Memory Reduction Methodology for Distributed-Arithmetic-Based DWT/IDWT Exploiting Data Symmetry

Amit Acharyya, Koushik Maharatna, *Member, IEEE*,
Bashir M. Al-Hashimi, *Fellow, IEEE*, and Steve R. Gunn, *Member, IEEE*

***Abstract*—In this brief, we show that by exploiting the inherent symmetry of the discrete wavelet transform (DWT) algorithm and consequently storing only the nonrepetitive combinations of filter coefficients, the size of required memory can be significantly reduced. Subsequently, a memory-efficient architecture for DWT/inverse DWT is proposed. It occupies 6.5-mm^2 silicon area and consumes 46.8-μW power at 1 MHz for 1.2 V using 0.13-μm standard cell technology.**

***Index Terms*—Distributed arithmetic (DA), low-power architecture, multiplierless implementation, very large scale integration, wavelet.**

I. Introduction

THE discrete wavelet transform (DWT) plays a central role in a number of signal and image processing applications [1]. Owing to its importance in real-time signal processing systems, its first hardware implementation has been carried out in [2]. Subsequently, significant research effort has been made to optimize DWT/inverse DWT (IDWT) implementation, like architectures based on the folded digit-serial approach [3] and low-complexity architectures with a reduced number of multipliers [4]–[15]. However, these hardware architectures do not adequately address the power and area consumption issues [15], which often are the two most important metrics in today's high-performance signal processing systems. The main power-consuming operation in DWT/IDWT computation is filtering, which requires a significant number of multiplications [12]. Distributed arithmetic (DA) can be adopted to eliminate the requirement of multiplication [12], which may lead to the reduction of power consumption. However, in the conventional DA-based approach, one needs to store all the possible combinations of filter coefficients in the memory, which increases exponentially in size with the frame length [12]. Thus, for a longer frame length, the advantage of using DA may eventually be lost, owing to the significant increase in memory size.

In this brief, we propose a novel methodology for memory reduction in the DA-based design of the DWT/IDWT architecture by exploiting its inherent algorithmic symmetry resulting in data repeatability. Subsequently, a 16-b fixed-point DWT/IDWT architecture is developed for a frame length of 16, which requires significantly less silicon area and power consumption compared to some of the published DWT/IDWT architectures. The rest of this brief is organized as follows. Section II outlines the adopted methodology, and Section III describes the DWT/IDWT architecture designed using the proposed methodology. Section IV analyzes the performance of the architecture, and Section V gives the conclusions.

Manuscript received October 9, 2008; revised December 2, 2008. First published March 21, 2009; current version published April 17, 2009. This work was supported in part by the EPSRC/UK under Grant EP/E035965/1. This paper was recommended by Associate Editor T. Zhang.

The authors are with the Pervasive Systems Centre, School of Electronics and Computer Science, University of Southampton, SO17 1BJ, Southampton, U.K. (e-mail: aa07r@ecs.soton.ac.uk; km3@ecs.soton.ac.uk; bmah@ecs.soton.ac.uk; srg@ecs.soton.ac.uk).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSII.2009.2015386

TABLE I
CONVENTIONAL AND REDUCED MEMORY REQUIREMENT TO STORE COMBINATIONS OF FILTER COEFFICIENTS IN DA FOR FRAME-LENGTH = 4 AND WORDLENGTH = 4

Adr	Data	Adr	Data	Adr	Data
0000	0	0001	h_2	0010	h_1
0011	h_2+h_1	0100	h_0	0101	h_0+h_2
0110	h_0+h_1	0111	$h_0+h_1+h_2$	1000	0
1001	h_2	1010	h_1	1011	h_2+h_1
1100	h_0	1101	h_0+h_2	1110	h_0+h_1
1111	$h_0+h_1+h_2$	—	—	—	—
00	h_0	01	h_2	10	h_1
11	h_1+h_2	—	—	—	—

