

Precision-Aware Self-Quantizing Hardware Architectures for the Discrete Wavelet Transform

Dong-U Lee, *Member, IEEE*, Lok-Won Kim, *Student Member, IEEE*, and John D. Villasenor, *Senior Member, IEEE*

Abstract—This paper presents designs for both bit-parallel (BP) and digit-serial (DS) precision-optimized implementations of the discrete wavelet transform (DWT), with specific consideration given to the impact of depth (the number of levels of DWT) on the overall computational accuracy. These methods thus allow customizing the precision of a multilevel DWT to a given error tolerance requirement and ensuring an energy-minimal implementation, which increases the applicability of DWT-based algorithms such as JPEG 2000 to energy-constrained platforms and environments. Additionally, quantization of DWT coefficients to a specific target step size is performed as an inherent part of the DWT computation, thereby eliminating the need to have a separate downstream quantization step in applications such as JPEG 2000. Experimental measurements of design performance in terms of area, speed, and power for 90-nm complementary metal-oxide-semiconductor implementation are presented. Results indicate that while BP designs exhibit inherent speed advantages, DS designs require significantly fewer hardware resources with increasing precision and DWT level. A four-level DWT with medium precision, for example, while the BP design is four times faster than the digital-serial design, occupies twice the area. In addition to the BP and DS designs, a novel flexible DWT processor is presented, which supports run-time configurable DWT parameters.

Index Terms—Fixed point arithmetic, image coding, very large scale integration (VLSI), wavelet transforms.

I. INTRODUCTION

ALTHOUGH THE JPEG 2000 standard [1] offers considerable coding efficiency and flexibility advantages over the original block DCT-based JPEG standard, it has yet to be widely adopted for several years since the standardization was completed. The reasons for this include the large installed base of devices and software using the block DCT-based JPEG as well as the computational burden involved in performing JPEG 2000 compression. A key element of JPEG 2000 is the discrete wavelet transform (DWT), which recursively decomposes an input image into subbands with different spatial frequency and orientation. The most commonly used DWT filters in JPEG

2000 are the biorthogonal lossless $5/3$ integer and lossy $9/7$ floating-point filter banks. In this paper, we focus on the DWT using $9/7$ filter, which provides very good compression quality but is particularly challenging to implement with high efficiency due to the irrational nature of the filter coefficients.

Although there is a rich literature on different hardware implementations of the DWT [2]–[6] and novel DWT algorithms [7], there has been much less attention directed to approaches in which the precision of the DWT computation is specifically considered as a design goal. The relatively few treatments of this problem include the work of Barua *et al.* in [8], Spiliotopoulos *et al.* in [9], Kotteri *et al.* in [10], and Benkrid *et al.* in [11]. The work in [8] considers the effects of quantizing the lifting coefficients of the $9/7$ DWT. The number of canonical signed digit (SD) terms for the coefficients are varied, and their effects on the peak signal-to-noise ratio (PSNR) and hardware area/speed are evaluated. The work in [9] conducts a similar analysis with the fixed-point data path fixed to 12 bits of integer and 12 bits of fractional precision, which provides sufficient dynamic range to compute a six-level DWT with over 50-dB PSNR. The work in [10] examines the effect on PSNR when quantizing filter coefficients for a convolution-based $9/7$ DWT, and [11] focuses on analyzing dynamic range requirements of the DWT across different subbands and decomposition levels.

In contrast to the previous work, which has been primarily directed to filter coefficients, in this paper, we address simultaneous optimization of not only the coefficient precision but also the internal data paths used in their computation and present a solution that is fully generalized with regard to precision, allowing design of a DWT to any desired accuracy. Using this approach, we show that the optimization technique can be used to minimize operand bit widths in a bit-parallel (BP) architecture and to minimize iterations in a digit-serial (DS) architecture. This enables implementations with a significant improvement in hardware resources and/or execution time while also ensuring that overflows are avoided and precision requirements are met.

In addition, we describe a highly flexible overall DWT architecture in which the target precision and number of DWT levels are configurable at run time. In the approach here, the quantization of the DWT coefficients to a target step size is inherently performed through the process of computing the DWT, thereby eliminating the need for a separate quantization step after the DWT is completed. While any hardware DWT implementation will of course result in DWT coefficients that are quantized in accordance with the precision used to compute and represent the DWT coefficients, traditionally, in JPEG 2000, quantization has been thought of (and thus implemented) as a separate downstream processing step occurring after the DWT. While

Manuscript received July 14, 2009; revised March 18, 2010 and March 21, 2011; accepted July 06, 2011. Date of publication August 04, 2011; date of current version January 18, 2012. This work was supported in part by the National Science Foundation under Grant CCF-0541453 and in part by the Office of Naval Research under Contract N00014-06-1-0253. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Lina J. Karam.

D.-U. Lee is with Mojix Inc., Los Angeles, CA 90025 USA (e-mail: dongu@mojix.com).

L.-W. Kim and J. D. Villasenor are with the Department of Electrical Engineering, University of California at Los Angeles, Los Angeles, CA 90095 USA (e-mail: knoblok@ucla.edu; villa@ee.ucla.edu).

Digital Object Identifier 10.1109/TIP.2011.2163519

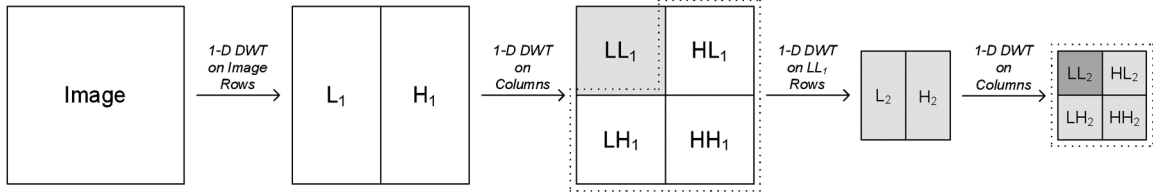


Fig. 1. Illustration of a two-level wavelet decomposition. The dotted portions are the final wavelet transformed data.

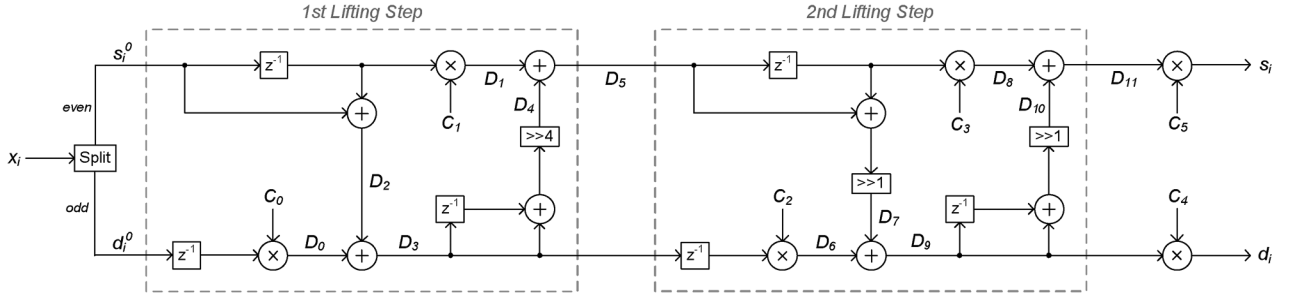


Fig. 2. Flipping structure for the lifting-based 1-D 9/7 DWT.

there are some environments in which the approach of taking a high-precision DWT and then lowering the precision through a subsequent quantization step will be still appropriate, there are also many applications, including those based on configurable hardware, in which it is more optimal to jointly address the DWT computation and coefficient quantization. To examine the specific hardware performance and tradeoffs associated with the solutions presented here, design implementations targeting a 90-nm CMOS process are described, and the quantitative area, speed, and energy characteristics are presented.

