2011 Third International Conference on Computational Intelligence, Communication Systems and Networks

VHDL Modelling of Fixed-point DWT for the Purpose of EMG Signal Denoising

Md. Rezwanul Ahsan, Muhammad Ibn Ibrahimy, Othman Omran Khalifa
Electrical and Computer Engineering Department, Faculty of Engineering,
International Islamic University Malaysia (IIUM)
Kuala Lumpur, Malaysia
Email: g0817533@student.iium.edu.my

***Abstract*—Wavelet Transform (WT) has been widely applied in biomedical signal analysis. This paper will present the denoising method of EMG signal using WT and its model processed by VHSIC (Very High Speed Integrated Circuit) Hardware Description Language (VHDL) model of it. The principle of wavelet denoising is first to decompose the signal by performing a WT, followed by applying suitable thresholds to the detail coefficients, zeroing all coefficients below their associated thresholds, and finally to reconstruct the denoised signal based on the modified detail coefficients. Discrete Wavelet Transform (DWT) is a method that uses wavelet analyzer in which case the signal split into small pieces preserving both time and frequency properties. The Second order of Daubechies family (db2) has been used to denoise EMG signals. The simulation, synthesis and verification of the design presents a fast and reliable prototyping of DWT for denoising of EMG signals.**

Keywords-Electromyography, VHDL, DWT, Daubechies, Fixed-point

I. INTRODUCTION

EMG signal is the measure of electrical currents that generate in a muscle during its contraction and thus represents neuromuscular activities. EMG signals find its utilization in various fields that may include clinical diagnosis, managing and controlling motor disability through rehabilitation engineering, biomedical applications, human-machine interface systems, interactive virtual-reality games even in many recreational and exercise equipment. The main difficulties in analyzing and applying the EMG signal in those above mentioned fields are due to its noisy and sensitive characteristics. Compared to other biosignals, EMG contains complicated types of noise that are caused by, for example, inherent equipment noise, electromagnetic radiation, power-line noise, motion artifacts, age of the muscle and the interaction of different tissues inside muscle. For that reason, pre-processing is needed to filter out the unwanted noises involved in EMG signals. EMG signals can easily be influenced by external noise sources and artifacts. The first three types of noise can be eliminated by using typical filtering procedures such as band-pass filter, band-stop filter or the use of good quality of equipment with proper electrode placement [1]. Whereas it is difficult to remove the effect of other noises/artifacts and interferences of random noises that are in between dominant frequency range of EMG signal.

II. LITERATURE REVIEW

When detecting and recording the EMG signal, there are two major factors related to the fidelity of the signal. The first term is the signal to noise ratio which describes the strength of EMG signals in compare to strength of noise signal. The other factor involves with the distortion of the signal that is any frequency component which has relative contribution to the EMG signal should remain unchanged [1]. The method based on Fourier Transform (FT) can perfectly identify and isolate the signal frequencies but unable to localize when a particular component occurred in time. Hence, it is difficult and impractical to obtain time localization from FT phase if the signal is contaminated with noise [2]. This deficiency can be overcome by utilizing Short-Time Fourier Transform (STFT) which represents the signal in both time and frequency domain through time windowing function. However, still it does not adopt an optimal time or frequency resolution for the non-stationary signal like EMG. Moreover, the time frequency domain resolution tradeoff of a window is constrained by the Heisenberg uncertainty principle [3].

The Wavelet Transform (WT) is an efficient tool for multi-resolution analysis of non-stationary and fast transient signals that make it especially suitable for the analysis of neurophysiological signals [4]. A wavelet is a waveform of effectively limited duration with an average value of zero. This is the basic form of analysis tool which has energy concentrated in time and utilized for the exploration of transient, non-stationary time-varying signals. The name 'wavelet' with its contemporary meaning was first mentioned by Grossman and Morlet [5] during the early eighties of the past century in the context of quantum physics. Working on digital signal processing Stephane Mallat [6] provided a new contribution to wavelet theory by connecting the term filters with mirror symmetry, the pyramid algorithm and orthonormal wavelet basis. Yves Meyer [7] constructed a continuously differentiable wavelet lacking and finally, Ingrid Daubechies [8] managed to add to Haar's work by constructing various families of orthonormal wavelet bases. The WT has many similarities with STFT but is basically different in its basis function called wavelets which are not of fixed length. Its time-frequency resolution plane provides good time localization at high frequencies and improved frequency discrimination at low frequencies [9]. It has found numerous applications in data compression and