II. Methodology

A. Motivational Example

The basic DA equation can be given as [17]

$$x_n = -x_{n,l}.2^l + \sum_{b=0}^{l-1} x_{n,b}.2^b \tag{1}$$

where $l =$ (total number of bits per sample). In dyadic space, a convolution-based wavelet filter can be represented as

$$w_a = \sum_n x_n h_{2a-n} \tag{2}$$

where x_n and h_n are input samples and filter coefficients, respectively. Considering frame-length $= 4$ and wordlength $= 4$ (as an example) and using (1) in (2) with $a = 1$, we get

$$\begin{aligned} w_1 = -[x_{2,3}h_0 + x_{1,3}h_1 + x_{0,3}h_2]2^3 + \cdots \\ +[x_{2,0}h_0 + x_{1,0}h_1 + x_{0,0}h_2]2^0 \end{aligned} \tag{3}$$

where x_{ij} is the ith sample's jth bit of the input data. The possible combinations of filter coefficients obtained from (3) are shown in the first six rows of Table I, which occupies 16 memory locations. However, it can be observed in Table I that there exists redundant (such as "0") and repetitive filter coefficients (such as "h_0," "h_1," "h_2," "$h_1 + h_2$," "$h_0 + h_1$," "$h_0 + h_2$," and "$h_0 + h_1 + h_2$") occupying more than a single memory location. Thus, if only the unique combinations of the filter coefficients are stored in the memory, the other filter coefficients can be obtained on the fly using simple addition operations. In this particular example, the proposed methodology leads to only four memory locations, as shown in the last two rows of Table I, rather than 16 locations in the conventional approach.

However, reducing memory at the expense of adders raises two particular issues. First, a new addressing scheme needs to be formulated to address the reduced memory system. Second, the hardware savings obtained due to the reduction in memory size can be negated if the total number of adders used in the design is more than a certain limit. These issues are discussed in Sections III and IV, respectively.

B. Proposed Methodology

Considering wavelet computation in dyadic space for frame length $(p) =$ number of filter coefficients and assuming that the ith level of resolution consists of j number of filter coefficients, the following relationships hold: $\forall i \in [1, \log_2 p]$, $j \in [1, p/2^i]$, where i and j are integers. Now, if the jth coefficient consists of s number of data samples, considering the causality of the system, s can be represented as

$$s = 1 + (j-1)2^i. \tag{4}$$

The maximum number of samples $(s_{\max})$ present at the ith level of resolution can be obtained by substituting $j = p/2^i$ in (4), which leads to

$$s_{\max} = 1 + p - 2^i. \tag{5}$$

It is to be noted from (5) that $(s_{\max} - 1)$ is an even number. In this methodology, we divide $(s_{\max} - 1)$ number of samples into k subframes. Each of the subframes consists of a pair of data samples x_{n-1} and x_n. As shown later in Section IV, this approach will lead to reduction in memory. k can be represented as

$$k = (p/2) - 2^{(i-1)}. \tag{6}$$

At each time instant, we check the binary value of the mth bit of x_{n-1} and x_n, and depending on their combination, we fetch the appropriate linear combination of the filter coefficients from the memory [as can be obtained by expanding (2)]. However, one of these combinations is "00," which means that no filter coefficient needs to be multiplied with the input data. As an example, in (3), if '$x_{1,3}x_{0,3}$' = '00,' then multiplying these input bits with h_1 and h_2 will always yield zero. Thus, one needs to store the combinations of filter coefficients corresponding to only three bit combinations of the sample pair (namely, "01," "10," and "11"). Arranging $(s_{\max} - 1)$ samples from $s_{\max}$ samples at the ith resolution level, we are left with only one sample. Since the mth bit of this sample can assume either "0" or "1," only one combination of filter coefficients needs to be stored in the memory corresponding to this sample. Thus, the total memory requirement (M_i) in this methodology for the ith level of resolution can be given by

$$M_i = k \times 3 + 1 = \left[(p/2) - 2^{(i-1)}\right] \times 3 + 1. \tag{7}$$

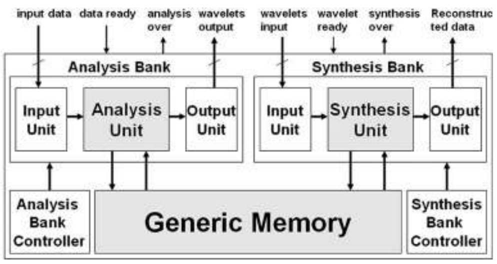

Fig. 1. Very large scale integration architecture of DA-based DWT and IDWT.

