s33=s31+s14;
s34=s10-s18;
s35=s33+s18;
s36=s14-s18;
s37=s4+s8;
s38=s4-s20;
s39=s37+s12;
s40=s8-s20;
s41=s39+s16;
s42=s12-s20;
s43=s41+s20;
s44=s16-s20;
m42=0.254000254000381*INPUT(1);
m1=0.359210604053550*s27;
s45=m1+m42;
m2=-0.371184290855335*s27;
s46=s35+s43;
m3=0.090171514658534*s35;
m4=0.229210235490481*s43;
m5=-0.106460583383005*s46;
s47=m3+m5;
s48=m4+m5;
s49=s22+s26;
s50=s24+s28;
s51=s22+s24;
s52=s26+s28;
s53=s49+s50;
m6=-0.036100033196720*s22;
m7=-0.078100047944899*s24;
m8=-0.116772960840795*s51;
m9=0.286861819809839*s26;
m10=-0.280541341615715*s28;
m11=-0.091988811024145*s52;
m12=0.060884183013370*s49;
m13=0.255757191799065*s50;
m14=-0.021575920589499*s53;
s54=m6+m12;
```

```
s55=m9+m12;
s56=m7+m13;
s57=m10+m13;
s58=m8+m14;
s59=m11+m14;
s60=s54+s58;
s61=s56+s58;
s62=s55+s59;
s63=s57+s59;
s64=s30+s38;
s65=s32+s40;
s66=s34+s42;
s67=s36+s44;
s68=s30+s34;
s69=s32+s36;
s70=s38+s42;
s71=s40+s44;
s72=s64+s66;
s73=s65+s67;
s74=s30+s32;
s75=s34+s36;
s76=s68+s69;
s77=s38+s40;
s78=s42+s44;
s79=s70+s71;
s80=s64+s65;
s81=s66+s67;
s82=s72+s73;
m15=0.472972261217078*s30;
m16=0.477863351912534*s32;
m17=-0.138340176742113*s74;
m18=-1.062784397189376*s34;
m19=-0.910707361134637*s36;
m20=0.723261222018955*s75;
m21=0.388629137543990*s68;
m22=0.049105962373569*s69;
m23=-0.204531229038880*s76;
m24=-0.094201906554648*s38;
m25=0.721961433523775*s40;
m26=-0.416450310827657*s77;
m27=0.226075551590858*s42;
m28=0.441581943060109*s44;
m29=-0.531586674286976*s78;
m30=-0.020934456904672*s70;
m31=-0.326445579600790*s71;
m32=0.259083404324019*s79;
m33=-0.126256784887477*s64;
m34=-0.399941595145436*s65;
m35=0.184930162523257*s80;
```

```
m36=0.278902948532839*s66;
m37=0.156375139358176*s67;
m38=-0.063891515910660*s81;
m39=-0.122564893546439*s72;
m40=0.092446539075740*s73;
m41=-0.018184058428379*s82;
s83=m15+m33;
s84=m24+m33;
s85=m16+m34;
s86=m25+m34;
s87=m17+m35;
s88=m26+m35;
s89=m18+m36;
s90=m27+m36;
s91=m19+m37;
s92=m28+m37;
s93=m20+m38;
s94=m29+m38;
s95=m21+m39;
s96=m30+m39;
s97=m22+m40;
s98=m31+m40;
s99=m23+m41;
s100=m32+m41;
s101=s83+s95;
s102=s89+s95;
s103=s85+s97;
s104=s91+s97;
s105=s87+s99;
s106=s93+s99;
s107=s84+s96;
s108=s90+s96;
s109=s86+s98;
s110=s92+s98;
s111=s88+s100;
s112=s94+s100;
s113=s101+s105;
s114=s103+s105;
s115=s102+s106;
s116=s104+s106;
s117=s107+s111;
s118=s109+s111;
s119=s108+s112;
s120=s110+s112;
s121=s45+m2;
s122=s121+s60;
s123=s121-s60;
s124=s121+s61;
s125=s123-s61;
```

