The Design of the Algorithms for the Signal Processing

KUPCZAK, Marek

Abstract: This contribution describes the results of the analysis and the implementation of the new designed algorithms to the digital signal processor (DSP). Designed algorithms are based on the discrete wavelet transforms (DWT). The algorithms, based on the one-dimensional DWT, process data in the vector form. The algorithms, based on the two-dimensional DWT, process data in the matrix form. Because the DWTs are the linear transforms, they can be written in the matrix form. These matrices are sparse. Due to many zero elements in each column of the matrix $W$, it’s evident that it’s not so efficient to calculate the approximation and the detail parts as the multiplication between the matrix $W$ and the input vector, respectively matrix. These algorithms, used in the digital signal processors, radically reduce the process time for the digital signal processing.

The presented results have been obtained with the support of the Czech Ministry of Education, Youth and Sports, during completing research project MSM 272300012.

Keywords: algorithms, Discrete Wavelet Transforms, Digital Signal Processing, DSP, DWT

1 VisualDSP++

VisualDSP++ is the software product of the Analog Devices. It’s assigned for the development of the DSP applications, it means for the digital signal processors (DSP). VisualDSP++ contains the integrated environment for the developing and the debugging of the applications. The user interface provides the access to the tools for the code developing. Among these tools belong C/C++ compiler, assembler, linker, libraries containing the mathematical functions, functions for the digital signal processing, I/O functions and so on.

For the developing of the own application we have three choices. We can use C/C++ language, assembly language or we can use both. There are several criteria which decide for the selection of the programming language, for example the performance and memory requirements, code portability or the existence of the already written codes.

The codes, written in the assembly language, will be generally always more efficient and they take less program memory space than codes written in C language. But the writing of the codes in C language has its reasons. One of them is that there are huge quantity of the already written codes in C or C++ language, and there aren’t very efficient programs for the compilation of the assembly codes between the different DSP platforms. Due to the C/C++ compiler there is a good portability of the codes written in C/C++ language between the DSP platforms.

The best way of the code writing is the use of the both programming languages. Very efficient way is the identification of the C parts by the simulation instrument, which take the biggest process time, and to write these parts in the assembly language.
2 C/C++ library

The C/C++ library is the collection of the functions, macros and the class templates that can be called from the source files. Concretely, C/C++ library for the ADSP-21xxx was used. The library can divided into four partial libraries: C library, C++ library, DSP library, I/O library. These libraries contain functions for the memory allocation, functions for the char and string routines, mathematical functions, functions for the digital signal processing, I/O functions.

3 The newly created functions

DSP library contains functions for the digital signal processing. It contains the functions for the computation of coherence, covariance and correlation (autocoh, crosscoh, autocorr, crosscorr), function for the computation with the complex data (cabsf, cexpf), functions for the filters (biquad, fir, iir), functions for the computation of the direct and the inverse Fast Fourier Transform (cfftN, ifftN, rfftN) and functions for the computation with the time windows (gen_bartlett, gen_blackman, gen_gaussian, gen_hamming, gen_hanning, gen_harris, gen_kaiser, gen_rectangular, gen_triangle).

But this library doesn’t contain functions for the signal processing and the signal analysis based on the discrete wavelet transforms (DWTs). That is why six new algorithms were designed. These algorithms can compute one-level and multi-level decomposition and reconstruction of the one-dimensional signals (in the vector form) and one-level decomposition and reconstruction of the two-dimensional signals (in the matrix form).

Discrete Wavelet Transforms

This work deals only with real dyadic wavelets with compacted support that create the orthogonal system. Discrete wavelet transforms map discrete data $y_i$, $i = 1, \ldots, N$ from the time domain into the time-frequency domain (wavelet domain) $Y_i$, $i = 1, \ldots, N$. The result is a vector (a matrix, respectively) of the same size as the input vector $y$ (a matrix $Y$, respectively). Discrete wavelet transforms belong between linear transforms, so they can be defined by the matrices of the size $N \times N$.

