Prediction-based Incremental Refinement For Binomially-factorized Discrete Wavelet Transforms

Yiannis Andreopoulos, Dai Jiang and Andreas Demosthenous

ABSTRACT

It was proposed recently that quantized representations of the input source (e.g. images, video) can be used for the computation of the two-dimensional discrete wavelet transform (2D DWT) *incrementally*. The coarsely-quantized input source is used for the initial computation of the forward or inverse DWT and the result is successively refined with each new refinement of the source description via an embedded quantizer. This computation is based on the direct two-dimensional factorization of the DWT using the generalized spatial combinative lifting algorithm. In this correspondence we investigate the use of prediction for the computation of the results, i.e. exploiting the correlation of neighboring input samples (or transform coefficients) in order to reduce the dynamic range of the required computations, and thereby reduce the circuit activity required for the arithmetic operations of the forward or inverse transform. We focus on binomial factorizations of DWTs that include (amongst others) the popular 9/7 filter-pair. Based on an FPGA arithmetic co-processor testbed, we present energy-consumption results for the arithmetic operations of incremental refinement and prediction-based incremental refinement in comparison to the conventional (non-refinable) computation. Our tests with combinations of intra and error frames of video sequences show that the former can be 70% more energy efficient than the latter for computing to half precision and remains 15% more efficient for full-precision computation.

Index Terms— Approximate Signal Processing, Discrete Wavelet Transform, Energy Consumption, Incremental Refinement of Computation, Lifting Scheme **EDICS:DSP-WAVL, MDS-ALGO**

I. INTRODUCTION

The two-dimensional discrete wavelet transform (2D DWT) has been established as one of the main tools for image compression [1], image denoising and other popular image processing operations [2]. In the vast majority of applications, the transform coefficients are produced to the maximum degree of precision and then they are quantized and processed as appropriate [1]. However, it has been recognized that this wastes system resources for the cases where severe quantization would render the majority of the coefficients not being used at all, or used at

The authors are with the University College London, Dept. of Electronic and Electrical Engineering, Torrington Place, WC1E 7JE, London, UK; Tel: +44-20-76797303; fax: +44-20-73889325; email: iandreop@ee.ucl.ac.uk (Y. Andreopoulos), djiang@ee.ucl.ac.uk (Dai Jiang), a.demosthenous@ee.ucl.ac.uk (A. Demosthenous). This work was supported by the EPSRC, grant EP/F020015/1.

very low precision [3]. For example, this is commonly the case for low-bitrate image and video coding applications [3] and resource-constrained image and video processing operations [4]. For this reason, previous work proposed schemes for approximate computation of transforms and signal processing operations [3]. A property that has been recognized to be of great importance is incremental refinement of computation [4]-[6], where the transform representation of a signal (image, video) is produced incrementally with the use of embedded (bitplane-based) quantization. In our recent work [4], this design has been theoretically analyzed both for the forward and the inverse two-dimensional multilevel DWT using the generalization of the spatial combinative lifting algorithm (SCLA) of Meng and Wang [7]. The overall framework is depicted in Figure 1. There, the multilevel DWT decomposition of the input source (video frame) occurs independently for each quantization threshold (bitplane), starting from the most-significant bitplane (MSB) and going down to the least-significant bitplane (LSB). The results are accumulated after each multilevel SCLA computation to form an incrementally-refined output. Similarly, for the DWT reconstruction, the MSBs of the transform-coefficients are inserted first and the multilevel inverse DWT reconstructs the image incrementally. Each additional processing step requires additional energy consumption. If the processing resources are terminated, one receives the decomposed or reconstructed image with the best-possible quality (controlled by the number of bitplanes processed).

