Code generator for implementing dual tree complex wavelet transform on reconfigurable architectures for mobile applications

Ferhat Canbay[1], *Vecdi Emre Levent*[2], *Gorkem Serbes*[3], *H. Fatih Ugurdag*[4], *Sezer Goren*[5], *Nizamettin Aydin*[1] ✉

[1]*Computer Engineering, Yildiz Technical University, Istanbul 34220, Turkey*
[2]*Computer Engineering, Ozyegin University, Istanbul 34794, Turkey*
[3]*Biomedical Engineering, Yildiz Technical University, Istanbul 34220, Turkey*
[4]*Electrical and Electronics Engineering, Ozyegin University, Istanbul 34794, Turkey*
[5]*Computer Engineering, Yeditepe University, Istanbul 34755, Turkey*
✉ *E-mail: naydin@yildiz.edu.tr*

Published in Healthcare Technology Letters; Received on 1st May 2016; Revised on 27th June 2016; Accepted on 29th June 2016

The authors aimed to develop an application for producing different architectures to implement dual tree complex wavelet transform (DTCWT) having near shift-invariance property. To obtain a low-cost and portable solution for implementing the DTCWT in multi-channel real-time applications, various embedded-system approaches are realised. For comparison, the DTCWT was implemented in *C* language on a personal computer and on a PIC microcontroller. However, in the former approach portability and in the latter desired speed performance properties cannot be achieved. Hence, implementation of the DTCWT on a reconfigurable platform such as field programmable gate array, which provides portable, low-cost, low-power, and high-performance computing, is considered as the most feasible solution. At first, they used the system generator DSP design tool of Xilinx for algorithm design. However, the design implemented by using such tools is not optimised in terms of area and power. To overcome all these drawbacks mentioned above, they implemented the DTCWT algorithm by using Verilog Hardware Description Language, which has its own difficulties. To overcome these difficulties, simplify the usage of proposed algorithms and the adaptation procedures, a code generator program that can produce different architectures is proposed.

1. Introduction: Physiological systems, which have inherent time-varying characteristics, produce non-stationary biomedical signals (BSs) carrying vital information about the state of human body. In the literature, classical Fourier transform (FT) is frequently used in the analysis of BSs [1]. However, in FT the time information is lost, and therefore it shows limited performance while processing non-stationary parts of BSs. Short time FT (STFT) is employed as a solution for the lack of time-information drawback of FT in various studies [2] by shifting a window onto the time axis and applying FT only to the signal component beneath this window, whereas, in the STFT, analysis window has fixed length for the entire time-axis resulting in a fixed time–frequency resolution. Continuous wavelet transform (WT) is frequently used in order to attain an adaptive time–frequency resolution by using wavelets with varying sizes; low frequencies are analysed over wide time wavelets, and high frequencies over narrow time wavelets. However, the complex WT (CWT) is computationally expensive and produces a vast amount of data at the end of analysis. Therefore, in real-time applications, in order to reduce memory requirements and to increase the computation speed of the analysis, discrete WT (DWT), in which the scale and translation parameters are discretised, is commonly employed [3]. Previously, DWT was used in the pre-processing, de-noising, and feature extraction steps of BSs analysis [4]. However, the DWT suffers from being shift-invariant and this results in corruptive sensitivity to the phase-shifts occurring in the input signal. As an example, when the DWT is used as a de-noising operator for embolic signals in [5], the distorted coefficients obtained with the DWT decreases the performance. Dual tree CWT (DTCWT), which is an improved version of the DWT with limited redundancy, is proposed in [6] to overcome the lack of shift-invariance property of the classical DWT. The DTCWT consists of two separate discrete wavelet trees (real and imaginary).

As a result of improvements in technology, continuous monitoring of the health condition of impaired people or people requiring special health treatments have become an active research area. Traditionally, for continuous monitoring, the patients are forced to stay at hospitals or to visit the hospital every few days; for example, in cerebral activity and heart rate monitoring cases [7]. However, the increasing availability of mobile devices, wireless networks, and embedded systems, and the tendency for them to become ubiquitous in our daily lives, creates a favourable technological environment for the emergence of novel, easy to use, and added-value applications for health care. Besides, the developing embedded-system technologies enable the design of real-time, multi-channel, and low-power mobile BS processing systems in various medical fields [8]. However, in mobile health care applications, power consumption and computational performance of these BS processing systems are the main concerns due to huge data size that is increasing day by day.