978-0-7695-4482-3/11 $26.00 © 2011 IEEE
DOI 10.1109/CICSyN.2011.58

feature enhancement for images, speech, and biosignals [10], [11], [12]

The rapid changing technology introducing much more development to Electronic Design Automation (EDA) tools which in turn reduce the design cycle, complexity and valuable time to the programmer. However, the VHDL based algorithm development and design process not yet free from complexity and time killing. Furthermore, it requires some sort of expertness and experience from hardware language field. There are very few numbers of related research works available because of quiet new and still developing field of research. Mallat's Fast Wavelet Transform (FWT) algorithm based implementation has been performed by Motra et. al [13]. For forward and inverse transform operation, the parallel distributed arithmetic Finite Impulse Response (FIR) filter is used as basic building block whereas the data stores in look-up-table (LUT) during operation. Haar and Daubechies Wavelet Transform has been implemented and a very good comparison is presented by Elfouly et. al. [14] on FPGA technology. Their design also utilized FIR filter as building block and LUT for storing data. Some other VHDL implementation of DWT can be seen in [15], [16], [17] for image compression, image denoising etc.

III. PROPOSED TECHNIQUE

This paper presents the 16-bit fixed-point based VHDL modelling of DWT for the purpose of EMG signal denoising. The development process carried on by developing the program in C++ platform first. Afterwards, VHDL model is designed by utilizing modular programming for each type of function like collecting data, forward transform, inverse transform, sorting and median calculation. The development process, result and performance of the designed VHDL model are verified by analyzing the outputs as mentioned in synthesis and simulation section.

IV. METHODOLOGIES

A WT decomposes a signal into shifted and scaled versions of the original (or mother) wavelet. The DWT is performed by successive low-pass and high-pass filtering of the discrete time-domain. The resolution of a signal, which is a measure of the detail information in the signal, is determined by the filtering operations. On the other hand the scale is determined by up-sampling and down-sampling (sub-sampling) operations. The ability of DWT to extract features from the signal is dependent on the appropriate choice of the mother wavelet function. Some of the popular standard families of the wavelet basic functions are Haar, Daubechies (db), Coeiflet (coef), Symmlet (sum), Morlet and Maxican Hat. Even though there is no well-defined rule for selecting a wavelet basis function in a particular application or analysis, some properties of the wavelets make a specific mother wavelet more suitable for a given application and signal type. On the basis of basic theoretical aspects for DWT [18], the signal details at an arbitrary scale is added to the approximation at that scale, then the signal approximation at increased resolution is obtained.

This research work involves with the implanting of Daubechies wavelet (db4) which is developed by Ingrid Daubechies. Daubechies wavelets are families of wavelets whose inverse wavelet transforms are are orthogonal. The wavelet transform using Daubechies wavelets result in progressively finer discrete samplings using recurrence relations. Every resolution scale is double that of the previous scale. For Daubechies wavelet transforms, the scaling signals and wavelets have slightly longer supportsthan Haar [19], i.e. they produce averages and differences using just a few more values from the signal.

However, this slight change provides a tremendous improvement in the capabilities of these new transforms. Daubechies wavelets use overlapping windows, so high frequency coefficients spectrum reflects at high frequency changes. The db4 has four coefficients, which are normalized according to the application field and requirements. For forward and inverse transform, the db4 coefficients are mentioned in Table 1.

TABLE I. DAUBECHIES 4-TAP (DB4) WAVELET COEFFICIENTS

Forward Transform		**Inverse Transform**	
Scaling Coefficients	***Wavelet Coefficients***	***Scaling Coefficients***	***Wavelet Coefficients***
h0 = (1+√3)/4√2	***g0 = h3***	***Ih0 = h2***	***Ig0 = h3***
h1 = (3+√3)/4√2	g1 = -h2	Ih1 = g2 = h1	Ig1 = g3 = -h0
h2 = (3-√3)/4√2	g2 = h1	Ih2 = h0	Ig2 = h1
h3 = (1-√3)/4√2	g3 = -h0	Ih3 = g0 = h3	Ig3 = g1 = -h2