```
s126=s121+s62;
s127=s125-s62;
s128=s121+s63;
s129=s127-s63;
s130=s47+s113;
s131=s47-s113;
s132=s47+s114;
s133=s131-s114;
s134=s47+s115;
s135=s133-s115;
s136=s47+s116;
s137=s135-s116;
s138=s48+s117;
s139=s48-s117;
s140=s48+s118;
s141=s139-s118;
s142=s48+s119;
s143=s141-s119;
s144=s48+s120;
s145=s143-s120;
s146=s122+s130;
s147=s122-s130;
s148=s122+s138;
s149=s147-s138;
s150=s124+s132;
s151=s124-s132;
s152=s124+s140;
s153=s151-s140;
s154=s126+s134;
s155=s126-s134;
s156=s126+s142;
s157=s155-s142;
s158=s128+s136;
s159=s128-s136;
s160=s128+s144;
s161=s159-s144;
s162=s129+s137;
s163=s129-s137;
s164=s129+s145;
s165=s163-s145;
m43=s45*0.707106781186547;

OUTPUT(1)=m43;
OUTPUT(2)=s165;
OUTPUT(3)=s161;
OUTPUT(4)=s146;
OUTPUT(5)=s157;
OUTPUT(6)=s164;
OUTPUT(7)=s162;
```

```
OUTPUT(8)=s154;
OUTPUT(9)=s153;
OUTPUT(10)=s152;
OUTPUT(11)=s160;
OUTPUT(12)=s156;
OUTPUT(13)=s158;
OUTPUT(14)=s148;
OUTPUT(15)=s150;
OUTPUT(16)=s149;

end
```

D.2 MatLab Implementation (Fixed-Point)

```
function [OUTPUT]=Burrus_DCT5_N16_simpl(INPUT)


OUTPUT=zeros(16,1);
s1 = INPUT(2)+INPUT(7);
s2 = INPUT(2)-INPUT(6);
s3 = s1+INPUT(6);
s4 = INPUT(7)-INPUT(6);
s5 = INPUT(16)+INPUT(4);
s6 = INPUT(16)-INPUT(14);
s7 = s5+INPUT(14);
s8 = INPUT(4)-INPUT(14);
s9 = INPUT(9)+INPUT(15);
s10 = INPUT(9)-INPUT(10);
s11 = s9+INPUT(10);
s12 = INPUT(15)-INPUT(10);
s13 = INPUT(5)+INPUT(8);
s14=INPUT(5)-INPUT(12);
s15 = s13+INPUT(12);
s16 =INPUT(8)-INPUT(12);
s17 =INPUT(3)+INPUT(13);
s18 = INPUT(3)-INPUT(11);
s19 = s17+INPUT(11);
s20 = INPUT(13)-INPUT(11);
s21 =s3+s7;
s22 = s3-s19;
s23=s21+s11;
s24= s7-s19;
s25=s23+s15;
s26=s11-s19;
s27=s25+s19;
s28=s15-s19;
s29=s2+s6;
s30=s2-s18;
s31=s29+s10;
```

```
s32=s6-s18;
s33=s31+s14;
s34=s10-s18;
s35=s33+s18;
s36=s14-s18;
s37=s4+s8;
s38=s4-s20;
s39=s37+s12;
s40=s8-s20;
s41=s39+s16;
s42=s12-s20;
s43=s41+s20;
s44=s16-s20;
m42=130*INPUT(1);
m1=183*s27;
s45=m1+m42;
m2=-190*s27;
s46=s35+s43;
m3=46*s35;
m4=117*s43;
m5=-55*s46;
s47=m3+m5;
s48=m4+m5;
s49=s22+s26;
s50=s24+s28;
s51=s22+s24;
s52=s26+s28;
s53=s49+s50;
m6=-18*s22;
m7=-40*s24;
m8=-60*s51;
m9=147*s26;
m10=-144*s28;
m11=-47*s52;
m12=31*s49;
m13=131*s50;
m14=-11*s53;
s54=m6+m12;
s55=m9+m12;
s56=m7+m13;
s57=m10+m13;
s58=m8+m14;
s59=m11+m14;
s60=s54+s58;
s61=s56+s58;
s62=s55+s59;
s63=s57+s59;
s64=s30+s38;
s65=s32+s40;
```