In this respect, the DTCWT has been implemented in various embedded systems with the aim of designing a real-time and multi-channel working sub-system that can be used in mobile health care systems. As a starting point, the DTCWT was first implemented in *C* programming language on a personal computer (PC) and also on a PIC microcontroller. The solution designed for the PC has shown sufficient speed performance for a single-input channel scenario. However, the speed of this PC implementation is dramatically affected by the reading/writing data operations. Additionally, in the multi-channel input scenario, for the real-time data, the speed of this implementation is dramatically reduced. Besides, this kind of implementation is not feasible for mobile applications. Therefore, to solve the non-portability drawback of the PC solution, the DTCWT was implemented on a microcontroller (PIC from Microchip). Comparing with PC-based solution; PIC implementation is very low cost and portable. However, considering the computational performance, PIC implementation can achieve a data

doi: 10.1049/htl.2016.0034

input rate that is far less for many multi-channel BSs. Hence, implementation of the DTCWT on a reconfigurable platform such as field programmable gate array (FPGA), which provides portable, low-cost, low-power, and high-performance computing, is considered as the most feasible solution. Initially, the system generator DSP design tool of Xilinx is employed for algorithm design. However, the solutions implemented by using such tools are not optimised in terms of area and power. To overcome all these drawbacks mentioned above, we considered to implement the DTCWT algorithm by using Verilog Hardware Description Language (VHDL), which has its own difficulties such as the necessity of choosing proper architecture satisfying the area and speed needs, long-time implementation procedure, difficulties in catching semantic errors, and the dependency on the program developer. To overcome these difficulties, simplify the usage of proposed algorithms, and simplify the adaptation procedures; a code generator program that can produce different architectures is proposed as the main contribution of this Letter.

In the literature, the classical DWT has already been implemented in various ways. For example in [9], a DWT is realised in FPGA by employing frame-partitioned architecture. In [10], a bit-serial architecture based on a time-interleaved structure is used in the implementation of the DWT and this results a modular and scalable architecture allowing a bit-level parameterisation. In [11], a DWT implementation, which exploits a look-up table (LUT)-based architecture of FPGAs and reformulating the wavelet computation in accordance with the distributed arithmetic algorithm, is proposed. In [12], this time, speed efficiency is considered by employing a low-pass and high-pass filter pair which is working concurrently in each level of transform. These implementations are only realised as computer-based simulations and no real-time hardware implementations are given. In our Letter, for the first time, various embedded-system solutions for implementing the DTCWT is realised. As the main contribution, to realise FPGA solution, which is concluded as the best, a code generator program is proposed.

This paper is organised as follows: Section 2.1 gives information about the DTCWT. Section 2.2 presents the first embedded-system solution as the desktop application using C language. Section 2.3 presents the implementation on PIC with C programming language as the second solution. Section 2.4 explains the implementation on system generator. Section 3 presents the VHDL solution and the proposed code generator. The last section presents the results and conclusions of this Letter.

2. Methods

2.1. Dual tree complex wavelet transform: The DTCWT [6], which was proposed in order to overcome the shift-variance drawback of ordinary DWT, utilises two discrete wavelet trees working in parallel on an input BS. In the DTCWT, real part of the transform is represented by the first wavelet tree while the imaginary part is represented by the second wavelet tree. Real and imaginary wavelet trees employ two different finite impulse response (FIR) filter sets that are working in parallel and satisfying the perfect reconstruction conditions. The block diagram which represents the analysis part of the DTCWT is given in Fig. 1. In analysis stages, low-pass and high-pass FIR filters that are labelled as '*g*' and '*h*', respectively, are employed in every level of algorithm.

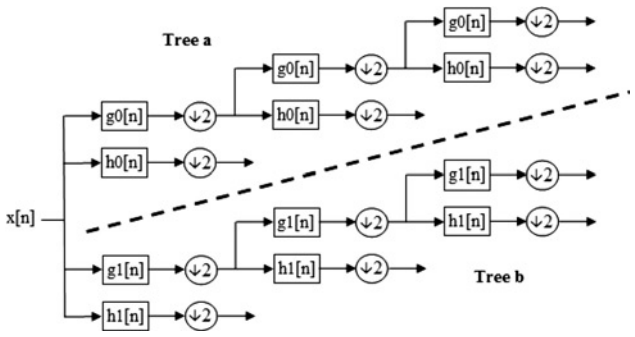

Fig. 1 *Block diagram for a three-level analysis part of DTCWT. Tree a represents the real part of the transform, whereas Tree b represents imaginary part (this figure is taken from [14])*