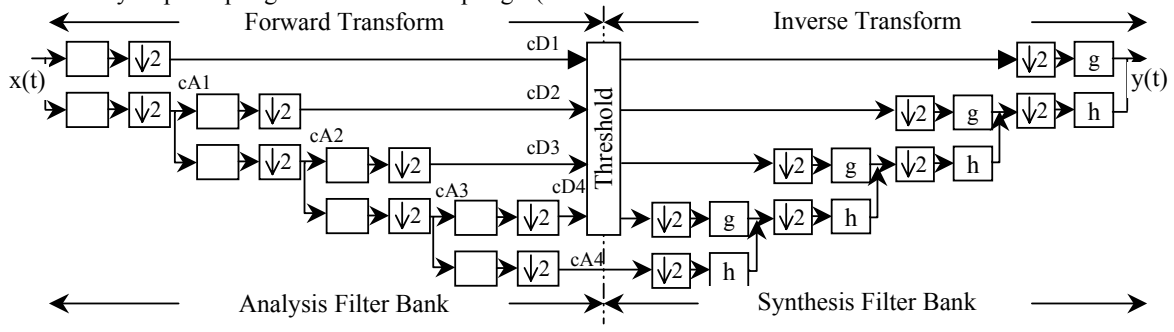

Figure 1. A four-level decomposition and reconstruction system.

The denoising of EMG signal is based on the principal work of Johnstone and Donoho [20]. To suppress the noise signature, they presented the thresholding of the DWT coefficients of an image and then reconstructed it. The method relies on the fact that noise commonly found as fine-grained structure in the image, and the wavelet transform provides a scale-based decomposition. Hence, most of the noise tends to be represented by wavelet coefficients at the finer scales. Discarding these coefficients would result in a natural filtering out of the noise on the basis of scale. The method involves thresholding of the wavelet coefficients to zero if their values are below a threshold. These coefficients are mostly those corresponding to noise. Therefore, the steps are computing DWT coefficients of a signal, find threshold, passing the detail coefficients through the threshold, then finally taking the inverse DWT. The threshold can be estimated by δ_{mad}(sqrt(2*ln(N)), where δ_{mad} is the median absolute deviation (median/0.6745) of the largest detail coefficient spectrum, N is the total number of sample data points. The whole process is shown in Fig. 1, where cA1, cA2, cA3, cA4 are approximate of 4 levels respectively and cD1, cD2, cD3, cD4 are the details of every level. The index denotes the level number. h and g are high frequency and low frequency filter coefficients respectively. The input x(t) is raw EMG signal and output y(t) is the reconstructed, denoised signal.

V. VHDL MODELLING

The overall program design first modelled in C++ because the functionality and reliability can be easily verified there with faster response. Afterwards, the program has been modelled using VHDL. For this purpose Altera Quartus II version 9.2 SP2, web edition used as the platform. The VHDL model comprise of structure, behavioural and physical version of the program. Furthermore, Quartus II provides the facility to implement the synthesized and designed model into available FPGA. The final VHDL model developed by utilizing modular programming. The top entity *DWT_Module* is the main module to manage and control other components of it. The components as a module are *Read_data, forward_DWT, bidSort, inverse_DWT* and *displayDT*. The name of the components clearly tells of its function. The schematic diagram of the VHDL model is depicted in Fig. 2. It is pretty good that the designed model is generic (any number of input can be feed and multi-level decomposition) and computational number system based on 16-bit fixed-point. For this purpose, IEEE proposed fixed-package [21] has been added to the libray and used in the program. Number of input data must be integer multiple of 32 (2^5) to perfom 4-level of transformation. There is another user defined package added namely *utilitypack* which defines the different data structure based on fixed-point, stores Daubechies forward and inverse coefficients, saves some constants and maintains a function to multiply two fixed-point number. The input data need to convert in fixed-point (16 bit binary, format 6 downto -9). The conversion of fixed-point to fractional decimal are as below:

0010110101110011 (16 bit, positive)
$=2^4+2^2+2^1+2^{-1}++2^{-3}+2^{-4}+2^{-5}+2^{-8}+2^{-9}$
=22.724609

1010110101110011 (16 bit, negative)
=-2's complement of "1010110101110011"
=-(0101001010001101)
$=-(2^5+2^3+2^0+2^{-2}+2^{-6}+2^{-7}+2^{-9})$
=-41.275391