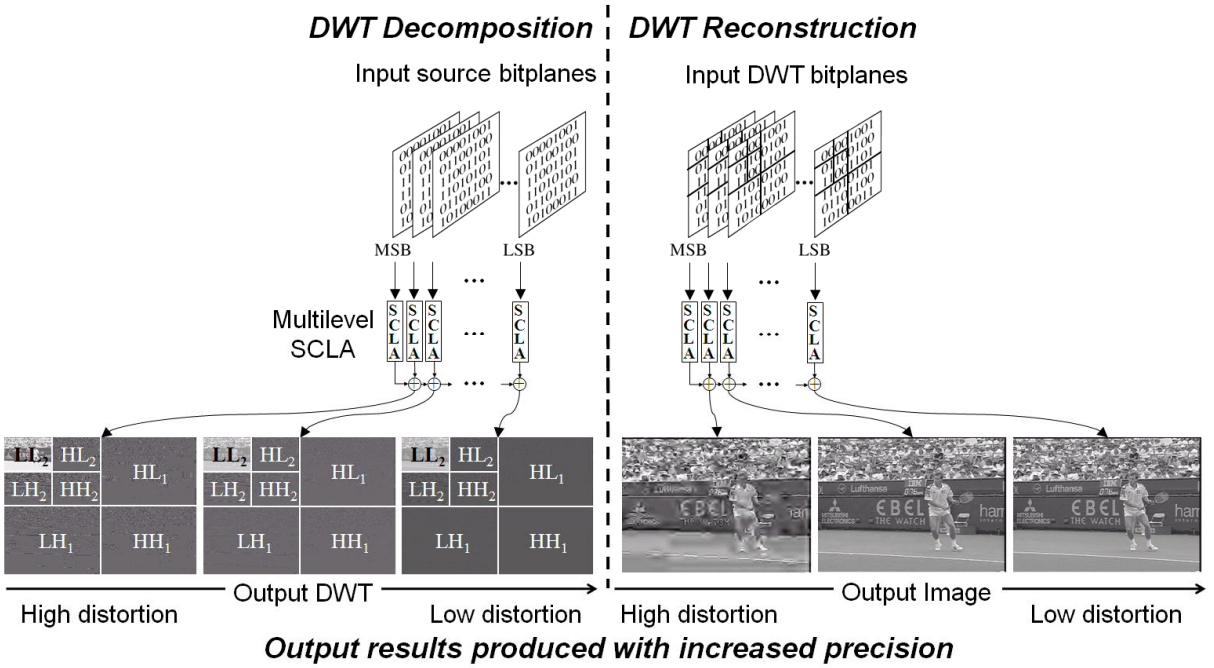

Figure 1. Operational refinement of computation for the multilevel DWT decomposition and reconstruction. The SCLA calculation per bitplane performs two decomposition (or reconstruction) levels and the results are accumulated at the end before moving to the next bitplane.

Although the framework of Figure 1 receives individual bits (per pixel or per wavelet coefficient), the dynamic range of computations performed is increasing according to: *(i)* the lifting coefficients of each lifting step; *(ii)* the

input-source statistics; *(iii)* the number of decomposition levels. In order to adapt the circuit activity according to varying input statistics, it is crucial to have arithmetic units that perform variable dynamic-range computation. Xanthopoulos [8] proposed a suitable framework for this purpose: for all arithmetic units, very low-cost "MSB-rejection" circuits are utilized, which identify the exact number of active bits within each element (adder or multiplier). In this paper, we use a "zero-detection" circuit to avoid performing parts of multiplications with zero inputs and demonstrate its effectiveness in conjunction with incremental computation on an FPGA arithmetic co-processor testbed.

The contribution of this paper is twofold: firstly, we propose incremental computation of the DWT with the use of prediction within each refinement layer (bitplane) of the input (Section II and III); in addition, via the utilized FPGA co-processor (introduced Section IV), we demonstrate the energy-distortion scalability offered by incremental computation and the proposed prediction-based incremental computation in comparison to the conventional (non-refinable) computation (Section V). Our results are relevant to DWT architectures localizing memory accesses to on-chip memory [9] [10], or to cases when the entire image can be stored on-chip, since energy consumption stems predominantly from arithmetic operations and not memory accesses in such cases [9] [10].