The rest of this paper is organized as follows: Section II gives an overview of the lifting-based DWT and JPEG 2000 quantization. Section III describes the design for the precision-aware BP DWT architecture, whereas Section IV discusses the precision-aware DS approach. Section V presents a configurable DS DWT architecture that provides the flexibility to change DWT levels and precision at run time. Section VI provides experimental results, and concluding remarks are given in Section VII.

II. DISCRETE WAVELET TRANSFORM

A. Lifting Approach

There are of course many references describing the DWT. For clarity, we briefly describe DWT aspects that are directly relevant to the subsequent design discussion. Fig. 1 illustrates the steps for performing a two-level DWT on an image. The 1-D DWT is first performed on the rows of the image producing low-frequency L_1 and high-frequency H_1 components. After performing a 1-D DWT again on the columns of L_1 and H_1 , the first level of decomposition is completed, and LL_1 , HL_1 , LH_1 , and HH_1 are obtained. This process can be recursively applied on LL_1 to produce the LL_2 , HL_2 , LH_2 , and HH_2 subbands.

The 9/7 DWT was originally implemented via convolution-based methods, in which low-pass and high-pass FIR filters are employed. In 1998, Daubechies and Sweldens [12] showed that DWT can be decomposed into a finite sequence of lifting steps,

which provides several advantages including lower computation and memory requirements and easier boundary management [13]. When lifting is used, the 9/7 filter can be expressed using the following steps:

$$P(z) = \begin{bmatrix} 1 & \alpha(1+z^{-1}) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ \beta(1+z) & 1 \end{bmatrix} \begin{bmatrix} 1 & \gamma(1+z^{-1}) \\ 0 & 1 \end{bmatrix} \\ \times \begin{bmatrix} 1 & 0 \\ \delta(1+z) & 1 \end{bmatrix} \begin{bmatrix} \zeta & 0 \\ 0 & 1/\zeta \end{bmatrix}$$

where $\alpha = -1.586134342$, $\beta = -0.05298011854$, $\gamma = 0.8829110762$, $\delta = 0.4435068522$, and $\zeta = 1.149604398$.

Fig. 2 illustrates the flipping structure described by Huang *et al.* [2] for the lifting-based 1-D 9/7 DWT. Although the flipping structure shares the same computational complexity with the traditional lifting scheme, it reduces the critical path considerably by flipping computation units with the inverses of multiplier coefficients. Constants C_0, \dots, C_5 are given by

$$\begin{aligned} C_0 &= 1/\alpha = -0.6304636206 \\ C_1 &= 1/(\alpha\beta) = 0.7437502472 \\ C_2 &= 1/(\beta\gamma) = -0.6680671710 \\ C_3 &= 1/(\gamma\delta) = 0.6384438531 \\ C_4 &= \alpha\beta\delta/\zeta = 2.065244244 \\ C_5 &= \alpha\beta\gamma\delta\zeta = 2.421021152. \end{aligned}$$

Input x , which has been dc level shifted by subtracting 2^{B_x-1} , is split between even and odd samples, i.e., s_i^0 and d_i^0 . Symmetric periodic extension [1] of x is performed at the boundaries. Outputs s_i (smooth) and d_i (detail) are the low- and high-pass filtered data.

Fig. 3 depicts a generic high-level architecture of the DWT designs described in this paper. The core is the 1-D DWT module, which performs the actual wavelet transform. It can be implemented via a BP, a DS architecture, or a run-time configurable architecture. The dual-port buffer is large enough to

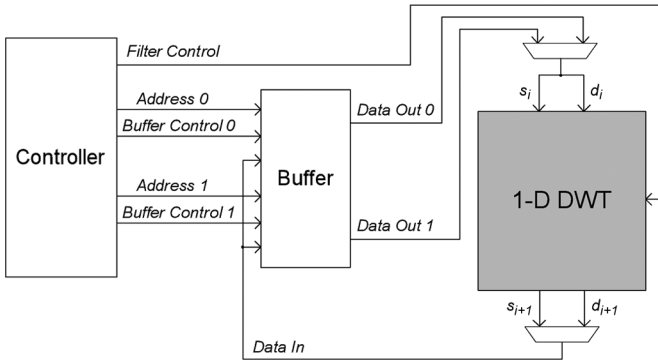


Fig. 3. Generic high-level architecture of the DWT designs presented in this paper.

hold two data frames and is used to store the original raw data, intermediate data, and/or the final transformed data. The controller manages the overall operation of design by generating control signals for the buffer (addresses, write enable, etc.) and the filter (target transform level, transform state, iterations for DS operators, etc.).

B. Quantization

Quantization is a key element for the lossy 9/7 DWT in governing achievable compression performance. The JPEG 2000 standard supports uniform dead-zone quantization, as well as trellis coded quantization [14]. Uniform dead-zone quantization is chosen in this paper due to its simplicity and hardware efficiency. This quantization approach uses equally sized bins, except for a quantizer “dead zone” centered at zero containing a bin double the size of the others. In typical implementations, the quantization step size is specified for the highest resolution subband and is decreased by a factor of two for each subsequent decomposition [1], [14], [15], thereby quantizing the subbands in approximate accordance to their MSE contributions. For example, using this quantization scheme, with reference to Fig. 1, if HL_1 , LH_1 , and HH_1 have a precision of n bits, then LL_2 , HL_2 , LH_2 , and HH_2 should have a precision of $n+1$ bits [16].

III. BP DWT DESIGN

We first consider a BP approach, which is appropriate when computing speed is the primary goal. Given the lifting framework described earlier, the design challenge lies in determining the appropriate number of integer and fractional bits to use in representing all the signals utilized during the computation. In the discussions that follow, two’s complement fixed-point representation is used for all signals. The number of integer bits, fractional bits, and the total number of bits of signal z are denoted by IB_z , FB_z , and B_z , respectively, where $B_z = IB_z + FB_z$. Choosing too many integer bits leads to unnecessary use of resources, whereas too few leads to overflow problems due to insufficient range. Too many fractional bits again wastes resources, whereas using too few causes unwanted loss of precision.