The first module *Read_data* collect the input 16-bit fixed point data and stored in array. When it completed storing all the input data, the output flag *en_read* set to 'high' or 1 which will act as input flag for starting the operation of *forward_DWT* module. The number of data points and decomposition levels is maintained in *utilitypack* package. When first level completed, the start_sort flag set to 'high' and absolute value of details coefficients sent as a input for bidirectional selection sort namely bidSort module to start its operation. The function of *bidSort* is to sort the first details coefficients for finding median and finally it calculates the threshold using this median and number of data points. After this, hard thresholding applied to all details coefficients from every levels. The thresholded details coefficients is then fed to *inverse_DWT* module with the flag *en_th* set 'high'. In this module, the approximation and details coefficients reconstructed to get back denoised signals. A separate module named *displayDT* used only to see the output array of *Read_data*, *forward_DWT* and *inverse_DWT* serially. This is done only to save number of pins in I/O port since it is fixed for different device family which are most of the cases very less for the designed model.

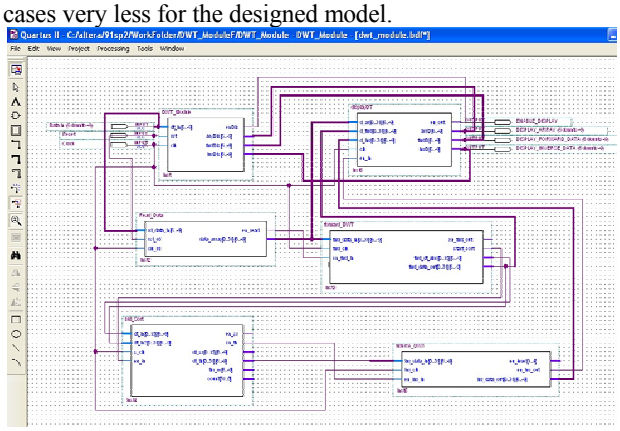

Figure 2. Schematic diagram of of Discrete Wavelet Transform.

VI. SYNTHESIS AND SIMULATION OF THE DESIGNED MODEL

The VHDL model for DWT has been analysed, synthesized and simulated for the device StratixIII, chipset EP3SE50F780I4L. The Quartus 9.1 CAD software has in-built tools for performing these operations which will also generate gate level architecture namely Register Transfer Level (RTL) diagram for the designed model and all modules. Also Technology Map of the designed DWT model

generated. In integrated circuit design, RTL description gives the full observation of the operation of a synchronous digital circuit. RTL design shows flow of signals between hardware registers, and the logical operations performed on those signals. The RTL or Technology Map helps to check the design visually before performing simulation and other verification processes. Fig. 3 shows the Technology Map view of the designed VHDL model.

The VHDL program model for DWT first tested for true construction property by performing forward and inverse transform. In that case, eight random data points of all possible combination used for two-level of decomposition.

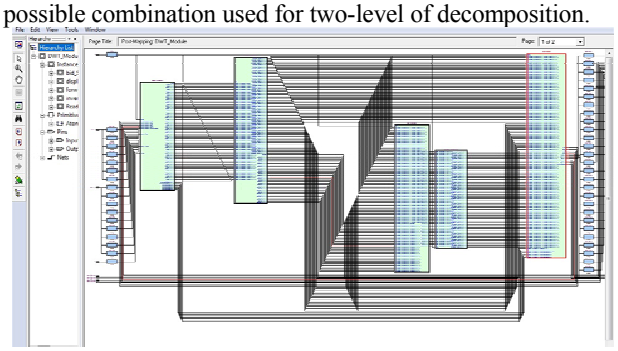

Figure 3. Technology map of the designed model.

The output of VHDL model and its comparison with C++ program output has outlined in the Table 2 and Table 3. It shows that there is very slight difference which is because of number of input bit used. Use of higher number bit will give better precision but that will cost complex computation, bulky memory reservation as well as inefficient time consumption.

After successful performance checking of the designed DWT model, the threshold module inserted for performing denoising on EMG signals. The designed model has been tested for denoising 16, 32 and 64 data points with 2, 3 and 4 level of decomposition and reconstruction. Here only the result from 32 data points is presented. The collected EMG data first multiplied by 100 before converting it to 16-bit fixed point as input to the top entity module as it is very small in value. The compilation summary of the designed model for 32 data points is shown in Fig. 4. The logic utilization is 42%, total registers used 4206, total block memory bits 512, DSP block 18-bit elements used 4 and total pins as I/O port 67. The output from *Read_Data, forward_DWT* and *inverse_DWT* are shown in Fig. 5 through vector waveform. The output data from designed VHDL model and C++ program has also been presented as a comparison. For forward transform, only approximation and details coefficients for different level present and in inverse transform the data points presents denoised and constructed signal. It clearly shows that the VHDL model performs very well with very little lack of decimal precision points.