II. Overview of SCLA-based DWT under Incremental Refinement of Computation

The 2D DWT of an $R \times C$ input matrix $\mathbf{X}$ consisting of image intensity values is expressed in the spatial domain by[1] $\mathbf{S} = \mathbf{E} \cdot \mathbf{X} \cdot \mathbf{E}^T$, where $\mathbf{E}$ is the analysis polyphase matrix consisting of alternating rows of low- and high-pass filters shifted by two (in order to apply the DWT downsampling), and $\mathbf{S}$ the 2D matrix of output wavelet coefficients. The Z-domain expression of the analysis matrix can be factored into K lifting steps [11]:

$$\mathbf{E}(z) = \prod_{k=1}^{K} \mathbf{U}_k(z)\mathbf{P}_k(z) = \prod_{k=1}^{K} \begin{bmatrix} 1 & 0 \\ a_{\mathrm{u}(k)}^{\mathrm{FXP}}(z+1) & 1 \end{bmatrix} \begin{bmatrix} 1 & a_{\mathrm{p}(k)}^{\mathrm{FXP}}(1+z^{-1}) \\ 0 & 1 \end{bmatrix}, \tag{1}$$

where $\mathbf{P}_k(z)$ and $\mathbf{U}_k(z)$ are the kth prediction and update lifting matrices. The lower- or upper-triangular 2×2 matrices of (1) contain symmetric Laurent binomials [11] [12], with $a_{\mathrm{u}(k)}^{\mathrm{FXP}}$ the fixed-point representation of the kth update filter coefficient (likewise for the predict step). Although the binomial lifting factorizations represented by (1) do not cover all possible factorizations, they do cover some of the most-popular ones found in practical applications [1] [11] [7] [12]. As an example, for the 9/7 filter-pair we have $K = 2$, i.e. two prediction and two update filters, each of which has two identical non-zero and non-unity taps, as shown in (1). Other popular filter-pairs also obey this rule, e.g. cubic B-spline filters [11], the 5/3 filter-pair [1] and the 7/5 filter-pair [14]. In

[1] We are not concerned with the scaling performed after the lifting analysis and before the lifting synthesis [11] because all scaling factors can be incorporated into the subsequent encoding or processing stage [12]. In addition, for notational simplicity, we assume that the image dimensions are integer multiples of $2^{L_{\max}}$, with $L_{\max}$ the number of wavelet decomposition levels.

addition, all symmetric or antisymmetric factorizations can be expressed as binomial lifting factorizations [12].

A. Overview of Incremental SCLA Computation

A reformulation of the 2D lifting scheme has been proposed in the spatial domain by Meng and Wang [7]:

$$\mathbf{S} = \mathbf{U}_K \left(\mathbf{P}_K \left(\cdots \left(\mathbf{U}_1 \left(\mathbf{P}_1 \cdot \mathbf{X} \cdot \mathbf{P}_1{}^T \right) \mathbf{U}_1{}^T \right) \cdots \right) \mathbf{P}_K{}^T \right) \mathbf{U}_K{}^T , \tag{2}$$

This equation computes the 2D DWT in a series of steps, starting from the inner part, i.e. $\mathbf{M}_1^{\mathrm{p}} = \mathbf{P}_1 \cdot \mathbf{X} \cdot \mathbf{P}_1{}^T$, and working outwards toward the final result $\mathbf{S}$. The computation is performed in squares of 2×2 input samples (2D polyphase components), as it will be explained in the following.

If we use double-deadzone embedded quantization of the input $\mathbf{X}$ with basic partition cell $\Delta = 1$ [1], each quantized coefficient $x^{\mathrm{quant}}[r,c]$ of input $\mathbf{X}$ ($0 \le r < R$, $0 \le c < C$) is expressed as:

$$x^{\mathrm{quant}}[r,c] = (-1)^{x^N[r,c]} \sum_{n=0}^{N-1} 2^n x^n[r,c] \tag{3}$$

with $x^n[r,c]$ the n th bit of quantized coefficient $x^{\mathrm{quant}}[r,c]$ (where $x^0[r,c]$ is the least-significant bit), and $x^N[r,c]$ is the sign bit. This is the popular case of successive approximation quantization (SAQ) [1], where, starting from the sign bit, each additional bitplane corresponds to increased precision in the approximation of $x[r,c]$. In the expressions that follow, we use the signed bits of the input, i.e. $\mathrm{x}^n[r,c] = (-1)^{x^N[r,c]} \cdot 2^n x^n[r,c]$ denotes the n th *signed bit* of quantized coefficient $x^{\mathrm{quant}}[r,c]$.