The memory requirement for the analysis bank can be computed by summing M_i for all i. However, at the last resolution level of the analysis bank, all that is left is one residue signal along with the wavelet coefficient. This residue signal consists of one sample for which, as discussed above, two combinations of filter coefficients are possible in which one is "0." Therefore, the combination of filter coefficients corresponding to this residue signal occupies one memory block only. Thus, according to this proposed methodology, the total memory requirement (TMR) can be represented as

$$TMR = 1 + \sum_{i=1}^{\log_2 p} M_i. \tag{8}$$

From (8), it is evident that TMR grows nearly linearly with the increase of frame length.

III. Architectural Overview

The block diagram of the DWT/IDWT architecture is shown in Fig. 1. At the block level, the architecture is similar to a standard DA-based architecture. However, the main novelty of the architecture lies in the formulation of a new addressing scheme and the corresponding address generation unit design for the reduced memory unit (shown as Generic Memory in Fig. 1) discussed in Section II.

A. Memory Unit

As outlined in Section II, in the memory unit of this architecture, only the nonrepetitive combinations of filter coefficients are stored from the set of data. The strategy for address generation is explained here with an example of frame length 16 for the first resolution level of the analysis bank. Fig. 2 shows the incoming data samples and the associated filter coefficients required for computing each wavelet coefficient in this case. It can be noted from Fig. 2 that the filter coefficients are the same for the different samples present in the same inclined slices. For example, as shown in the encircled regions in Fig. 2, the filter coefficients associated with the samples for computing the third wavelet coefficient are the same for the last five samples of the fourth wavelet coefficient computation. Mathematically, the filter coefficients associated with the qth data sample of the rth and $(q+2)$th data samples of the $(r+1)$th wavelet coefficients are the same (symmetry property). This symmetry can be represented in generalized form for the ith level of resolution as follows:

$$x_m^j = x_{m+2^i}^{j+1} \tag{9}$$

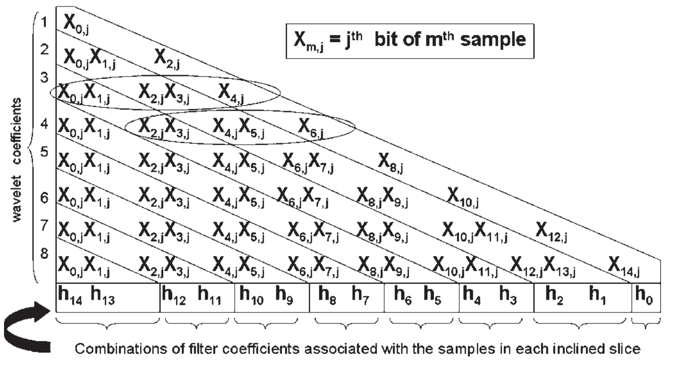
$X_{m,j}$ = jth bit of mth sample
wavelet coefficients
Combinations of filter coefficients associated with the samples in each inclined slice

Fig. 2. Example of address generation logic for first-resolution-level wavelet coefficients with a frame length of 16. x and h denote input data samples and filter coefficients, respectively.

TABLE II
EXAMPLE OF ADDRESS GENERATION LOGIC SAMPLES x_2 AND x_3 OF THE THIRD WAVELET COEFFICIENT IN THE FIRST RESOLUTION LEVEL

$x_{2,j}$	$x_{3,j}$	Memory
0	0	no fetch
0	1	h_1
1	0	h_2
1	1	$h_1 + h_2$

where $m =$ sample number, and $j =$ wavelet coefficient. While designing the address generation logic, this symmetry has been exploited at different levels of resolution. Table II shows how the appropriate filter coefficients are fetched from the memory for different combinations of the jth bit of samples x_2 and x_3 for the third wavelet coefficient computation. This way, we effectively deal with the first issue related to the new addressing scheme raised in Section II-A.