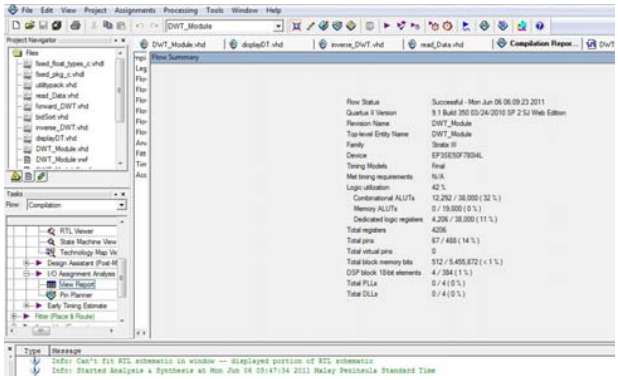

Figure 4. Compilation Summary.

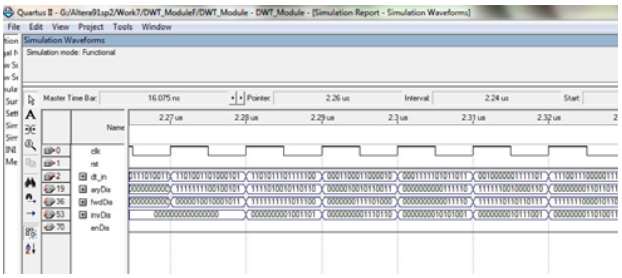

Figure 5. Vector waveform output from Designed VHDL model.

VII. CONCLUSION

The functionality of VHDL model for denoising EMG signal has been tested successfully, although 32 data points are used. The average difference between the reconstructed data is 0.002230. The comparison results also show that its performance level is in the satisfactory region (upto two decimal points). If more precision in the output result is necessary, then 32-bit of fixed-point can be the solution. However, if the processor with higher speed and larger memory not used, then computational cost in complexity and time will increase considerably. The future work may include floating point data and implementation of the model in a physical FPGA device.

REFERENCES

[1] M. B. I. Reaz, M. S. Hussain, and F. Mohd-Yasin, "Techniques of EMG signal analysis: detection, processing, classification and applications," Biological procedures online, vol. 8, no. 1, pp. 11–35, 2006.

[2] M. P. Wachowiak, G. S. Rash, P. M. Quesada, and A. H. Desoky, "Wavelet-based noise removal for biomechanical signals: a comparative study," Biomedical Engineering, IEEE Transactions on, vol. 47, no. 3, pp. 360-368, 2000.

[3] D. K. Kumar, N. D. Pah, and A. Bradley, "Wavelet analysis of surface electromyography," Neural Systems and Rehabilitation Engineering, IEEE Transactions on, vol. 11, no. 4, pp. 400–406, 2003.

[4] C. F. Jiang and S. L. Kuo, "A comparative study of wavelet denoising of surface Electromyographic signals," in Engineering in Medicine and Biology Society, 2007. EMBS 2007. 29th Annual International Conference of the IEEE, pp. 1868–1871, 2007.

[5] A. Grossmann and J. Morlet, "Decomposition of Hardy Functions into Square Integrable Wavelets of Constant Shape," SIAM Journal on Mathematical Analysis, vol. 15, no. 4, p. 723, 1984.

[6] S. Mallat, "A theory for multiresolution signal decomposition: the wavelet transform," IEEE Trans. Pattern Anal. Machine Intell, vol. 11, no. 7, pp. 674–693, 1989.

[7] Y. Meyer, Ondelettes. Hermann, 1990.

[8] I. Daubechies, "Orthonormal bases of compactly supported wavelets," Communications on pure and applied mathematics, vol. 41, no. 7, pp. 909–996, 1988.

[9] J. A. Crowe, "The wavelet transform and its application to biomedical signals," in Time-Frequency Analysis of Biomedical Signals (Digest No. 1997/006), IEEE Colloquium on, pp. 2–1, 1997.

[10] I. Daubechies, "Ten lectures on wavelets, volume 61 of CBMS-NSF Regional Conference Series in Applied Mathematics," Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, vol. 118, 1992.