A. Integer Bit-Width Determination

For IB determination, we use the approach described in [17], which is based on computing the roots of the derivatives

TABLE I
BP APPROACH INTEGER BIT WIDTHS (IB) AND FRACTIONAL BIT WIDTHS (FB) FOR A TWO-LEVEL DWT WITH A PRECISION REQUIREMENT OF $FB_{HH_1} = 8$. THE IBs IN THE BRACKETS ARE THE IBs BEFORE THE INCREASE DUE FOR BINARY POINT ALIGNMENT

Signal	IB Level 1		IB Level 2		FB
	Row	Column	Row	Column	
C_0	1	1	1	1	15
C_1	1	1	1	1	18
C_2	1	1	1	1	14
C_3	1	1	1	1	19
C_4	3	3	3	3	12
C_5	3	3	3	3	12
x_i / \hat{x}_i	0 (0)	1 (1)	2 (1)	3 (2)	8 or 19
D_0	2 (0)	3 (0)	4 (1)	5 (1)	16
D_1	0 (0)	1 (1)	2 (1)	3 (2)	18
D_2	2 (1)	3 (2)	4 (2)	5 (3)	16
D_3	2 (2)	3 (2)	4 (3)	5 (3)	15
D_4	0 (-1)	1 (-1)	2 (0)	3 (0)	18
D_5	1 (1)	2 (1)	3 (2)	4 (2)	18
D_6	1 (1)	2 (2)	3 (2)	4 (3)	18
D_7	1 (1)	2 (1)	3 (2)	4 (2)	18
D_8	0 (0)	1 (0)	2 (1)	3 (1)	16
D_9	0 (0)	1 (0)	2 (1)	3 (1)	15
D_{10}	0 (0)	1 (0)	2 (1)	3 (1)	16
D_{11}	0 (0)	1 (0)	2 (1)	3 (1)	16
s_i / \hat{s}_i	1 (1)	2 (1)	3 (2)	4 (2)	19
d_i / \hat{d}_i	1 (1)	2 (1)	3 (2)	4 (2)	19

of each signal. Since the binary point needs to be aligned for additions, the two addition operands need to share the same IB. Hence, for the 1-D DWT shown in Fig. 2, the following signal pairs need to share the same IB, i.e., (D_0, D_2) , (D_1, D_4) , (D_6, D_7) , and (D_8, D_{10}) . Practically, this implies that the IB should be set to the larger IB of the two, e.g., $IB_{D_0} = IB_{D_2} = \max(IB_{D_0}, IB_{D_2})$. Furthermore, s_i and d_i are either data sets for the subsequent DWT or the final outputs; they should also share the same IB.

Let x_i be the input data, s_i and d_i be the output data for DWT level i row processing, \hat{x}_i be the input data, and \hat{s}_i and \hat{d}_i be the output data for DWT level i column processing. Note that x_{i+1} will be a subset of \hat{s}_i . s_i and d_i will contribute to L_i and H_i , respectively, whereas \hat{s}_i contributes to LL_i and HL_i , and \hat{d}_i contributes to LH_i and HH_i .

Table I shows the integer bit widths for the 1-D DWT signals in Fig. 2 for a two-level DWT. Input x_0 is assumed to be a dc level shifted 8-bit value over $[-0.5, 0.5]$. There are considerable variations in dynamic range across signals, and the dynamic range increases with DWT level (due to deepening computation chain). In the table, certain multiplication outputs require fewer IBs than the sum of IBs of the multiplicands. While this may seem counterintuitive, it occurs because the operands rarely utilize the full dynamic range of their allocated bit widths. For instance, examining multiplication $D_6 = C_2 \times D_3$ in level 1 row processing, $C_2 = -0.6684$ (1 IB required) and $D_3 = [-1.3128, 1.3074]$ (2 IBs required), and hence, $D_6 = [-0.8734, 0.8770]$ (1 IB required).

B. Fractional Bit-Width Optimization

The fractional bit-width optimization is executed in two steps, i.e., a static step based on analytical models to obtain the initial set of bit widths [17], followed by a dynamic step based on simulation that further reduces the bit widths using a PSNR delta threshold. The target precision metric used for the static step is the unit in the last place (ulp) error criterion, which is a way of specifying the worst case (maximum absolute) error. The static step finds the set of bits that guarantee less than 2-ulp error at the final quantized DWT outputs. If a signal has 8 fractional bits, its ulp will be 2^{-8} , and the optimization step will guarantee a worst case error of 2^{-7} .

1) *Static Optimization*: The worst case (maximum absolute error) quantization errors for truncation and round-to-nearest are given by

$$\text{Truncation: } E_z = \max(0, 2^{-\text{FB}_z} - 2^{-\text{FB}_{z'}}) \quad (1)$$

$$\text{Round-to-nearest: } E_z = \begin{cases} 0, & \text{if } \text{FB}_z \geq \text{FB}_{z'} \\ 2^{-\text{FB}_z-1}, & \text{otherwise} \end{cases} \quad (2)$$

where $\text{FB}_{z'}$ is the full precision of unquantized z . For addition $z = x + y$ and multiplication $z = x \times y$, $\text{FB}_{z'}$ is defined as follows:

$$z = x + y: \quad \text{FB}_{z'} = \max(\text{FB}_x, \text{FB}_y) \quad (3)$$

$$z = x \times y: \quad \text{FB}_{z'} = \text{FB}_x + \text{FB}_y. \quad (4)$$

Assuming truncation, for addition $z = x + y$ and multiplication $z = x \times y$, worst case error E_z of z is given by

$$z = x + y: \quad E_z = E_x + E_y + \max(0, 2^{-\text{FB}_z} - 2^{-\text{FB}'_z}) \quad (5)$$

$$z = x \times y: \quad E_z = \max(y)E_x + \max(x)E_y + E_x E_y + \max(0, 2^{-\text{FB}_z} - 2^{-\text{FB}'_z}). \quad (6)$$

Applying the above to the 1-D DWT in Fig. 2 with truncation for arithmetic operations and rounding for constants, we get

$$\begin{aligned} E_{D_0} &= \max(d_i^0) \times 2^{-\text{FB}_{C_0}-1} + C_0 \times E_{d_i^0} \\ &\quad + E_{d_i^0} \times 2^{-\text{FB}_{C_0}-1} \\ &\quad + \max\left(0, 2^{-\text{FB}_{D_0}} - 2^{-\text{FB}_{C_0}-\text{FB}_{d_i^0}}\right) \end{aligned} \quad (7)$$

$$\begin{aligned} E_{D_1} &= \max(s_i^0) \times 2^{-\text{FB}_{C_1}-1} + C_1 \times E_{s_i^0} \\ &\quad + E_{s_i^0} \times 2^{-\text{FB}_{C_1}-1} \\ &\quad + \max\left(0, 2^{-\text{FB}_{D_1}} - 2^{-\text{FB}_{C_1}-\text{FB}_{s_i^0}}\right) \end{aligned} \quad (8)$$

$$E_{D_2} = 2 \times E_{s_i^0} + \max\left(0, 2^{-\text{FB}_{D_2}} - 2^{-\text{FB}_{s_i^0}}\right) \quad (9)$$

$$\begin{aligned} E_{D_3} &= E_{D_0} + E_{D_2} \\ &\quad + \max\left(0, 2^{-\text{FB}_{D_3}} - 2^{-\max(\text{FB}_{D_0}, \text{FB}_{D_2})}\right) \end{aligned} \quad (10)$$

$$\vdots \quad (11)$$

$$\begin{aligned} E_{s_i} &= \max(D_{11}) \times 2^{-\text{FB}_{C_5}-1} + C_5 \times E_{D_{11}} \\ &\quad + E_{D_{11}} \times 2^{-\text{FB}_{C_5}-1} \\ &\quad + \max(0, 2^{-\text{FB}_{s_i}} - 2^{-\text{FB}_{C_5}-\text{FB}_{D_{11}}}) \end{aligned} \quad (12)$$

$$\begin{aligned} E_{d_i} &= \max(D_9) \times 2^{-\text{FB}_{C_4}-1} + C_4 \times E_{D_9} \\ &\quad + E_{D_9} \times 2^{-\text{FB}_{C_4}-1} \\ &\quad + \max(0, 2^{-\text{FB}_{d_i}} - 2^{-\text{FB}_{C_4}-\text{FB}_{D_9}}). \end{aligned} \quad (13)$$