B. Analysis/Synthesis Banks

The analysis and synthesis banks are designed following the equations given in [3] and [6], which represent wavelet analysis and synthesis in dyadic space. The DA-based filter design technique has been applied to realize the finite-impulse response filters of these blocks, which replaces multiplication operations by additions only, thereby making the entire architecture multiplierless. In addition, there are two control units responsible for controlling the analysis and synthesis banks. As shown in Fig. 1, two active-low single-bit signals—"data ready" and "wavelet ready"—enables the analysis bank and synthesis bank, respectively. These signals may be enabled by the user or the previous processing unit in case of a complete system. When the output of these banks are ready, the respective controllers produce two active-low single-bit signals—"analysis over" and "synthesis over"—which indicates the completion of the process.

The complete architecture is designed using fixed-point 2's complement arithmetic for a frame length of 16, with each sample having a 16-b word length. The complete architecture is modeled in very high speed integrated circuit hardware description language (VHDL) and is synthesized using a 0.13-μm CMOS standard cell library. The address generation logic has been implemented following the relationship described in (9) and Fig. 2. The implementation results are discussed next.

IV. PERFORMANCE ANALYSIS

A. Hardware Cost

It was mentioned in Section II-A that in the proposed methodology, there is a possibility that the area saving due to the reduction of memory size may be outweighed by the requirement of extra adders. This section explicitly addresses this issue in terms of the total number of transistor savings. To do that, the total memory requirement using conventional DA is derived first. For simplicity, here, only the memory requirement for the analysis bank is considered. The memory requirement for the synthesis bank can be derived using the same approach. Following the same notations adopted in Section II-B, the number of memory blocks (N_{ij}) required for computing the jth wavelet coefficient at the ith resolution level using conventional DA can be given by

$$N_{ij} = 2^s = 2^{1+(j-1)2^i}. \tag{10}$$

Since each wavelet coefficient is a convolution sum of the input samples and filter coefficients, the computation of j number of wavelet coefficients at the ith level of resolution requires j times of application of DA. It means that (10) has to be iterated j times, where j varies from 1 to $p/2^i$ for each i. Thus, the total memory required for the ith level of resolution can be expressed as

$$N_i = \sum_{j=1}^{p/2^i} N_{ij} = 2 \times (2^p - 1) \Big/ \left((4)^{2^{(i-1)}} - 1\right). \tag{11}$$

The memory requirement for the analysis bank can be computed by summing N_i for all i. However, since the last level of resolution consists of one wavelet coefficient and one residue signal, which requires two more memory blocks (refer to Section II-B), the total conventional memory requirement (CMR) for the complete analysis bank can be expressed as

$$CMR = 2 + \sum_{i=1}^{\log_2 p} N_i. \tag{12}$$

Compared to CMR, the total memory requirement in our proposed methodology is given by (8). The total hardware cost required in our approach is the summation of TMR and the number of extra adders required for generating the appropriate filter coefficients on the fly.

We define a parameter "adder penalty" (AP) to find out the number of required adders. Unlike TMR, AP is dependent on the number of coefficients present per level of resolution. Here, we will restrict ourselves to the basic ripple-carry adder structure for a proof of concept. However, we believe that other structures can be used to improve performance and power. In our proposed methodology, to compute the jth wavelet coefficient at the ith resolution level, we need $(s-1)/2$ number of W-b adders, where s is given by (4). Denoting the adder requirement for the ith stage as P_i and expressing s in terms of j, we get

$$\begin{aligned} P_i &= \sum_{j=1}^{p/2^i} (j-1)2^{(i-1)} \times W \\ &= (p/2^2) \times \left[(p/2^i) - 1\right] \times W. \end{aligned} \tag{13}$$

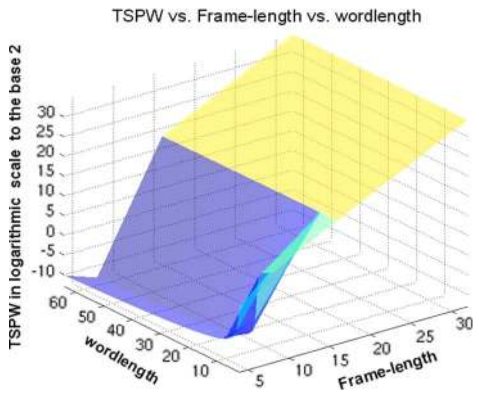

Fig. 3. Variation of $TSPW$ with frame length and word length.