2.2. Desktop application using **C** language: In the first part of this Letter, as a starting point, the DTCWT is implemented on a desktop computer by using C programming language. Both fixed-point and floating-point formats are employed. In implementation, classical FIR filtering process is carried out by sliding the FIR filter coefficients on the input data as shown in Fig. 2. In the test phase of C implementation, 1600 pieces of 4096-point inputs are fed off-line and five levels of decomposition are applied resulting in the detail/approximation coefficients. The duration of decomposition process proves that with this kind of implementation 6.4 MHz input signals can be processed efficiently in real-time. However, in a real-life data acquisition system, the computational speed of the real-time DTCWT implementation is dramatically affected from reading/writing data operations. For example, when the reading/writing operations and the DTCWT decomposition is carried out together, the duration of the whole process increases 15 times due to the speed differences between the main processor and peripherals. Additionally, C implementation is performed in both floating-point and fixed-point formats and it is seen that fixed-point format is 35% more efficient than the floating-point format.

2.3. Implementation on PIC with **C** programming language: As a second step to achieve mobility property desired in biomedical applications, the DTCWT is implemented on PIC environment. Owing to the limited memory resources of PICs, the approach explained in Section 3 cannot be used and a new approach is needed. In the new approach, the new-coming ten samples (the length of the filters are also 10) of the input data is held in memory and the filtering operation is applied on this ten samples resulting in the detail/approximation coefficients. Later, a new ten sample input data comes and filtering process is applied to the second set, and this process goes on until the end of input data. Unlike the previous method that stores all input data, in this approach only ten samples incoming data is stored and processed. Algorithm that is used in PICs during the implementation of the DTCWT can be seen in Fig. 3. The algorithm is tested on PIC 18F452 model by using the fixed-point and floating-point arithmetic. It is observed that the fixed-point implementation is ~35% faster than floating-point arithmetic.

2.4. Implementation on system generator: To achieve mobility and fulfil the real-time processing speed requirements in the biomedical applications, we decide to implement the DTCWT in FPGAs as a system on chip. System generator DSP design tool of Xilinx, which enables Mathworks model-based FPGA design environment, is employed in the implementation due to its simplicity in FPGA design. In system generator-based design, all of the downstream FPGA implementation steps including

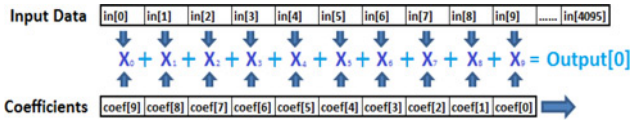

Fig. 2 *Implementation of the DTCWT in C language*

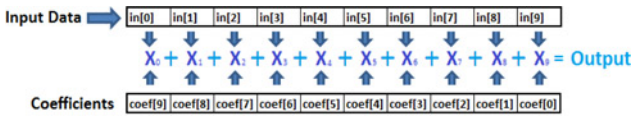

Fig. 3 *Implementation of the DTCWT on PIC*

synthesis and place and route steps are automatically performed by the Xilinx tool to generate the FPGA programming file [13].

In this section, only the real tree of the DTCWT is modelled and tested by using the fixed-point format for improving the speed efficiency. In our model, the input data is read from and also the outputs are written to a file in a PC, and the filter coefficients are stored in registers of FPGA design. At the end of simulation phase, it is observed that the outputs of system generator-based implementation (detail/approximation coefficients) are almost same with the outputs of MATLAB results. However, the design generated from system generator could not achieve the desired area efficiency so that the design could not entirely fit to the Xilinx Spartan-6 board. The results of the trials also show that only a low-pass or a high-pass filter module can fit to the Spartan-6 board. Therefore, in order to achieve desired field efficiency, the DTCWT is implemented using hard-coded Verilog language in subsequent design steps. The DTCWT architecture obtained from the system generator tool can be seen in Fig. 4.

3. Verilog implementation and proposed code generator: The previous attempts show that in order to achieve a real-time, low-power, and mobile (area efficient) implementation of the DTCWT, the best approach would be to implement the algorithms by using hardware description language such as Verilog, VHDL, system C etc. Considering the speed, power, and area constraints, in this Letter three different versions of the DTCWT were implemented on Spartan-6 FPGA board by using Verilog programming language. In this Letter, three different implementations of the DTCWT with desired specifications on a reconfigurable platform are given. In the first implementation, an adder and a multiplier for each tree are used. In the second implementation, an adder and a multiplier are used for each input channels resulting in reduced area. In the third implementation, to improve the area efficiency one step further, an adder and a multiplier are used for all channels. The details of these three implementations can be seen in our previous studies [14–16].