These error expressions consider the worst case error bounds at each node and can be recursively derived for any number of DWT levels. Let L be the total number of DWT levels, i denote level i of the DWT, and FB_{HH_1} be the number of fractional bits of LL_1 . When processing rows, s_i and d_i are intermediate results and are quantized to the appropriate internal precision. When processing columns, however, since d_i is always the final output, it is quantized to $\text{FB}_{LL_1} - 1 + i$ bits. If $i < L$, the portion of s_i that contributes to LL_i will be passed to the next DWT stage and hence is quantized to the internal precision. All other cases, s_i is the final output and is quantized to $\text{FB}_{LL_1} - 1 + i$ bits.

The internal data paths are quantized using standard truncation (truncation toward infinity by chopping off least significant bits), whereas the final DWT outputs are quantized toward zero, as discussed further in Section II-B. The bit widths of the internal data paths are found using the error expressions above in conjunction with simulated annealing (see [17]). Since the quantization scheme of JPEG 2000 uses increasing precision with i , the bit widths are dominated by the precision requirements of $i = L$.

To validate the analytical model for fractional bits, a series of experimental computations were performed using both a finite precision implementation with the target number fractional bits and with double precision. For example, according to the model, when a two-level transform with a precision level of $\text{FB}_{HH_1} = 8$ is desired (and, hence, $\text{FB}_{HH_2} = 9$), the fractional bits should be $\text{FB}_{D_0} = 19$, $\text{FB}_{D_1} = 21$, etc. Experiments performed using several images, as well as random test vectors, show an actual worst case error of 3.86×10^{-3} (0.99 ulp) for HH_1 and 1.91×10^{-3} (0.98 ulp) for HH_2 , which are well within the 2-ulp error bound.

2) *Dynamic Optimization*: The analytical optimization scheme is conservative in the sense that it assumes that the worst case error can concurrently occur at all nodes, which is extremely likely to occur in practice. As a result, the computed FBs will be in general larger than required. Moreover, it is PSNR, not internal arithmetic accuracy, which is used in practice for numerical assessment of images represented by inverse transforming a quantized DWT representation. In order to tune the precision decisions to more closely relate to PSNR, we perform a secondary simulation-based bit-width refinement step to further reduce the FBs. Using binary search, the FBs are uniformly reduced until either 1) the PSNR loss of a set of test images fall above 0.1 dB compared with the statically optimized set or 2) the ulp error of any of the DWT outputs exceed 2 ulp. This process typically reduces the FBs by approximately 20%. The last column in Table I shows the dynamically optimized FBs for a two-level DWT with a precision requirement of $\text{FB}_{HH_1} = 8$. For example, these precision achieve a PSNR of 43.7 dB for the 512×512 Goldhill image [18].

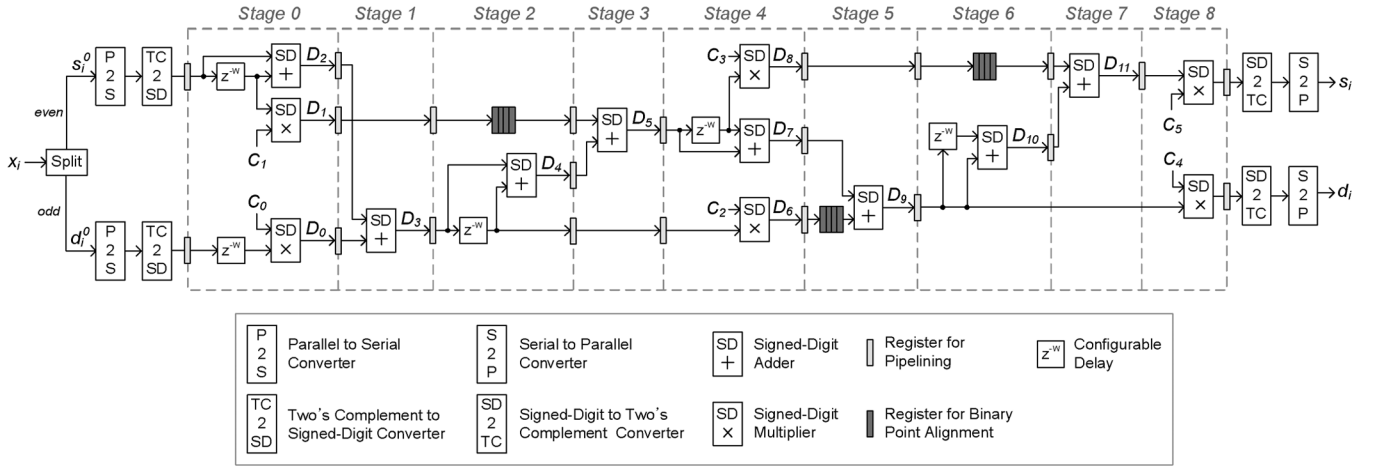


Fig. 4. DS 1-D 9/7 DWT data flow.

IV. DS DWT DESIGN

A. Overview

While DS arithmetic has a significant advantage over BP in terms of circuit area, a key challenge in DS design involves minimizing the number of iterations. For the DS representations used here, we use a radix-2 SD redundant number system [19]. Due to redundancy, SD operations do not propagate carries and hence are able to run in most significant digit first (MSDF) [19] mode (also known as online arithmetic). This MSDF property makes it attractive for the DS DWT approach since it allows for varying the number of iterations to obtain different precision. In radix-2 SD, the following set is used to represent a digit: $\{-1, 0, 1\}$. We use binary bits $b'10$, $b'00$, and $b'01$ to indicate -1 , 0 , and 1 , respectively.

Fig. 4 illustrates the DS 1-D 9/7 DWT solution. The incoming two's complement data is first serialized and converted into SD representation. The serial SDs are then passed into the DS DWT, which is partitioned into nine pipeline stages that run in parallel. After the last stage, the DWT-transformed data is converted back into two's complement representation [20] and parallelized into words. This approach reduces the memory requirement since two's complement occupies half the area of the equivalent SD representation. Both SD addition and SD multiplication produce one digit per cycle, starting from the most significant digit. For SD addition, we use the architecture described by Koren [21]. For SD multiplication, where one of the operands is a constant, we use a structure similar to the one proposed by Ercegovac and Lang [22].