Any function $f \in \ell_2(\mathbb{R})$ can be decomposed into the form

$$ f(t) = \sum_{j,k} d_{jk} \psi_{jk}(t) $$

(1)

where $\psi$ is a wavelet function and $\varphi$ is a scaling function. These functions create an orthogonal base.

This form corresponds to the multi-level decomposition $\ell_2(\mathbb{R}) = \bigoplus_{j=-\infty}^{\infty} W_j$. It can be further modify to the form (2) by any arbitrary index $j_0$.

$$ f(t) = \sum_{k} c_{j_0 k} \varphi_{j_0 k}(t) + \sum_{j \geq j_0} \sum_{k} d_{jk} \psi_{jk}(t) $$

(2)

The first sum of the form (2) presents an orthogonal projection $P_{j_0}$ of the function $f$ into the subspace of the scaling functions $V_{j_0}$, the other sums are the orthogonal additions of the subspaces of the wavelet functions $W_j$.

The decomposition algorithm (cascade algorithm and dyadic wavelets) can be written in the matrix form:

Let the length of the input signal $f$ is $N = k \cdot 2^m$, let the length of the vector $h = \{ h_s, s \in \mathbb{Z} \}$ is $K$ and let the individual filter coefficients $h_s$ are non-zero.
Further we can define a matrix $H_k$ of the size $\left(2^{m-k} \times 2^{m-k+1}\right)$, $k = 1, 2, \ldots$ with individual elements by the form (3) for a concrete row and column index $(i, j)$.

$$h_s, s = (K-1) + (i-1) - 2(j-1) \mod 2^{m-k+1}$$  \hspace{1cm} (3)

It’s evident that $i$th row is practically only the first row shifted to the right by $2(i-1)$ units. This property is caused by using the modulo operator in the form (3).

Simply we can define a matrix $G_k$ by using the vector $g$. The filter coefficients of the vectors $g$ and $h$ are bound together by the form $g_s = (-1)^s h_{K+1-s}$. The total matrix is then $W_k = \begin{pmatrix} H_k \\ G_k \end{pmatrix}$.

Because the matrix $W$ is orthogonal, then it’s very easy to find the inverse matrix because $W^{-1} = W^T$ (only for real matrices).

The first step of the signal decomposition is defined as $Y = \begin{pmatrix} cA_1 \\ cD_1 \end{pmatrix} = W \cdot y$, where $cA_1$ is the vector of the approximation coefficients and $cD_1$ is the vector of the detail coefficients. The process of the decomposition is able to iterate. The consequence of this process is shown in Fig. 1.

![Figure 1 – The consequence of using the multiresolution analysis](image)

**The case study**

The theoretical ground is used for the realization of the three steps of the one-dimensional signal decomposition and reconstruction to the approximation and detail parts, it means to the vectors containing the approximation coefficients and detail coefficients. The signal contains 1024 samples and it was obtained for the demo data file `leleccum.m` of program MATLAB, which contains 4320 samples. This signal is shown in Fig. 2.
To analyse this signal, the wavelet basis Daubechies2 was chosen which contains four coefficients. It means that two vectors $h$ and $g$ were created. These vectors contain four filter coefficients.

$$h = (h_0, h_1, h_2, h_3)$$  \hspace{1cm} (4)

$$g = (g_0, g_1, g_2, g_3) = (h_3, -h_2, h_1, -h_0)$$  \hspace{1cm} (5)

where

$$h_0 = 0.4829629131445342$$  \hspace{1cm} (6)

$$h_1 = 0.8365163037378080$$  \hspace{1cm} (7)

$$h_2 = 0.2241438680420134$$  \hspace{1cm} (8)

$$h_3 = -0.1294095225512604$$  \hspace{1cm} (9)

By using these vectors we can construct the matrix $W_1$ for the first decomposition level of the input signal which contains 1024 samples. So the size of the matrix $W_1$ is $1024 \times 1024$. The output signal $Y_{1024}$, which contains 512 approximation coefficients and 512 detail coefficients, could be simply obtain by multiplication between this matrix and the input signal, so

$$Y_1 = \begin{pmatrix} cA_i \\ cD_i \end{pmatrix} = W_1 \cdot y_i$$  \hspace{1cm} (10)