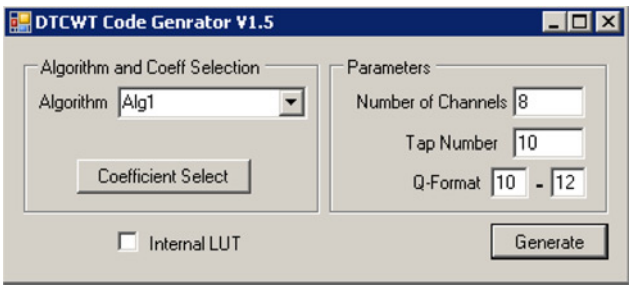

Fig. 5 *Code generator interface*

As seen in previous sections, the DTCWT algorithm is implemented using various architectures depending on the area and speed needs of the real-time system. While implementing these different approaches some difficulties are faced. These difficulties can be listed as:

- Choosing a proper architecture satisfying the area and speed needs.
- Long-time implementation procedure.
- Difficulties in catching semantic errors.
- Dependency on the program developer.

To overcome these difficulties, simplify the usage of proposed algorithms and simplify the adaptation procedures; a code generator program that can produce different architectures is developed. By using this code generator, various combinations of proposed algorithms can be generated in a very simple way. The interface of the code generator is given in Fig. 5.

In the code generator following features can be selected:

- Number of TAP.
- Bit width.
- Coefficient selection.

Architecture type:

- One adder and one multiplier for every tree.
- One adder and one multiplier for every channel.
- One adder and one multiplier for all channel.
- Internal LUTs or one LUT for coefficients.

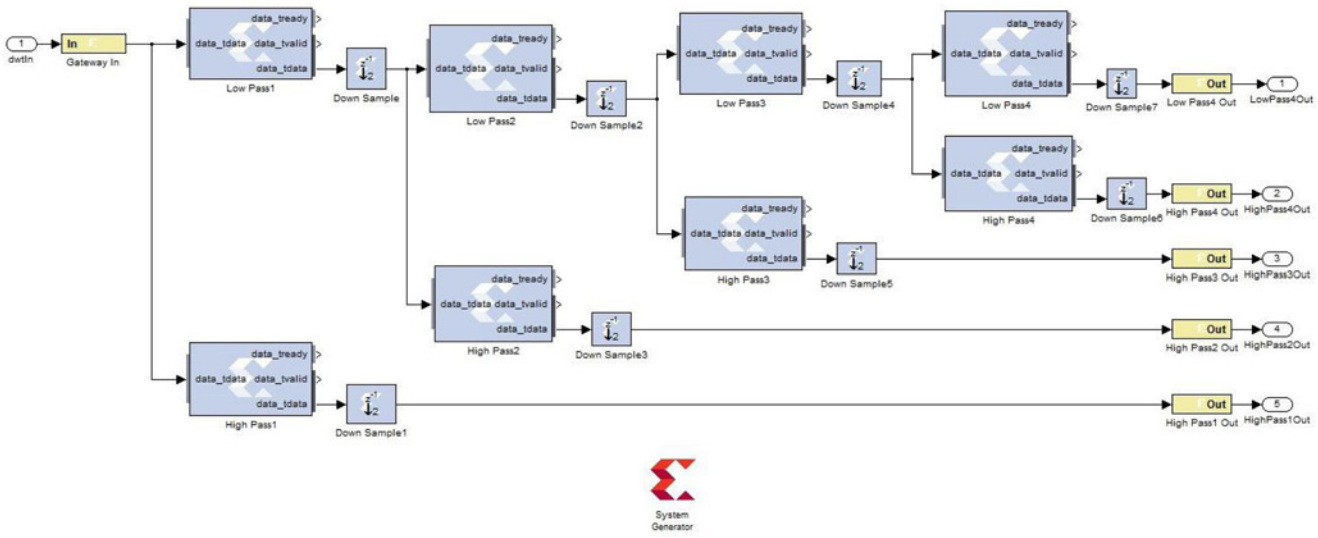

Fig. 4 *Implementation of the DTCWT on system generator*

doi: 10.1049/htl.2016.0034

Table 1 Register and LUT number results for all combinations

Bit width	Register	LUT	LUT-Flip-Flop (FF) pairs
12	2129	3445	1325
18	2319	3753	1421
32	2773	4062	1618