Thus, the total adder requirement for the analysis bank can be given by

$$AP = \sum_{i=1}^{\log_2(p)-1} P_i. \quad (14)$$

In (14), the upper limit of the summation is set to $\log_2(p) - 1$ due to the fact that the last resolution level does not need any addition because of the presence of only one sample in the corresponding wavelet coefficient. Considering that a single-bit memory cell consists of t number of transistors and one single-bit adder consists of Kt number of transistors, transistor savings (TS) can be given as

$$TS = [CMR \times W - (TMR \times W + AP \times K \times W)] \times t. \quad (15)$$

More specifically, from (15), we define a new metric called transistor savings per word length $(TSPW)$ as

$$TSPW = [CMR - (TMR + AP \times K)] \times t. \quad (16)$$

To obtain effective savings in hardware in terms of total transistor count in the proposed methodology, $TSPW$ should be positive. Considering that a single-bit SRAM cell requires six transistors and one single-bit adder requires 18 transistors [18], we have plotted the variation of $TSPW$ with different values of frame length and word length in Fig. 3. It can be observed that for smaller frames (< 8), $TSPW$ is negative, which means that for frame length < 8, the proposed methodology does not achieve hardware savings over the conventional DA-based method. However, for frame-length $= 8$, $TSPW$ is positive for word lengths of up to seven, but above that, $TSPW$ becomes negative. This means that greater than a word length of seven, the rate of quadratic growth in AP dominates over the rate of exponential growth of CMR. Such negative–positivie transition of $TSPW$ can also be observed in Fig. 3. For a longer frame length (> 8), $TSPW$ increases for a fixed word length. This means that for longer frames, the value of CMR dominates over the summation of AP and TMR, resulting in hardware savings with the proposed methodology. It is to be noted that for longer frames as well, keeping the frame length fixed, $TSPW$ starts falling gradually with the increase in word length. However, this falling rate is much less compared to the rate of increase in CMR for longer frames. This overall analysis demonstrates that this proposed methodology is suitable for longer frames.

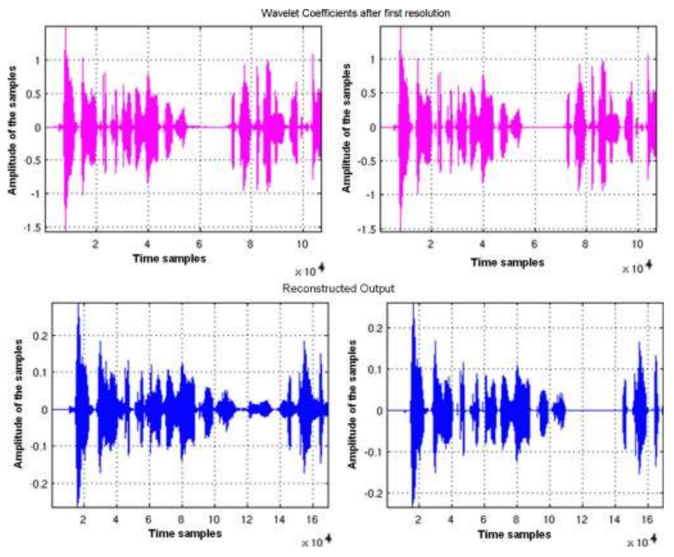

Fig. 4. Comparative study between software and hardware results. The left column represents the generated results of the C-model, and the right column represents the generated results of the VHDL model (word length = 16 b). The top figures represent wavelet coefficients of first resolution level, and the bottom figures represent the reconstructed output.