```
s66=s34+s42;
s67=s36+s44;
s68=s30+s34;
s69=s32+s36;
s70=s38+s42;
s71=s40+s44;
s72=s64+s66;
s73=s65+s67;
s74=s30+s32;
s75=s34+s36;
s76=s68+s69;
s77=s38+s40;
s78=s42+s44;
s79=s70+s71;
s80=s64+s65;
s81=s66+s67;
s82=s72+s73;
m15=242*s30;
m16=245*s32;
m17=-71*s74;
m18=-544*s34;
m19=-466*s36;
m20=370*s75;
m21=199*s68;
m22=25*s69;
m23=-105*s76;
m24=-48*s38;
m25=370*s40;
m26=-213*s77;
m27=116*s42;
m28=226*s44;
m29=-272*s78;
m30=-11*s70;
m31=-167*s71;
m32=133*s79;
m33=-65*s64;
m34=-205*s65;
m35=95*s80;
m36=143*s66;
m37=80*s67;
m38=-33*s81;
m39=-63*s72;
m40=47*s73;
m41=-9*s82;
s83=m15+m33;
s84=m24+m33;
s85=m16+m34;
s86=m25+m34;
s87=m17+m35;
```

```
s88=m26+m35;
s89=m18+m36;
s90=m27+m36;
s91=m19+m37;
s92=m28+m37;
s93=m20+m38;
s94=m29+m38;
s95=m21+m39;
s96=m30+m39;
s97=m22+m40;
s98=m31+m40;
s99=m23+m41;
s100=m32+m41;
s101=s83+s95;
s102=s89+s95;
s103=s85+s97;
s104=s91+s97;
s105=s87+s99;
s106=s93+s99;
s107=s84+s96;
s108=s90+s96;
s109=s86+s98;
s110=s92+s98;
s111=s88+s100;
s112=s94+s100;
s113=s101+s105;
s114=s103+s105;
s115=s102+s106;
s116=s104+s106;
s117=s107+s111;
s118=s109+s111;
s119=s108+s112;
s120=s110+s112;
s121=s45+m2;
s122=s121+s60;
s123=s121-s60;
s124=s121+s61;
s125=s123-s61;
s126=s121+s62;
s127=s125-s62;
s128=s121+s63;
s129=s127-s63;
s130=s47+s113;
s131=s47-s113;
s132=s47+s114;
s133=s131-s114;
s134=s47+s115;
s135=s133-s115;
s136=s47+s116;
```

```
s137=s135-s116;
s138=s48+s117;
s139=s48-s117;
s140=s48+s118;
s141=s139-s118;
s142=s48+s119;
s143=s141-s119;
s144=s48+s120;
s145=s143-s120;
s146=s122+s130;
s147=s122-s130;
s148=s122+s138;
s149=s147-s138;
s150=s124+s132;
s151=s124-s132;
s152=s124+s140;
s153=s151-s140;
s154=s126+s134;
s155=s126-s134;
s156=s126+s142;
s157=s155-s142;
s158=s128+s136;
s159=s128-s136;
s160=s128+s144;
s161=s159-s144;
s162=s129+s137;
s163=s129-s137;
s164=s129+s145;
s165=s163-s145;
m43=floor(bitsra(s45*362,9));

OUTPUT(1)=m43;
OUTPUT(2)=s165;
OUTPUT(3)=s161;
OUTPUT(4)=s146;
OUTPUT(5)=s157;
OUTPUT(6)=s164;
OUTPUT(7)=s162;
OUTPUT(8)=s154;
OUTPUT(9)=s153;
OUTPUT(10)=s152;
OUTPUT(11)=s160;
OUTPUT(12)=s156;
OUTPUT(13)=s158;
OUTPUT(14)=s148;
OUTPUT(15)=s150;
OUTPUT(16)=s149;