4. Results: In the proposed Letter, various embedded-system approaches are tested for implementing the DTCWT in a real-time and mobile system. Results indicate that a hard-coded FPGA solution would be the optimum choice in order to succeed both the low-area and low-power requirements of mobile systems. However, in BS acquisition systems different number of input channels can be employed and in the wavelet analysis different length filters can be used. Therefore, an automated tool that can be used in generating optimum hardware architecture for the analysis is desired. The proposed code generator can generate the optimum architecture for all our four approaches. The results of multi-channel approach [16] can be seen in Table 1. Code generator is used with word sizes of 12, 18, and 32 bits. The register and LUT number results are listed for all combinations. With the proposed code generator, an optimum solution with desired power consumption rate and area usage is achieved. With a single Spartan-6 FPGA board, a multi-channel system working with 55 parallel channels, which can never be achieved by the system generator tool provided by Xilinx, can be designed.

To prove that the FPGA implementation works as intended, a test signal, which is an embolic signal recorded from a patient [17], is used and the resultant coefficients of first approach (one adder and one multiplier for every tree) on an FPGA are compared with the coefficients obtained from traditional MATLAB implementation using floating-point arithmetic. To assess the similarity of two results, a metric known as signal-difference ratio (SDR) [18] is used. SDR is given as

$$\mathrm{SDR} = \frac{\sum \left| X_{\mathrm{floating}} - X_{\mathrm{fixed}} \right|}{\sum \left| X_{\mathrm{floating}} \right|} \times 100 \qquad (1)$$

where X_{floating} is the reference signal, X_{fixed} is the actual signal. Comparison results are reported in Table 2. In this table, the SDR between floating-point MATLAB simulation and fixed-point FPGA implementation with a word size of 32 bits are listed. According to this table, it is possible to say that there appears to be a very small difference between MATLAB simulation and actual implementations. This is actually due to quantisation error caused by fixed-point implementation in FPGA, as opposing to floating-point implementation in MATLAB.

The area and speed comparisons of four architectures for N-channel 32 bit structure can be seen in Table 3. All the experiments are done in FPGA Spartan-6 board and a PC having Intel Pentium i7 processor with 1.5 GHz clock speed. As expected, the fourth approach (one adder and one multiplier for N-channel with an LUT) is the most area efficient and also the slowest architecture.

Table 2 Signal difference ratio between MATLAB and FPGA results

Level	Floating-point–fixed-point
1	0.00008
2	0.00008
3	0.00009
4	0.00012
5	0.00014

Table 3 Adder/multiplier number and maximum frequency results for N-channel 32 bit emboli data

Architecture	Adder and multiplier number	Maximum frequency, kHz
one adder and one multiplier for each tree	$2N$	1840
one adder and one multiplier for each DTCWT	N	920
one adder and one multiplier for N-channel	1	$920/N$
one adder and one multiplier for N-channel with an LUT	1	$920/N$

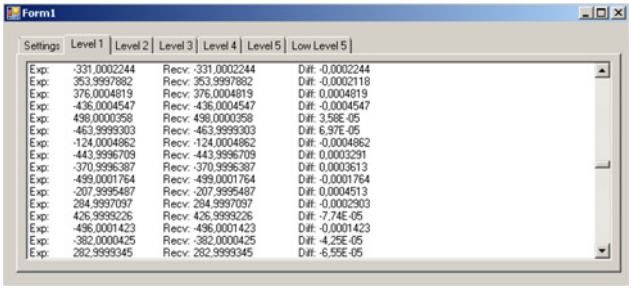

Fig. 6 *Graphical user interface (GUI) of the desktop application for the FPGA–PC interaction and result comparison*

In Fig. 6, the interface of the FPGA–PC interaction application is shown. The application reads data from a file and sends the data to FPGA via Ethernet. The application also receives the results from FPGA and shows difference between the MATLAB results. The application can be seen on [19] video.

With the proposed code generator, an optimum solution with desired power consumption rate and area usage is achieved. The code generated by our code generator uses only 7% area of Spartan-6 FPGA for 8-channel 32 bit architecture while Xilinx system generator uses all areas for only the FIR filter section of the same algorithm.

5. Conclusion: In this Letter, we presented an implementation of DTCWT code generator tool. We proved that our code generator is able to generate much more optimised algorithms in terms of the area utilisation and power consumption compared with other commercially available tools such as Xilinx system generator tool. As a future study, the code generator's capability can be extended to include optimised code generation for other signal processing algorithms.