The inverse matrix to the matrix $W_1$ is simply given by the form (11), because this matrix is real and orthogonal

$$W_1^{-1} = W_1^T$$  \hspace{1cm} (11)

The graphical representation of the decomposition results on the first level in the time-frequency domain is shown in Fig. 3 and 4. The reconstruction of the signal on the first level is same with the input signal.
The second step of the signal decomposition and signal reconstruction is analogous, so we create the matrix \( W_2 \) or respectively the inverse matrix \( W_2^\top \) of the size \( 512 \times 512 \). The approximation and detail coefficients of the second signal decomposition are given by the form

\[
Y_2 = \begin{pmatrix} cA_2 \\ cD_2 \end{pmatrix} = W_2 \cdot cA_1 \tag{13}
\]

The third step of the signal decomposition and signal reconstruction is analogous too. The size of matrices \( W_3 \) and \( W_3^\top \) is \( 256 \times 256 \). The approximation and detail coefficients of the second signal decomposition are given by the form

\[
Y_3 = \begin{pmatrix} cA_3 \\ cD_3 \end{pmatrix} = W_3 \cdot cA_3 \tag{14}
\]

The graphical representation of the decomposition results on the second level in the time-frequency domain is shown in Fig. 5 and 6. The reconstructed signal is same with Fig. 3. The graphical representation of the decomposition results on the third level in the time-frequency domain is shown in Fig. 7 and 8. The reconstructed signal is same with Fig. 5. In Fig. 9 is shown the comparison of the signal decomposition results.
Figure 5 – The approximation part on the second decomposition level

Figure 6 – The detail part on the second decomposition level

Figure 7 – The approximation part on the third decomposition level

Figure 8 – The detail part on the third decomposition level
If we consider that the sampling frequency is \( f_s = 100 \text{ Hz} \), so we can analyze the input signal approximately up to the Nyquist frequency \( f_N = \frac{f_s}{2} = 50 \text{ Hz} \).

Very important property of the wavelet transforms is that we don’t analyze the signal only in the frequency domain like Fourier transform but we obtain well localized signal in the time domain too. It means we localize the input signal in the time-frequency domain.

It’s evident that disadvantage of the process is that to obtain the approximation and the detail coefficients we have to multiple the huge matrix with the data vector or data matrix, respectively. But the transform matrices are sparse.
\[
W = \begin{pmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 &
\[ cA_i = \sum_{j=0}^{K-1} h_j \cdot y_{pozice+j} \]

\[ cD_i = \sum_{j=0}^{K-1} g_j \cdot y_{pozice+j} \]  

(15)

To calculate each individual coefficient of signal decomposition or reconstruction, we don’t have to multiply the huge \textit{sparse} matrix with the input data vector or matrix, but it’s possible to calculate these coefficients as the multiply-accumulate cycle between the filter coefficients and the individual data vector or matrix, respectively. The length of the total sum of the operations for the computation of each approximation or detail coefficient isn’t \(N\) (ordinarily hundreds or thousands of data), but is same with the length of the filter vectors \(h\) and \(g\) (ordinarily units of data). By this approach we obviate redundant operations like the multiplication with zero elements of the transform matrix \(W\) or \(W^{-1}\), respectively.

4 Conclusions

The six new algorithms for the digital signal processing were designed and implemented to the integrated software product VisualDSP++. The programming language was selected C language for its good portability between different DSP platforms.

The first four algorithms were designed for one- and multi-level decomposition and reconstruction of the one-dimensional signal presented in the vector form.

The second two algorithms were designed for one-level decomposition and reconstruction of the two-dimensional signal presented in the matrix form.

These algorithms are based on the discrete wavelet transforms. The selected wavelet bases were real dyadic wavelets because the created transform matrices are orthogonal and the inverse transform matrix is same as the transpose matrix.

Because the transform matrices are going to be very huge but on the other hand these matrices are \textit{sparse} the new approach for computation of the approximation and detail coefficients was designed. In future these algorithms will implemented to the digital signal processor and check their functionality.

5 References