B. Functional Validation and Error Analysis

To do the functional validation of the proposed methodology, we generated a wavelet analysis model in C language running on a personal computer. The functional output of the C-model is compared with the VHDL model of the proposed methodology-based architecture. As test vectors, we have used 264 000 samples of a female speech signal recorded in real life. The comparison result is shown in Fig. 4. For brevity, we have shown only the wavelet coefficients generated at the first level of resolution for the analysis bank. The result from the C-model is shown on the left side of Fig. 4, while the hardware output as simulated from the VHDL description is shown on the right side. It is evident that the designed architecture shows functional similarity to the software model. However, because of fixed-point implementation, the actual hardware is prone to have errors due to the finite-word-length representation of the input data and the truncation error, which gets accumulated at every arithmetic operation. To examine the overall effect of these sources of error (Fig. 5), we plotted the probability of error with respect to bit position at which such numerical error occurs. This is derived from the VHDL implementation by converting the absolute value of error (E) to binary using $E_2 = |\ln(E)/\ln(2)|$, where E_2 is the bit position (from the most significant bit) at which the error occurs (in magnitude). It is to be noted from Fig. 5 that the error probability reaches to 0.05 at the ninth bit position, which is ignorable for all practical purposes. The maximum probability of error (0.18) occurs at the twenty-first bit position, which physically means that in this case, a 16-bit word length can be considered as a good practical choice.

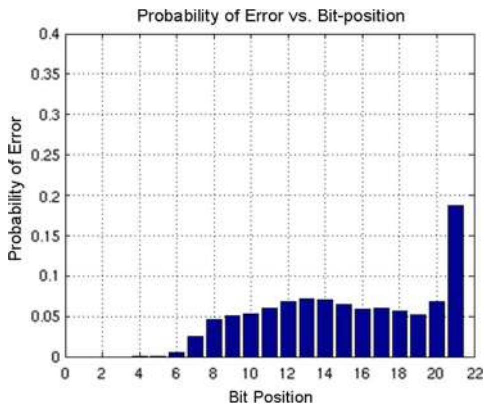

Fig. 5. Probability of error versus bit position in the proposed architecture.

TABLE III
COMPARISONS OF THE PROPOSED ARCHITECTURE WITH OTHER REPORTED ARCHITECTURES

Arch-itect-ure	Word length [bits]	Freq-uency [MHz]	Techn-ology [μm]	Area [mm^2]	Power [mW]
[8]	8	20	1.2	70	500 - 1000
[10]	8	25	0.8	8.5	855
[11]-I	16	50	0.35	25	35.66
[11]-II	16	50	0.35	25	29.43
[12]	—	—	—	—	33.90
[14]	—	—	0.13	—	12.88
[13]	13	–	—	—	15.15
[16]	—	200	0.13	—	9.74
Proposed	16	1	0.13	6.5	0.0468

C. Comparison With Other Architectures

The proposed architecture is synthesized by Synopsys Design Compiler (DC) using 0.13-μm standard cell CMOS technology. The synthesized area and power consumption of the proposed-methodology-based architecture are 6.5 mm^2 and 46.8 μW at 1-MHz frequency for $V_{DD} = 1.2$ V. The power value is obtained by feeding continuously 264 000 16-b random vectors into the synthesized netlist and applying Synopsys Prime Time.

Table III shows the comparison of the area requirement and power consumption of the proposed-methodology-based DWT/IDWT architecture with those of previously proposed architectures. Since different architectures use different technologies, it is unfair to compare them on a one-to-one basis. However, these results are provided to give an insight about the performance of the proposed-methodology-based architecture. Most of the results shown in Table III are taken from [16]. Table III shows that the proposed-memory-reduction-methodology-based DWT/IDWT architecture compares very favorably in terms of area and power with respect to the other reported architectures. It is to be noted that due to the unavailability of an appropriate memory module in our standard cell library, the architecture is implemented using registers. We believe that the use of appropriate memory will significantly reduce the area and power consumption.

V. CONCLUSION

In this brief, a novel methodology for reducing the memory requirement of the DA-based DWT/IDWT architecture has been proposed. Hardware analysis has shown that significant transistor savings can be achieved for frame length > 8. Contrary to the conventional DA, where the total memory size is dependent on the number of coefficients and the frame length, the proposed methodology makes the memory size dependent only on the frame length. This reduction of hardware implies a reduction of power consumption, which is validated through the design of a DWT/IDWT architecture with 16-b word length and a frame length of 16.