B. Integer Width Determination

As in the BP approach, the goal here is to use the minimum number of integer digits for each signal while avoiding overflow. Moreover, the number of integer digits of the addition operands need to be identical for binary point alignment. The binary point of a digit can be adjusted via increasing or decreasing the number of integer digits. This is easily achievable for the BP case by simple shifting. In MSDF, however, the number of integer digits needs to be adjusted by inserting and removing delay elements, e.g., registers.

For the addition $z = x + y$, the SD adder leads to $ID_z = \max(ID_x, ID_y) + 2$, and for the multiplication $z = x \times y$, the SD multiplier leads to $ID_z = ID_x + ID_y + 2$, where ID_z is the number of integer digits in z . Due to the redundant nature of the SD operators used here, the addition and multiplication exhibit one and two extra bits, respectively, with respect to the true dynamic range of z . The IDs for the constants are $ID_{C_0, \dots, C_3} = 0$ and $ID_{C_4, C_5} = 2$. The integer digits are one less than the ones required in the two's complement BP case due to the larger dynamic range of the SD.

We conduct the integer digit analysis for $i = 0$. The analysis ensures that the number of integer digits for all paths increase by one with i , ensuring that variable shifters (which are expensive in hardware) are not required. s_0^o and d_0^o are the pixels from an image whose range is over $[-0.5, 0.5)$, and hence, $ID_{s_0^o} = ID_{d_0^o} = 0$. Knowing that $ID_{C_0} = 0$ leads to $ID_{D_0} = ID_{s_0^o} + ID_{C_0} + 2 = 2$. Following a similar derivation gives $ID_{D_1} = 2$ and $ID_{D_4} = 6$. These correspond to the input operands of an adder and thus need to share the same ID. The true dynamic range of D_1 and D_4 are 0 and -1 IDs, respectively. One possibility is to reduce ID_{D_1} by two digits and ID_{D_4} by six digits, making them both zero IDs. The only way to achieve this is by removing registers, but there are no registers inherent to DWT that can be eliminated. Therefore, instead, we add four more registers to the path of D_1 to achieve $ID_{D_1} = 6$.

Similar strategies are employed in other parts of the 1-D DWT where the binary points need to be aligned. Note that for D_9 path, pipelining registers after stages 6 and 7 are removed to save hardware. After the two multipliers in stage 8, there are 20 integer digits for the s_i path and 16 integer digits for the d_i path. The number of integer bits needed to represent the true dynamic range of the two signals is 1 integer bit for each. Thus, during the SD to two's complement conversion process, we ignore the first 19 and 15 digits of the s_i and d_i paths. These extra integer parts are essentially byproducts of the ID increase needed for binary point alignment.

C. Minimizing the Number of DS Iterations

In a DS implementation, increasing the number of iterations gives more precision but costs more execution time. The goal

TABLE II
NUMBER OF DIGITS ALLOCATED FOR THE COEFFICIENTS IN A THREE-LEVEL
DWT WITH A PRECISION REQUIREMENT OF $FB_{HH_1} = 8$

Signal	C_0	C_1	C_2	C_3	C_4	C_5
Digits	17	16	18	18	20	20

TABLE III
NUMBER OF ITERATIONS ALLOCATED TO EACH 1-D DWT STEP IN A
THREE-LEVEL DWT WITH A PRECISION REQUIREMENT OF $FB_{HH_1} = 8$

Level	1		2		3	
	Row	Column	Row	Column	Row	Column
Iterations	25	25	26	27	27	27

of iteration optimization is thus to use the minimum number of iterations while meeting the specified error requirement. This is analogous to determining the minimum number of fractional bits with the direct approach. The worst case error for the DS addition $z = x + y$ is given by

$$E_z = E_x + E_y + \max(0, 2^{-FD_z} - 2^{-FD_x}) + \max(0, 2^{-FD_z} - 2^{-FD_y}) \quad (14)$$

where the last two terms are quantization errors due to using a subset of digits of x and y , which is a function of the number of iterations. The worst case error for DS multiplication $z = x \times y$ where y is a constant is given by

$$E_z = \max(x) \times 2^{-FD_y-1} + y \times \max(0, 2^{-FD_x} - 2^{-FD_x'}) + \max(0, 2^{-FD_x} - 2^{-FD_x'}) \times 2^{-FD_y-1} \quad (15)$$

where term 2^{-FD_y-1} is due to rounding the constant and term $\max(0, 2^{-FD_x} - 2^{-FD_x'})$ is due to using subset $-FD_x'$ from FD_x for the multiplication.

Using the DS addition and multiplication error expressions in (14) and (15), static and dynamic coefficient and iteration optimization are performed. Table II shows the optimized numbers of digits for the coefficients in a three-level DWT with a precision requirement of $FB_{HH_1} = 8$, and Table III shows the iterations at each DWT stage. The iterations increase with increasing level, in accordance with the increasing accuracy requirements discussed previously.

V. RUN-TIME CONFIGURABLE DS ARCHITECTURE

The BP and digit-serial architectures discussed in Sections III and IV enable optimized computation of a single level of the DWT at a single precision requirement. However, many DWT applications involve multilevel DWT decompositions. Thus, it is of high interest to have a single reconfigurable DWT processor that supports different DWT levels (e.g., levels 1–7) and precision (e.g., precision 2–14) at run time. Varying these parameters provides the ability to vary compression ratios, image quality, and processing time. Adding this flexibility to the BP approach would mean that all of the operators would need to be large enough to support the highest level and precision. When performing a DWT at a low level and/or precision, this would involve significant hardware inefficiency. The DS approach, however, is well suited for this reconfigurability as the number of it-

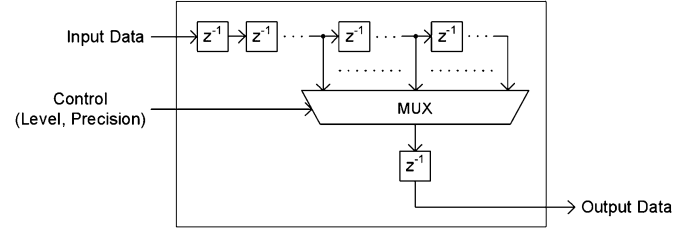


Fig. 5. Configurable shift register used in the run-time configurable design. This structure allows the amount of delay to be easily and flexibly changed in response to changes in the level of precision needed in the computations.

erations and, thus, the precision can be simply varied at run time as a function of the DWT level.

In order to make the DS approach configurable, the following changes are required.

- 1) A table containing the number of iterations required for each operator for the range of target combinations of DWT levels and precision is generated. The entries of this table is determined using the techniques described in Section IV.
- 2) Shift registers that need to delay by a word (such as the configurable delay elements in Fig. 4) need to be large enough to support the widest possible (which will most likely be the highest level and precision).
- 3) These shift registers need to be configurable to support different amounts of delays. This is achieved by utilizing a multiplexer, as illustrated in Fig. 5. The multiplexer taps off various stages of the delay chain, effectively serving as a run-time configurable shift register.
- 4) Extra control circuitry is needed, which indicate correct end of frame and end of line, etc., for different levels and precision.