For incremental refinement of the computation of a 2D binomial factorization given by (2) under the SAQ approximation of each input sample $x[r,c]$ shown in (3), the computation of the first step, $\mathbf{M}_1^{\mathrm{p}} = \mathbf{P}_1 \cdot \mathbf{X} \cdot \mathbf{P}_1{}^T$, for each bitplane n, $0 \le n < N$, is [4] ($0 \le i < R/2$, $0 \le j < C/2$):

$$p_1^n[2i,2j] = \mathrm{x}^n[2i,2j] \tag{4}$$

$$p_1^n[2i,2j+1] = \mathrm{x}^n[2i,2j+1] + a_{\mathrm{p}(1)}^{\mathrm{FXP}} \left(\mathrm{x}^n[2i,2j] + \mathrm{x}^n[2i,2j+2) \right] \tag{5}$$

$$p_1^n[2i+1,2j+1] = \mathrm{x}^n[2i+1,2j+1] + a_{\mathrm{p}(1)}^{\mathrm{FXP}} \left(\mathrm{x}^n[2i+1,2j] + \mathrm{x}^n[2i+1,2j+2] + p_1^n[2i,2j+1] + p_1^n[2i+2,2j+1] \right) \tag{6}$$

$$p_1^n[2i+1,2j] = \mathrm{x}^n[2i+1,2j] + a_{\mathrm{p}(1)}^{\mathrm{FXP}} \left(\mathrm{x}^n[2i,2j] + \mathrm{x}^n[2i+2,2j] \right) \tag{7}$$

where $p_1^n\left[2i + \{0,1\}, 2j + \{0,1\}\right]$ is the output quadrant of coefficients, as computed from the signed quantized values (signed bitplanes) $\mathrm{x}^n[r,c]$ of the input. Equation (4) is a simple copy operation between input and output, while (5)-(7) add to the input bit of each case a factor that depends on the n th signed bit values of the samples in the spatial neighbourhood of $\mathrm{x}^n\left[2i + \{0,1\}, 2j + \{0,1\}\right]$ and the lifting-filter coefficient $a_{\mathrm{p}(1)}^{\mathrm{FXP}}$.

The second matrix product, $\mathbf{M}_1^{\mathrm{u}} = \mathbf{U}_1 \cdot \mathbf{M}_1^{\mathrm{p}} \cdot \mathbf{U}_1{}^T$, is produced by reusing the results of (4)-(7):

$$u_1^n\left[2i+1,2j+1\right] = p_1^n\left[2i+1,2j+1\right] \tag{8}$$

$$u_1^n\left[2i+1,2j\right] = p_1^n\left[2i+1,2j\right] + a_{\mathrm{u}(1)}^{\mathrm{FXP}} \left(p_1^n[2i+1,2j-1] + p_1^n[2i+1,2j+1] \right) \tag{9}$$

$$u_1^n\left[2i,2j\right] = p_1^n\left[2i,2j\right] + a_{\mathrm{u}(1)}^{\mathrm{FXP}}\left(p_1^n[2i,2j-1] + p_1^n[2i,2j+1] + u_1^n[2i-1,2j] + u_1^n[2i+1,2j]\right) \tag{10}$$

$$u_1^n\left[2i,2j+1\right] = p_1^n\left[2i,2j+1\right] + a_{\mathrm{u}(1)}^{\mathrm{FXP}}\left(p_1^n[2i-1,2j+1] + p_1^n[2i+1,2j+1]\right) \tag{11}$$