[11] P. J. Sparto, M. Parnianpour, E. A. Barria, and J. M. Jagadeesh, "Wavelet analysis of electromyography for back muscle fatigue detection during isokinetic constant-torque exertions," Spine, vol. 24, no. 17, p. 1791, 1999.

[12] P. Wellig and G. S. Moschytz, "Analysis of wavelet features for myoelectric signal classification," in Electronics, Circuits and Systems, 1998 IEEE International Conference on, vol. 3, pp. 109–112, 1998,.

[13] A. S. Motra, P. K. Bora, and I. Chakrabarti, "An efficient hardware implementation of DWT and IDWT," in TENCON 2003. Conference on Convergent Technologies for Asia-Pacific Region, vol. 1, pp. 95–99, 2003,.

[14] F. H. Elfouly, M. I. Mahmoud, M. I. M. Dessouky, and S. Deyab, "Comparison between Haar and Daubechies, Wavelet Transformations on FPGA Technology," International Journal of Computer, Information, and Systems Science, and Engineering, vol. 2, no. 1, 2008.

[15] D. Dia et al., "Multi-level Discrete Wavelet Transform Architecture Design," in World Congress on Engineering 2009, London, UK, 1-3 July, vol. 1, 2009.

[16] Z. Razak and M. Yaacob, "VHDL Development of Discrete Wavelet Transformation," Malaysian Journal of Computer Science, vol. 15, no. 1, pp. 84–92, 2002.

[17] H. Liao, M. K. Mandal, and B. F. Cockburn, "Efficient implementation of lifting-based discrete wavelet transform," Electronics letters, vol. 38, no. 18, pp. 1010–1012, 2002.

[18] P. S. Addison, "The Illustrated Wavelet Transform Handbook: Introductory Theory and Applications in Science, Engineering, Medicine and Finance.", pp. 65-119, PA, Institute of Physics Publishing, 2002 .

[19] A. Haar, "Zur Theorie der orthogonalen Funktionensysteme.(Erste Mitteilung.)," Math. Ann, vol. 69, pp. 331–371, 1910.

[20] D. L. Donoho and I. M. Johnstone, "Ideal spatial adaptation via wavelet shrinkage," Biometrika, vol. 81, no. 3, pp. 425–455, 1994.

[21] D. Bishop, "Fixed point package user's guide," Packages and bodies for the IEEE, pp. 1076–2008.

TABLE II. INPUT AND OUTPUT OF WAVELET TRANSFORM IN FIXED-POINT FORMAT

Input		**Forward Transform**		**Inverse Transform**	
Decimal	***Fixed-point***	***Level-1***	***Level-2***	***Level-1***	***Level-2***
1.0	0000001000000000	1111011011101111	0000110111011101	1111011011110011	0000000111111101
-2.94	1111101000011111	0001001111110001	1111001000110010	0001001111100111	1111101000100110
-4.3275	1111011101011000	0000000010000000	0000001010001110	0000000001111101	1111011101011011
12.3	0001100010011010	1111010010110101	1111000101000110	1111010011000000	0001100010000111
7.8102	0000111110011111	1110110111110110			0000111110001110
-0.1721	1111111110101000	0000100011010000			1111111110101010
-14.5	1110001100000000	1110010011110111			1110001100010010
0.8924	0000000111001001	0000011111011000			0000000111010000

TABLE III. COMPARISON IN TERMS TRANSFORMATION OUTPUT BETWEEN VHDL AND C++

Forward Transform				**Inverse Transform**			
Level-1		***Level-2***		***Level-1***		***Level-2***	
VHDL	***C++***	***VHDL***	***C++***	***VHDL***	***C++***	***VHDL***	***C++***
-4.533203	-4.538115	6.931641	6.940247	-4.525391	-4.538115	0.994141	1.000000
9.970703	9.972008	-6.902344	-6.908747	9.951172	9.972008	-2.925781	-2.940000
0.250000	0.262501	1.277344	1.301330	0.244141	0.262501	-4.322266	-4.327500
-5.646484	-5.651847	-7.363281	-7.379460	-5.625000	-5.651847	12.263672	12.300000
-9.019531	-9.030895					7.777344	7.810200
4.406250	4.419528					-0.167969	-0.172100
-13.517578	-13.532622					-14.464844	-14.500000
3.921875	3.932839					0.906250	0.892400