VI. IMPLEMENTATION RESULTS

Designs are synthesized using Synopsys Design Compiler with the standard-cell library of Infineon 90-nm CMOS technology. Power results are obtained via Synopsys Power Compiler, which performs gate-level power simulation using user-supplied data (images).

A. PSNR

Table IV shows the PSNR variations of 512×512 DWT transformed images at varying precision (FB_{HH_1}). Images are DWT transformed using the precision-optimized implementations described above and an inverse DWT (IDWT) is performed in software using double-precision floating-point to ensure the PSNR numbers accurately measure the impact of the forward DWT computation. The PSNRs shown here are the average PSNRs of the DS implementations taken across different levels. The PSNRs for BP designs are very similar (less than 0.5-dB variation) since both BP and DS are optimized for the same precision target. Variations across different levels of DWT are also minimal (typically less than 1 dB), which is perhaps due to respecting the quantization requirement discussed in Section II-B. Two extra bits of precision gives us approximately 13 dB of additional PSNR on average. Although PSNRs at high precision may appear excessive, in practical applications, the IDWT

TABLE IV
PSNR RESULTS IN DECIBELS OF VARIOUS 512×512 DWT TRANSFORMED IMAGES WITH DIFFERENT PRECISION (FB_{HH_1})

Precision [bits]	Boat	Goldhill	Lena	Mandrill	Peppers
2	29.6	28.5	30.9	23.7	30.7
4	35.5	33.6	35.8	31.8	35.1
6	43.9	43.5	42.7	43.8	42.8
8	56.5	57.1	55.8	57.6	56.6
10	70.2	69.9	69.5	70.5	70.1
12	82.9	82.1	82.2	82.7	82.6
14	95.1	94.3	94.4	94.8	94.8

will be likely implemented using a fixed point, leading to additional PSNR loss on the final image. In addition to the results in Table IV, PSNRs for a double-precision floating-point DWT were also computed. For level 1, the error due to double-precision varied from 315 to 318 dB, and for a seven-level DWT, the PSNR varied from 301 to 307 dB. Of course, these PSNR values are much higher than the results shown in the table, which of course involve much lower precision than double-precision floating point.

B. Speed

Since BP operators process a word every cycle and DS operators process a word in multiple cycles, DS architectures require more clock cycles. However, DS operators are fast with no carry propagation. Furthermore, the speed of the DS operators is independent of the word size. Fig. 6 compares the average execution times for the BP and DS implementations for computing a word in a 1-D DWT. All DS designs in the figure have a maximum clock speed of 435 MHz. In contrast, the operators in BP designs are slower due to carry propagation, which is proportional to the operand bit-width, which in turn is a function of the target precision. The maximum clock speeds of the BP designs range from 41 to 101 MHz. The DS exhibits a highly linear behavior with precision and decomposition level. This is primarily due to the constant clock speed. With BP designs, there is a lower rate of increase in execution times with precision and level. Although the DS designs have faster clock speeds, their overall execution times are considerably slower due to their multicycle characteristics.

C. Area

Fig. 7 compares the area variations of BP and DS implementations. The gate counts shown here include all of components in Fig. 3, with the exception of the buffer. As discussed in Section II-A, the buffer is large enough to store two frames worth of data. It was omitted because memories can be realized in several different ways and because we wanted the results to highlight the key components focused in this paper. Both implementations show a relatively linear increase in area, with increasing precision and level. With the BP designs, the area rapidly grows with increased precision requirements, mainly due to the growing two's complement BP operator sizes. DS operators used here are fixed at one digit per cycle and achieve high precision results by increasing the number of iterations.

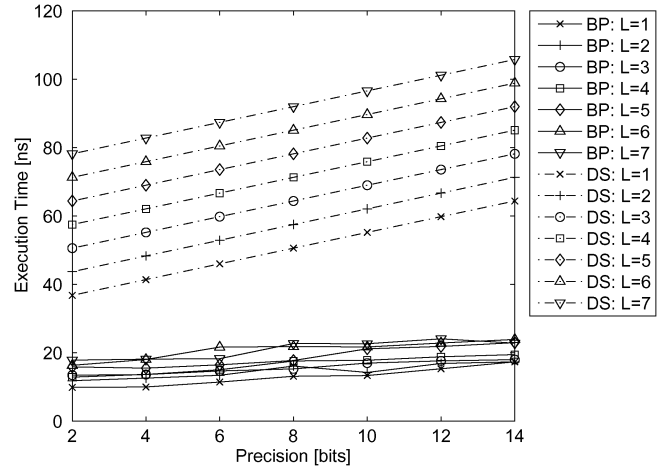


Fig. 6. Execution time (time for one 1-D DWT filtered word) comparisons of BP and DS implementations. L is the number of DWT levels performed.

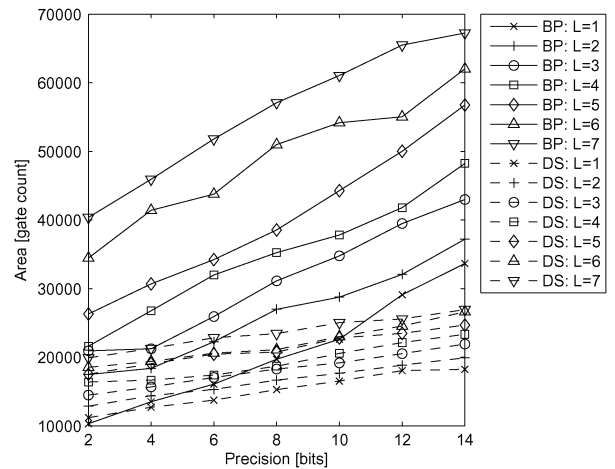


Fig. 7. Area comparisons of BP and DS implementations. L is the number of DWT levels performed.

The only hardware increase with increasing precision requirements are 1) the depths of shift registers between stages and 2) a moderate gain in multiplier complexity to support wider coefficients. Since these two overheads are relatively small, there is a much slower rate of area increase with the DS designs. Interestingly, at very low precision and levels, DS designs are somewhat larger than the BP designs. This is due to the fixed overheads associated with the DS approach such as two's complement from/to signed-digit converters and control circuitry for serial operators. Beyond level 1 and precision of 2 bits, however, these overheads become less significant, and the DS designs become increasingly cost effective. At level 7 and precision 14 bits, the DS implementation is approximately one third of the area of its BP counterpart.

The run-time configurable DS DWT processor, which can perform any combination of levels 1–7 and precision of 2–14 bits, was also implemented using Synopsys Design Compiler for 90-nm Infineon CMOS. The resulting processor is capable of running at 435 MHz (same as the fixed DS designs) and requires 32 908 gates, which is 22% larger than the fixed DS design for level 7 and precision 14. This 22% area increase is due to the overheads discussed in Section V.