6. Declaration of Interests: Conflict of interest: None declared.

7 References

[1] GothwalH., Kedawat S., Kumar R.: 'Cardiac arrhythmias detection in an ECG beat signal using fast Fourier transform and artificial neural network', *J. Biomed. Sci. Eng.*, 2011, **4**, (4), pp. 289–296

[2] Yu Y., Kwan B., Lim C., *ET AL.*: 'Video-based heart rate measurement using short-time Fourier transform'. 2013 Int. Symp. on Intelligent Signal Processing and Communications Systems (ISPACS), 12–15 November 2013, pp. 704–707

[3] Mallat S.: 'A theory for multiresolution signal decomposition: the wavelet representation', *IEEE Trans. Pattern Anal. Mach. Intell.*, 1989, **11**, pp. 674–693

[4] Aydin N., Marvasti F., Markus H.S.: 'Embolic Doppler ultrasound signal detection using discrete wavelet transform', *IEEE Trans. Inf. Technol. Biomed.*, 2004, **8**, (2), pp. 182–190

[5] Serbes G., Aydin N.: 'Denoising embolic Doppler ultrasound signals using dual tree complex discrete wavelet transform'. 2010 Annual Int.

Conf. of the IEEE Engineering in Medicine and Biology Society (EMBC), 2010, pp. 1840–1843
[6] Selesnick W., Baraniuk R.G., Kingsbury N.G.: 'The dual-tree complex wavelet transform', *IEEE Signal Process. Mag.*, 2005, **22**, (6), pp. 123–151
[7] Achten J., Jeukendrup A.E.: 'Heart rate monitoring applications and limitations', *Sports Med.*, 2003, **33**, (7), pp. 517–538
[8] Dong M.J., Yung K.G., Kaiser W.J.: 'Low power signal processing architectures for network microsensors'. 1997 Int. Symp. on Low Power Electronics and Design, 1997 Proc., 1997, pp. 173–177
[9] Mei-hua X., Zhang-jin C., Feng R., *ET AL.*: 'Architecture research and VLSI implementation for discrete wavelet packet transform.' Conf. on High Density Microsystem Design and Packaging and Component Failure Analysis, Shanghai, 2006, pp. 1–4
[10] Nibouche M., Bouridane A., Nibouche O., *ET AL.*: 'Design and FPGA implementation of orthonormal inverse discrete wavelet transforms.' 2001 IEEE Third Workshop on Signal Processing Advances in, Taiwan, 2001, pp. 356–359
[11] Al-Haj A.M.: 'Fast discrete wavelet transformation using FPGAs and distributed arithmetic', *Int. J. Appl. Sci. Eng.*, 2003, **1**, (2), pp. 160–171
[12] Farahanı M.A., Eshghı M.: 'Wavelet packet transform on FPGA'. Proc. of the Eighth WSEAS Int. Conf. on Acoustics & Music: Theory & Applications, Vancouver, Canada, 19–21 June 2007
[13] http://www.xilinx.com/support/documentation/sw_manuals/xilinx11/sysgen_gs.pdf
[14] Canbay F., Levent V.E., Serbes G., *ET AL.*: 'Field programmable gate arrays implementation of dual tree complex wavelet transform'. 37th Annual Int. Conf. of the IEEE Engineering in Medicine and Biology Society (EMBC'2015), Milan, Italy, 2015
[15] Canbay F., Levent V.E., Serbes G., *ET AL.*: 'An area efficient real time implementation of dual tree complex wavelet transform in field programmable gate arrays'. 15th IEEE Int. Conf. on Bioinformatics and Bioengineering (BIBE'2015), Belgrade, Serbia, 2015
[16] Canbay F., Levent V.E., Serbes G., *ET AL.*: 'A multi-channel real time implementation of dual tree complex wavelet transform in field programmable gate arrays'. 15th IEEE Int. Conf. on Bioinformatics and Bioengineering (MEDICON'2016), Paphos, CYPRUS, 2016
[17] Serbes G., Aydin N.: 'A complex discrete wavelet transform for processing quadrature Doppler ultrasound signals'. Ninth Int. Conf. on Information Technology and Applications in Biomedicine, 2009, pp. 1–4
[18] Serbes G., Aydin N.: 'Denoising performance of modified dual-tree complex wavelet transform for processing quadrature embolic Doppler signals', *Med. Biol. Eng. Comput.*, 2014, **52**, (1), pp. 29–43
[19] https://www.youtube.com/watch?v=NXJ7wLfOIcM

doi: 10.1049/htl.2016.0034