end
```

Bibliography

[1] K. R. Rao, P. Yip, V. Britanak, *Discrete Cosine and Sine Transforms: General Properties, Fast Algorithms and Approximations* Elsevier, 2007.

[2] S. Winograd, "On Computing the Discrete Fourier Transform" in *Mathematics of Computation*, v. 32, n. 141, pp. 175–199, Jan. 1978.

[3] ——, *Arithmetic Complexity of Computations* Philadelphia, Pennsylvania, SIAM CBMS-NSF, 1980.

[4] K. R. Rao, D. N. Kim, J. J. Hwang, *Fast Fourier Transform: Algorithms and Applications* Springer, 2010.

[5] C. S. Burrus, M. Frigo, S. Johnson, M. Pueschel, I. Selesnick, *Fast Fourier Transforms* Rice University, Houston, Texas, Connexions, 2008.

[6] R. E. Blahut, *Fast Algorithms for Signal Processing* Cambridge, United Kingdom, Cambridge University Press, 2010.

[7] U. Meyer-Baese, *Digital Signal Processing with Field Programmable Gate Arrays* Heidelberg, Germany, Springer, 2014.

[8] W. W. W. Community. [Online]: https://wikipedia.org

[9] E. Chu, *Discrete and Continuous Fourier Transforms: Analysis, Applications and Fast Algorithms* Boca Raton, Florida, CRC Press, 2008.

[10] R. Lyons. Four ways to compute an inverse fft using the forward fft algorithm. [Online]: https://www.dsprelated.com/showarticle/800.php

[11] P. Duhamel, B. Piron, J. M. Etcheto, "On computing the inverse DFT" in *IEEE Transactions on Acoustics, Speech, and Signal Processing*, v. 36, n. 2, pp. 285–286, Feb. 1988.

[12] H. J. Nussbaumer, *Fast Fourier Transform and Convolution Algorithms* Heidelberg, Germany, Springer-Verlag, 1981.

[13] M. Masera, M. Martina, G. Masera, "Odd type DCT/DST for video coding: Relationships and low-complexity implementations" in *2017 IEEE International Workshop on Signal Processing Systems (SiPS)*, Oct. 2017, pp. 1–6.

[14] Y. A. Reznik, "Relationship between DCT-II, DCT-VI, and DST-VII transforms" in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, May 2013, pp. 5642–5646.

[15] M. T. Heideman, "Computation of an odd-length DCT from a real-valued DFT of the same length" in *IEEE Transactions on Signal Processing*, v. 40, n. 1, pp. 54–61, Jan. 1992.

[16] G. Bi, Y. Zeng, *Transforms and Fast Algorithms for Signal Analysis and Representations* New York, Springer Science & Business Media, 2004.

[17] C. S. Burrus. Programs for short ffts. [Online]: https://cnx.org/contents/eL72ctwp@4/Appendix-4-Programs-for-Short-FFTs

[18] G. Bi, "Index mapping for prime factor algorithm of discrete cosine transform" in *Electronics Letters*, v. 35, n. 3, pp. 198–200, 1999.

[19] K. R. Rao, P. Yip, *Discrete Cosine Transform: Algorithms, Advantages, Applications* San Diego, California, Academic Press, Inc., 1990.

[20] M. Puschel, J. M. F. Moura, "Algebraic Signal Processing Theory: Cooley-Tukey Type Algorithms for DCTs and DSTs" in *IEEE Transactions on Signal Processing*, v. 56, n. 4, pp. 1502–1521, Apr. 2008.

[21] J. Kello, "DCT-V for Video Coding: A reconfigurable implementation for length 32 and 4" Tesi di laurea, 2018.

[22] I. W. Selesnick, C. S. Burrus, "Automatic generation of prime length FFT programs" in *IEEE Transactions on Signal Processing*, v. 44, n. 1, pp. 14–24, Jan. 1996.

[23] I. Selesnick. Fft programs for prime lengths. [Online]: https://www.ece.rice.edu/dsp/software/pfft.shtml

[24] J. H. McClellan, C. M. Rader, *Number Theory in Digital Signal Processing* Englewood Cliffs, New Jersey, Prenctice-Hall, 1979.

[25] M. Puschel, J. M. F. Moura, J. R. Johnson, D. Padua, M. M. Veloso, B. W. Singer, J. Xiong, F. Franchetti, A. Gacic, Y. Voronenko, K. Chen, R. W. Johnson, N. Rizzolo, "SPIRAL: Code Generation for DSP Transforms" in *Proceedings of the IEEE*, v. 93, n. 2, pp. 232–275, Feb 2005.