where $u_1^n\left[2i+\{0,1\},2j+\{0,1\}\right]$ is the output quadrant of coefficients, as computed for the output coefficients $p_1^n[r,c]$ of (4)-(7). Again, (8) is a simple copy operation, while, for (9)-(11), the fixed-point lifting coefficient $a_{\mathrm{u}(1)}^{\mathrm{FXP}}$ is used to update the input. The remaining steps $k = 2,\ldots,K$ to complete the single-level decomposition using the 2D lifting formulation of (2) are performed as in (4)-(7) and (8)-(11) with the replacement of the input with the output of each previous step and the replacement of the coefficients $a_{\mathrm{p}(1)}^{\mathrm{FXP}}$ and $a_{\mathrm{u}(1)}^{\mathrm{FXP}}$ by $a_{\mathrm{p}(k)}^{\mathrm{FXP}}$ and $a_{\mathrm{u}(k)}^{\mathrm{FXP}}$, respectively. All steps can be performed in-place as in the conventional lifting decomposition, with the reuse of the memory for the $\mathbf{M}_1^{\mathrm{p}}$ and $\mathbf{M}_1^{\mathrm{u}}$ arrays. Once all steps are completed, we perform the reordering and addition to the previous results for the high-frequency coefficients by:

$$\begin{aligned} \mathrm{HL}_1\left[i,j\right] &\leftarrow \mathrm{HL}_1\left[i,j\right] + u_K^n[2i,2j+1], \\ \mathrm{LH}_1\left[i,j\right] \leftarrow \mathrm{LH}_1\left[i,j\right] + u_K^n[2i+1,2j], \quad & \mathrm{HH}_1\left[i,j\right] \leftarrow \mathrm{HH}_1\left[i,j\right] + u_K^n[2i+1,2j+1], \end{aligned} \tag{12}$$

where HL_1, LH_1 and HH_1 are the three high-frequency subbands of level one (see Figure 1) and $a \leftarrow b$ assigns b to a. Notice that the low-frequency coefficients ($u_K^n[2i,2j]$) are not involved in this process, as they are the input for the subsequent decomposition level, as described in the following.

B. Multilevel Extension

The multilevel extension of the bitwise computation of the DWT for $L_{\max}$ levels was performed in our previous work [4] via a "frequency-first" incremental refinement of computation: for each input bitplane n, the lifting-scheme computation is continued for all subsequent levels after the first level and the results are accumulated at the end, before moving to the next input bitplane. This is achieved by reformulating the first prediction step of (4)-(7) for all levels l, $2 \le l \le L_{\max}$ by ($0 \le i < R/2^l$, $0 \le j < C/2^l$):

$$p_1^n\left[2i,2j\right] = u_K^n\left[4i,4j\right] \tag{13}$$

$$p_1^n\left[2i,2j+1\right] = u_K^n\left[4i,4j+2\right] + a_{\mathrm{p}(1)}^{\mathrm{FXP}}\left(u_K^n[4i,4j] + u_K^n[4i,4j+4]\right) \tag{14}$$

$$p_1^n\left[2i+1,2j+1\right] = u_K^n\left[4i+2,4j+2\right] + a_{\mathrm{p}(1)}^{\mathrm{FXP}}\left(u_K^n[4i+2,4j] + u_K^n[4i+2,4j+4] + p_1^n[2i,2j+1] + p_1^n[2i+2,2j+1]\right) \tag{15}$$

$$p_1^n\left[2i+1,2j\right] = u_K^n\left[4i+2,4j\right] + a_{\mathrm{p}(1)}^{\mathrm{FXP}}\left(u_K^n[4i,4j] + u_K^n[4i+4,4j]\right) \tag{16}$$

where we use the low-frequency outputs $u_K^n\left[4i+\{0,2\},4j+\{0,2\}\right]$ of the last update step of the previous level. The subsequent update step and all additional pairs of lifting steps, as well as the incrementation of all high-frequency subbands, $\mathrm{HL}_l, \mathrm{LH}_l, \mathrm{HH}_l$, are performed as shown previously, with: $0 \le i < R/2^l$, $0 \le j < C/2^l$. The low-frequency subband coefficients produced at the last decomposition level are incremented by:

$$\mathrm{LL}_{L_{\max}}\left[i,j\right] \leftarrow \mathrm{LL}_{L_{\max}}\left[i,j\right] + u_K^n[2i,2j] \tag{17}$$

This process is then carried out for the next bitplanes $n-1,\ldots,0$. Source code for the entire process is available [13].