REFERENCES

[1] S. Mallat, *A Wavelet Tour of Signal Processing*. New York: Academic, 1999.
[2] G. Knowels, “VLSI architecture for the discrete wavelet transform,” *Electron. Lett.*, vol. 26, no. 15, pp. 1184–1185, Jul. 19, 1990.
[3] K. K. Parhi and T. Nishitani, “VLSI architectures for discrete wavelet transforms,” *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 1, no. 2, pp. 191–202, Jun. 1993.
[4] A. Grzeszczak, T. H. Yeap, and S. Panchanathan, “VLSI architecture for discrete wavelet transform,” in *Proc. Can. Conf. Elect. Comput. Eng.*, Sep. 1994, vol. 2, pp. 461–464.
[5] M. Vishwanath, R. M. Owens, and M. J. Irwin, “VLSI architectures for the discrete wavelet transform,” *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 42, no. 5, pp. 305–316, May 1995.
[6] X. Chen, T. Zhou, Q. Zhang, W. Li, and H. Min, “A VLSI architecture for discrete wavelet transform,” in *Proc. IEEE Int. Conf. Image Process.*, Sep. 1996, vol. 1, pp. 1003–1006.
[7] T. C. Denk and K. K. Parhi, “VLSI architectures for lattice structure based orthonormal discrete wavelet transforms,” *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 44, no. 2, pp. 129–132, Feb. 1997.
[8] A. Grzeszczak, M. K. Mandal, S. Panchanathan, and T. Yeap, “VLSI implementation of discrete wavelet transform,” *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 4, no. 4, pp. 421–433, Dec. 1996.
[9] J. Fridman and E. S. Manolakos, “Distributed memory and control VLSI architectures for the 1-D discrete wavelet transform,” in *Proc. IEEE Workshop VLSI Signal Process.*, Oct. 1994, vol. VII, pp. 388–397.
[10] C. Yu, C. A. Hsieh, and S. J. Chen, “Design and implementation of a highly efficient VLSI architecture for discrete wavelet transform,” in *Proc. IEEE Custom Integr. Circuit Conf.*, 1997, pp. 237–240.
[11] J. M. Jou, Y. H. Shiau, and C. C. Liu, “Efficient VLSI architectures for the biorthogonal wavelet transform by filter bank and lifting scheme,” in *Proc. IEEE Int. Symp. Circuits Syst.*, 2001, pp. 529–532.
[12] M. Alam, C. A. Rahman, W. Badawy, and G. Jullien, “Efficient distributed arithmetic based DWT architecture for multimedia applications,” in *Proc. IEEE Int. Workshop Syst.-on-Chip Real-Time Appl.*, 2003, pp. 333–336.
[13] X. Cao, Q. Xie, C. Peng, Q. Wang, and D. Yu, “An efficient VLSI implementation of distributed architecture for DWT,” in *Proc. IEEE 8th Workshop Multimed. Signal Process.*, 2006, pp. 364–367.
[14] M. Martina and G. Masera, “Low-complexity, efficient 9/7 wavelet filters VLSI implementation,” *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 53, no. 11, pp. 1289–1293, Nov. 2006.
[15] K. G. Oweiss, A. Mason, Y. Suhail, A. M. Kamboh, and K. E. Thomson, “A scalable wavelet transform VLSI architecture for real-time signal processing in high-density intra-cortical implants,” *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 54, no. 6, pp. 1266–1278, Jun. 2007.
[16] M. Martina and G. Masera, “Multiplierless, folded 9/7—5/3 wavelet VLSI architecture,” *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 54, no. 9, pp. 770–774, Sep. 2007.
[17] F. J. Taylor, “An analysis of the distributed arithmetic digital filter,” *IEEE Trans. Acoust., Speech Signal Process.*, vol. ASSP-34, no. 5, pp. 1165–1170, Oct. 1986.
[18] A. Bellaouar and M. I. Elmasry, *Low-Power Digital VLSI Design: Circuits and Systems*. New York: Springer-Verlag, 1995.