TABLE IV. COMPARISON BETWEEN OUTPUT FROM VHDL MODEL AND C++ FOR 32 DATA POINTS

Input			**Forward Transform**			**Inverse Transform at Level-4**		
x.100	***16-bit Fixed-point***		***VHDL***		***C++***	***VHDL***		***C++***
			16-bit fixed point	***Decimal***		***16-bit fixed point***	***Decimal***	
-0.42725	1111111100100101	***cA4***	0000010010001011	2.272534	2.271484	0000000000000000	0.000000	-0.001438
-5.64575	1111010010110110		1111111111011100	-0.075264	-0.070313	0000000001001101	0.150391	0.148897
2.34985	0000010010110011	***D***	0000000111101000	0.951895	0.953125	0000000001110110	0.230469	0.230779

Input		Forward Transform				Inverse Transform at Level-4		
x.100	*16-bit Fixed-point*		*VHDL*		*C++*	*VHDL*		*C++*
			16-bit fixed point	*Decimal*		*16-bit fixed point*	*Decimal*	
0.12207	0000000000111110		0000000000011110	0.06079	0.058594	0000000010101001	0.330078	0.331002
-1.7395	1111110010000110		1111110110110111	-1.146372	-1.142578	0000000010111001	0.361328	0.362772
0.42725	0000000011011011	*cD3*	1111111000010110	-0.960307	-0.957031	0000000011010011	0.412109	0.412883
5.49316	0000101011111100		0000001110110111	1.85417	1.857422	0000000011110101	0.478516	0.481337
2.04468	0000010000010111		1111101001110100	-2.782896	-2.773438	0000000100010101	0.541016	0.544877
-2.99072	1111101000000101		1111110111000011	-1.125645	-1.119141	0000000100010011	0.537109	0.539962
-2.2583	1111101101111100		1111011010000101	-4.745964	-4.740234	0000000100011011	0.552734	0.553389
0.33569	0000000010101100		0000000100100001	0.570149	0.564453	0000000100101010	0.582031	0.585159
0.91553	0000000111010101	*cD2*	0000100011101100	4.472893	4.460938	0000000100111000	0.609375	0.612014
4.42505	0000100011011010		0000010101110100	2.731906	2.726563	0000000101001111	0.654297	0.657211
-0.61035	1111111011001000		0000010000000000	2.006766	2.000000	0000000101100101	0.697266	0.697493
-2.0752	1111101111011010		0000101101001101	5.662719	5.650391	0000000101110111	0.732422	0.732860
1.83105	0000001110101001		1111011111100100	-4.068445	-4.054688	0000000110001001	0.767578	0.769545
1.40381	0000001011001111		0000011001110101	3.227483	3.228516	0000000100011011	0.552734	0.550755
3.44849	0000011011100110		1111110000000101	-1.99292	-1.990234	0000000011001110	0.402344	0.400420
-1.46484	1111110100010010		0000011101111000	3.736956	3.734375	0000000010100100	0.320313	0.318539
-4.48608	1111011100000111		1111101011011001	-2.580281	-2.576172	0000000001110000	0.218750	0.218315
6.50024	0000110100000000		0000000101110111	0.731855	0.732422	0000000001100001	0.189453	0.186546
0.64087	0000000101001000		0000011101111110	3.747751	3.746094	0000000001000111	0.138672	0.136434
-5.27954	1111010101110001		1111100111100110	-3.056105	-3.050781	0000000000100101	0.072266	0.067980
0.39673	0000000011001011		1111111010111100	-0.633051	-0.632813	0000000000000011	0.005859	0.004441
4.94385	0000100111100011	*cD1*	1111111111110111	-0.013377	-0.017578	0000000000000110	0.011719	0.009356
-0.7019	1111111010011001		0000110010100101	6.323132	6.322266	0000000000000000	0.000000	-0.004072
-0.36621	1111111101000101		1111010011010011	-5.592867	-5.587891	1111111111101111	-0.033203	-0.035841
-2.2583	1111101101111100		0000101000011111	5.068901	5.060547	1111111111100001	-0.060547	-0.062696
6.37817	0000110011000010		0000000010011101	0.30188	0.306641	1111111111001011	-0.103516	-0.107893
1.67847	0000001101011011		0000101000101000	5.07838	5.078125	1111111110110110	-0.144531	-0.148175
0.24414	0000000001111101		0000001001010110	1.169223	1.167969	1111111110100100	-0.179688	-0.183543
-4.48608	1111011100000111		0000011010101111	3.34322	3.341797	1111111110010001	-0.216797	-0.220227