C. Inverse SCLA and Inverse DWT

Concerning the inverse SCLA, the process is exactly anti-symmetric, as in the conventional lifting computation: all lifting steps are performed in reverse order by solving the forward bitwise lifting equations for the coefficients being predicted or updated during the forward transform. More explicitly, (8)-(11) are inverted by solving for $p_1^n[2i+\{0,1\},2j+\{0,1\}]$ and replacing the terms $p_1^n[\bullet,\bullet]$ inside the parentheses of (9) and (11) by $u_1^n[\bullet,\bullet]$. These equations would then perform the inverse update step and derive $p_1^n[2i+\{0,1\},2j+\{0,1\}]$ from inputs $u_1^n[2i+\{0,1\},2j+\{0,1\}]$. Inversion of (4)-(7) occurs as for (8)-(11), with $\mathrm{x}^n[\bullet,\bullet]$ inside the parentheses of (5) and (7) replaced by $p_1^n[\bullet,\bullet]$. The inversion equations are executed in reverse order; first the inverted (11), followed by the inverted (10), up to the inverted (4).

For the K th pair of lifting steps, the inputs $u_K^n[2i+\{0,1\},2j+\{0,1\}]$ consist of the n th signed bitplane of the input wavelet coefficients $\mathrm{LL}_{L_{\max}}[i,j],\mathrm{HL}_l[i,j],\mathrm{LH}_l[i,j],\mathrm{HH}_l[i,j]$, $l=L_{\max},L_{\max}-1,\ldots,1$; they are linked via (12) and (17) (if $l=L_{\max}$). Hence, in the inverse transform, the signed bits of the input low and high-frequency wavelet coefficients of all decomposition levels are used, starting from the coarsest level, inverting all lifting steps, and increasing the resolution up to the pixel domain. All levels are inverted within each increment layer n, and the final reconstructed increments for all pixels are added to the output pixels (see the right side of Figure 1) in order to progressively reduce the distortion of the reconstructed image. This process is then carried out for the next bitplanes $n-1,\ldots,0$. Source code for the inverse transform is available online [13].

III. Predictive Incremental Refinement of Computation for the DWT

Since neighboring input pixels or transform coefficients are expected to be correlated[2], we can attempt to reduce the required computations for the incremental computation algorithm presented previously by introducing a prediction scheme between the pixels or coefficients of the input. For each of the 2×2 input 2D polyphase components of each lifting step, we form the prediction in the direction of the 2D lifting-based filtering presented in the previous section.

A. Proposed Formulation

The first component of the first step [shown in (4)] remains unchanged as this is simply an assignment operation. The second component applies the lifting filter along the j-axis. By writing (5) with the replacement of j by $j-1$ and subtracting the resulting expression from (5) we get the increment of computation for $p_1^n[2i,2j+1]$ if we have previously computed $p_1^n[2i,2j-1]$. This is given by:

[2] The DWT should decorrelate the input signal. However, in practice, neighboring subband coefficients remain correlated. Image compression and denoising applications use this by forming context models based on each coefficient's neighborhood.

$$p_1^n[2i,2j+1] = p_1^n[2i,2j-1] + \left(\mathrm{x}^n[2i,2j+1] - \mathrm{x}^n[2i,2j-1]\right) + a_{\mathrm{p}(1)}^{\mathrm{FXP}}\left(\mathrm{x}^n[2i,2j+2] - \mathrm{x}^n[2i,2j-2]\right) \quad (18)$$

Notice that (18) assumes that the calculation is applied "rows-first", i.e. for each position $j = 1,\ldots,C/2-1$ within each row $i = 0,1,\ldots,R/2-1$. Furthermore, if $\mathrm{x}^n[2i,2j-2] = \mathrm{x}^n[2i,2j+2]$, we avoid the multiply-accumulate step with factor $a_{\mathrm{p}(1)}^{\mathrm{FXP}}$ in (18), which is performed by the original incremental lifting of (5). Hence, with a multiplier design that is dynamically adjustable to the input sparsity, we expect to have decreased circuit activity for the multiplication operations.