TABLE V

COMPARISONS BETWEEN BP AND DS IMPLEMENTATIONS FOR A FOUR-LEVEL 8-BIT DWT. THE PROCESSING TIME IS THE TIME TAKEN FOR THE ENTIRE FOUR-LEVEL DWT ON A 512×512 IMAGE. THE POWER RESULTS ARE OBTAINED WITH SYNOPSIS DESIGN COMPILER USING A SET OF TEST VECTORS. THE STATIC ENERGY IS COMPUTED ASSUMING A 5-s STANDBY TIME

Approach	Area [gates]	Clock Speed [MHz]	Processing Time [ms]	Dynamic Power [mW]	Static Power [mW]	Dynamic Energy [mJ]	Static Energy [mJ]	Total Energy [mJ]
Bit-Parallel	35228	56	7.4	6.8	3.5	0.05	17.3	17.4
Digit-Serial	18680	435	29.8	23.5	1.7	0.70	8.6	9.3

D. Power

Power dissipation is determined by the combination of static power and dynamic power. Static power largely results from transistor leakage current, whereas dynamic power is primarily due to switching activities for charging and discharging load capacitance. Although, traditionally, dynamic power has been dominant, static power is becoming increasingly significant as process geometry shrinks [23].

The dynamic energy consumption is given by

$$E_D = C_L \times V_{DD}^2 \times f \times T \quad (16)$$

where C_L is the load capacitance (which is largely a function of area), V_{DD} represents the supply voltage, f is the clock frequency, and T is the execution time. Considering constant V_{DD} , the C_L term favors the DS approach due to its low area requirement, but the f and T terms favor the BP approach due to its lower clock speed and processing time. Column 7 in Table V shows dynamic energy of the two approaches when performing a four-level 8-bit DWT on a 512×512 image. Although the DS design requires half the area of the BP design, its high clock speed and processing time cause it to consume 12 times more dynamic energy.

The static energy consumption is given by

$$E_S = V_{DD} \times I_S \times T \quad (17)$$

where V_{DD} represents the supply voltage, I_S is the leakage current (which is proportional to the area), and T is the power-on time. It is shown that the DS approach has a considerable static energy advantage due to its lower area requirement, which is shown in the second to the last column in Table V.

To conduct a representative overall energy consumption assessment between the two methods, we consider an image processor on a camera. We assume that when the camera is powered on, the image processor is initially in standby mode (and thus dissipating static power). When the user selects a scene to shoot and presses the shutter, the image processor switches to active mode (and thus dynamic power) to process the image taken by the user. A 512×512 image and four-level 8-bit DWT is assumed with a standby time of 5 s. With reference to the last three columns in Table V, although the dynamic energy usage of the DS approach is much higher, the device is in active mode only a very small fraction of the time, and thus, the dominant energy consumption is static, resulting in an energy advantage of about a factor of two for the DS design.

Clearly, this ratio is a function of the standby time assumption. Fig. 8 provides an energy comparison as standby time is varied. Note that at 0-s standby time, only dynamic energy is

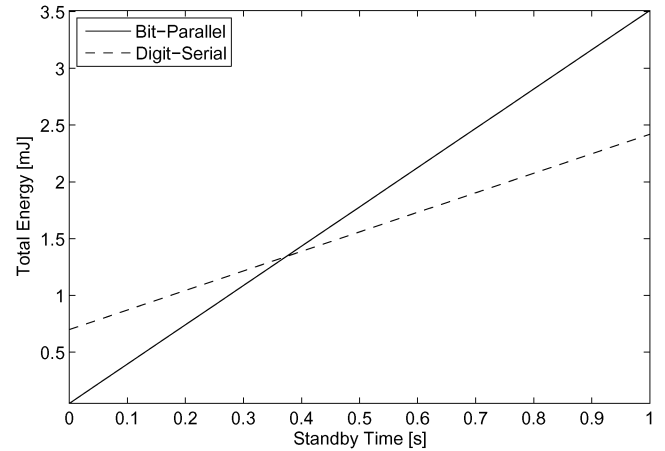


Fig. 8. Total energy consumption variation with standby time for a four-level 8-bit DWT on a 512×512 image.

consumed. If, for example, the standby time is reduced to 0.2 s, the ratio of DS energy to BP energy becomes 1:4, i.e., the DS approach is 40% less energy efficient than the BP approach. Therefore, applications with high quality and throughput requirements such as high-definition video will be better suited with the BP approach and high precision, whereas applications with lower resolution requirements could be potentially more appropriately handled with a DS approach and lower precision. While the specific power and energy values will of course vary with many factors, including not only standby time but also precision and number of levels, the larger message is that the framework presented here allows these factors to be readily compared, thereby making it possible to choose a design that is optimized for a given set of constraints and/or requirements.

E. Comparisons With a Related Work

Fig. 9 compares BP designs using the proposed bit-width optimized approach and the fixed bit-width approach in [9]. The DWT level is fixed at two, and the 512×512 “Goldhill” image is used to obtain the PSNRs. The fixed-point data path in [9] is tied at 24 bits for all signals, i.e., 12 integer bits and 12 fractional bits. Looking at the PSNR results (solid lines), while the proposed and [9] exhibit similar image quality at very low precision, the gap widens as the precision increases. When the target DWT precision is high, the fixed 24-bit data path in [9] bottleneck the transform accuracy, resulting in a PSNR “knee” of around 47 dB. With the approach in this paper, however, bit widths are varied according to the DWT precision requirement, thus producing a continuously increasing trend in the figure.

Area comparisons between the two are given in the same figure (dashed lines). The area for [9] is constant since the

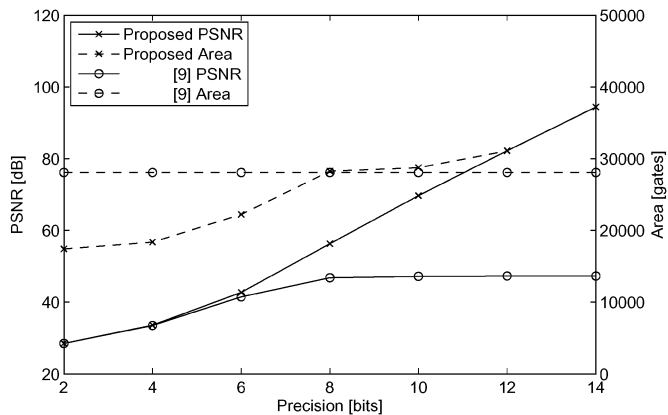


Fig. 9. PSNR and area comparisons of the proposed bit-width optimized BP approach against the approach in [9]. The DWT level is fixed at two, and the “Goldhill” image is used for PSNR computation.

widths of the data paths are kept constant. The area for the proposed approach changes in order to tailor the data paths for the target precision. An interesting observation is that at low precision (below 8 bits), the proposed design occupies considerably less area while providing equal or better image quality. This is because, in contrast to the proposed, which allocates the bits adaptively, the fixed data path approach can lead to bits being wasted (surplus dynamic range, overly precise computations, etc.).

VII. CONCLUSION

We have presented precision-aware approaches and associated hardware implementations for performing the DWT. Both BP and DS design methodologies and results have been presented. These methods enable use of an optimal amount of hardware resources in the DWT computation. Moreover, this framework enables quantization, which is traditionally performed after the DWT in algorithms such as JPEG 2000, to be specifically incorporated into the computation of the DWT itself. We have also presented a highly flexible configurable DWT processor and examined the energy and power tradeoffs between the associated BP and DS designs, in particular, highlighting the differing respective roles of static and dynamic power in each. We believe that design methods and architectures such as those presented here play a significant role in the design of future energy- and precision-optimized DWT implementations.