The third polyphase component [shown in (6)] applies diagonal filtering along both the i and the j-axis by reusing the results of the second component computed previously. For this reason, we break the computation in two parts, with the first part performing prediction across the j-axis and the second performing prediction across the i-axis and incrementing the result of the previous computation. The first part is:

$$\begin{aligned} p_{1,\mathrm{part}}^n[2i+1,2j+1] = p_{1,\mathrm{part}}^n[2i+1,2j-1] + \left(\mathrm{x}^n[2i+1,2j+1] - \mathrm{x}^n[2i+1,2j-1]\right) \\ + a_{\mathrm{p}(1)}^{\mathrm{FXP}}\left(\mathrm{x}^n[2i+1,2j+2] - \mathrm{x}^n[2i+1,2j-2]\right) \end{aligned} \quad (19)$$

and it is applied for each position $j = 1,\ldots,C/2-1$ within each row $i = 0,1,\ldots,R/2-1$. The second part is:

$$p_1^n[2i+1,2j+1] = p_1^n[2i-1,2j+1] + a_{\mathrm{p}(1)}^{\mathrm{FXP}}\left(p_{1,\mathrm{part}}^n[2i+2,2j+1] - p_{1,\mathrm{part}}^n[2i-2,2j+1]\right) \quad (20)$$

and it is applied for each $i = 1,\ldots,R/2-1$ within each column $j = 0,1,\ldots,C/2-1$. Similarly as before, if $\mathrm{x}^n[2i+1,2j+2] = \mathrm{x}^n[2i+1,2j-2]$ in (19), or $p_{1,\mathrm{part}}^n[2i+2,2j+1] \simeq p_{1,\mathrm{part}}^n[2i-2,2j+1]$ in (20), we reduce the circuit activity required for the costly multiply-accumulate operations performed by the incremental lifting of (6). However, since (19) and (20) involve two multiplications, this may not be cost-effective in comparison to (6). Hence, we can selectively disable these operations and perform (6) instead if this is deemed to be more efficient.

Finally, the last component of the quadrant of computation of the first lifting step [shown in (7)] is written in the same manner as the second component, albeit applying the prediction in the i-axis:

$$p_1^n[2i+1,2j] = p_1^n[2i-1,2j] + \left(\mathrm{x}^n[2i+1,2j] - \mathrm{x}^n[2i-1,2j]\right) + a_{\mathrm{p}(1)}^{\mathrm{FXP}}\left(\mathrm{x}^n[2i+2,2j] - \mathrm{x}^n[2i-2,2j]\right) \quad (21)$$

and it is applied for each $i = 1,\ldots,R/2-1$ within each column $j = 0,1,\ldots,C/2-1$.

The second lifting step [update step of (8)-(11)] is modified in the same manner by applying the prediction as shown in (18)-(21). The expressions can be derived following the same rules and are omitted for brevity of description. Subsequent pairs of lifting steps follow the same rules. One final aspect relates to the border treatment of the prediction-based incremental refinement of the DWT computation. This is performed by applying the original equations for incremental refinement for the initial row ($i = 0$) or the initial column ($j = 0$) of each case.