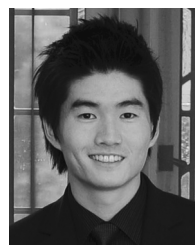
ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for their comments.

REFERENCES

- [1] M. Rabbani and R. Joshi, “An overview of the JPEG 2000 still image compression standard,” *Signal Process.: Image Commun.*, vol. 17, no. 1, pp. 3–48, Jan. 2002.
- [2] C. Huang, P. Tseng, and L. Chen, “Flipping structure: An efficient VLSI architecture for lifting-based discrete wavelet transform,” *IEEE Trans. Signal Process.*, vol. 52, no. 4, pp. 1080–1089, Apr. 2004.
- [3] K. Kotteri, S. Barua, A. Bell, and J. Carletta, “A comparison of hardware implementations of the biorthogonal 9/7 DWT: Convolution versus lifting,” *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 52, no. 5, pp. 256–260, May 2005.

- [4] C. Cheng and K. Parhi, “High-speed VLSI implementation of 2-D discrete wavelet transform,” *IEEE Trans. Signal Process.*, vol. 56, no. 1, pp. 393–403, Jan. 2008.
- [5] B. Wu and C. Lin, “A high-performance and memory efficient pipeline architecture for the 5/3 and 9/7 discrete wavelet transform of JPEG2000 codec,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 15, no. 12, pp. 1615–1628, Dec. 2005.
- [6] C. Xiong, J. Tian, and J. Liu, “Efficient architectures for two-dimensional discrete wavelet transform using lifting scheme,” *IEEE Trans. Image Process.*, vol. 16, no. 3, pp. 607–614, Mar. 2007.
- [7] N. Mehrseresh and D. Taubman, “An efficient content-adaptive motion-compensated 3-D DWT with enhanced spatial and temporal scalability,” *IEEE Trans. Image Process.*, vol. 15, no. 6, pp. 1397–1412, Jun. 2006.
- [8] S. Barua, K. Kotteri, A. Bell, and J. Carletta, “Optimal quantized lifting coefficients for the 9/7 wavelet,” in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2004, vol. 5, pp. 193–196.
- [9] V. Spiliotopoulos, N. Zervas, Y. Andreopoulos, G. Anagnostopoulos, and C. Goutis, “Quantization effect on VLSI implementations for the 9/7 DWT filters,” in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2001, vol. 2, pp. 1197–1200.
- [10] K. Kotteri, A. Bell, and J. Carletta, “Design of multiplierless, high-performance, wavelet filter banks with image compression applications,” *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 51, no. 3, pp. 483–494, Mar. 2004.
- [11] A. Benkrid, K. Benkrid, and D. Crookes, “Optimal wordlength calculation for forward and inverse discrete wavelet transform architectures,” *Opt. Eng.*, vol. 43, no. 2, pp. 455–463, Feb. 2004.
- [12] I. Daubechies and W. Sweldens, “Factoring wavelet transforms into lifting steps,” *J. Fourier Anal. Appl.*, vol. 4, no. 3, pp. 247–269, May 1998.
- [13] T. Acharya and C. Chakrabarti, “A survey on lifting-based discrete wavelet transform architectures,” *J. VLSI Signal Process.*, vol. 42, no. 3, pp. 321–339, Mar. 2006.
- [14] M. Marcellin, M. Lepley, A. Bilgin, T. Flohr, T. Chinen, and J. Kasner, “An overview of quantization in JPEG 2000,” *Signal Process.: Image Commun.*, vol. 17, no. 1, pp. 73–84, Jan. 2002.
- [15] K. Varma and A. Bell, “JPEG2000—Choices and tradeoffs for encoders,” *IEEE Signal Process. Mag.*, vol. 21, no. 6, pp. 70–75, Nov. 2004.
- [16] M. Weeks, “Precision for 2-D discrete wavelet transform processors,” in *Proc. IEEE Workshop Signal Process. Syst.*, 2000, pp. 80–89.
- [17] D. Lee and J. Villasenor, “A bit-width optimization methodology for polynomial-based function evaluation,” *IEEE Trans. Comput.*, vol. 56, no. 4, pp. 567–571, Apr. 2007.
- [18] Fractal Coding and Analysis Group, Image Repository: Greyscale Set 2 Univ. Waterloo. Waterloo, ON, Canada, 2009 [Online]. Available: <http://links.uwaterloo.ca/Repository.html>
- [19] M. Ercegovac and T. Lang, *Digital Arithmetic*. San Mateo, CA: Morgan Kaufmann, 2004.
- [20] M. Ercegovac and T. Lang, “On-the-fly conversion of redundant into conventional representations,” *IEEE Trans. Comput.*, vol. C-36, no. 7, pp. 895–897, Jul. 1987.
- [21] I. Koren, *Computer Arithmetic Algorithms*. Natick, MA: A.K. Peters, 2002.
- [22] M. Ercegovac and T. Lang, “Fast multiplication without carry-propagate addition,” *IEEE Trans. Comput.*, vol. 39, no. 11, pp. 1385–1390, Nov. 1990.
- [23] N. Kim, T. Austin, D. Baauw, T. Mudge, K. Flautner, J. Hu, M. Irwin, M. Kandemir, and V. Narayanan, “Leakage current: Moore’s law meets static power,” *Computer*, vol. 36, no. 12, pp. 68–75, Dec. 2003.



Dong-U Lee (M’05) received the B.Eng. degree in information systems engineering and the Ph.D. degree in computing from Imperial College London, London, U.K., in 2001 and 2004, respectively.

From 2005 to 2007, he was a Postdoctoral Researcher at the Department of Electrical Engineering, University of California at Los Angeles, Los Angeles, where he developed high-performance hardware designs for wireless communications and mathematical function evaluations. He is currently a Manager with Mojix Inc., Los Angeles, CA, where

he is focusing on implementation aspects of radio-frequency identification readers. His research interests include computer arithmetic, communications, design automation, reconfigurable computing, and video image processing.



Lok-Won Kim (S'09) received the M.S. degree in electrical engineering from Korea University, Seoul, Korea, in 2003. He is currently working toward the Ph.D. degree in the Department of Electrical Engineering, University of California at Los Angeles, Los Angeles.

His research interests include optimized hardware design methodologies and automation, as well as reliable, secure, and trustworthy system-on-a-chip architectures.



John D. Villasenor (SM'98) received the B.S. degree from the University of Virginia, Charlottesville, in 1985 and the M.S. and Ph.D. degrees from Stanford University, Stanford, CA, in 1986 and 1989, respectively, all in electrical engineering.

He is a Professor with the Department of Electrical Engineering, University of California at Los Angeles, Los Angeles, and a Nonresident Senior Fellow in the Governance Studies Program and the Center for Technology Innovation at the Brookings Institution. His current research interests are technologies and policy implications associated with how information is acquired,

processed, stored, and delivered.