The modification for the calculation performed for the subsequent levels [(14)-(16)] is performed as follows [we omit (13) since it is just an assignment operation]. For (14), for each $1 \le j < C/2^l$ of each row $0 \le i < R/2^l$:

$$p_1^n\left[2i,2j+1\right]=p_1^n\left[2i,2j-1\right]+\left(u_K^n\left[4i,4j+2\right]-u_K^n\left[4i,4j-2\right]\right)+a_{\mathrm{p}(1)}^{\mathrm{FXP}}\left(u_K^n[4i,4j+4]-u_K^n[4i,4j-4]\right). \quad (22)$$

For (15), for each $1\le j<C/2^l$ of each row $0\le i<R/2^l$, we first calculate:

$$\begin{aligned}p_{1,\mathrm{part}}^n\left[2i+1,2j+1\right]=p_{1,\mathrm{part}}^n\left[2i+1,2j-1\right]+\left(u_K^n\left[4i+2,4j+2\right]-u_K^n\left[4i+2,4j-2\right]\right)\\ +a_{\mathrm{p}(1)}^{\mathrm{FXP}}\left(u_K^n[4i+2,4j+4]-u_K^n[4i+2,4j-4]\right)\end{aligned} \quad (23)$$

and then, for each $1\le i<R/2^l$ of each column $0\le j<C/2^l$ we calculate $p_1^n\left[2i+1,2j+1\right]$ with (20). Similarly as before, if (23) and (20) are deemed to be inefficient, we can apply (15) instead.

Finally, for (16), for each $1\le j<C/2^l$ of each row $0\le i<R/2^l$ we have:

$$p_1^n\left[2i+1,2j\right]=p_1^n\left[2i-1,2j\right]+\left(u_K^n\left[4i+2,4j\right]-u_K^n\left[4i-2,4j\right]\right)+a_{\mathrm{p}(1)}^{\mathrm{FXP}}\left(u_K^n[4i+4,4j]-u_K^n[4i-4,4j]\right). \quad (24)$$

The calculations at the borders are performed by applying the original equations for incremental refinement [(13)-(16)] for the initial row ($i=0$) or the initial column ($j=0$) of each case. Indicative source code for the proposed approach (forward and inverse) is available online [13].

B. Discussion

The proposed prediction-based incremental refinement of computation is applied per input bitplane. As such, it can be selectively applied to some input bitplanes, e.g. the four most-significant ones, and the original equations (4)-(11) can be applied for the remaining bitplanes. In addition, it can be applied selectively on some parts of the lifting steps, e.g. we can apply (6) instead of the proposed (19) and (20), or even on selected input frames where the correlation between neighboring input samples is stronger (this is actually used in the experimental results of Section V). This creates a "hybrid" approach that uses prediction only on some parts of the computation. This can be beneficial because, for example, correlation between neighbouring inputs is expected to decrease when moving to lower bitplanes, since this corresponds to the low-amplitude parts of the input image, which are expected to contain fine-grain noise. Similarly, decreased correlation between neighboring samples is observed when dealing with error frames instead of video frames, since error frames are the result of temporal prediction.

Even though the description of the paper focused on single-bitplane inputs, a number of bitplanes can be inserted together and the computation can utilize all of them together as one increment layer. For example, for an 8-bit input image, the three MSBs can be processed first, followed by the three intermediate bitplanes and the two LSBs. This creates only three increment layers of computation. This can reduce the expected overhead when increasing the number of increment layers. In our recent work on software designs for incremental image processing [6], we found that three or four increment layers provide for sufficient quality-complexity scalability without significant overhead in the overall execution time. From the implementation perspective, a crucial element needed in order to take advantage of the reduction of the multiplication bitwidth is a multiplier unit with adjustable circuit activity according to the input bit patterns. This is the topic of the following section.

IV. FGPA-BASED ARITHMETIC CO-PROCESSOR TESTBED FOR ENERGY MEASUREMENTS

In order to substantiate the potential benefits of the proposed computation in a real environment, we setup an experimental testbed consisting of a main platform (personal computer) realizing the transform decomposition or reconstruction algorithmic flow in software. All arithmetic operations (multiplications, additions) are transferred to a Xilinx Virtex II XC2V1500 FPGA and are computed based on customized designs of multipliers and adders placed on the FPGA. We then measure and report the energy consumption of these operations for the different algorithms (conventional, incremental refinement and prediction-based incremental refinement). We used the ISE 8.2 FPGA development environment with the XPower 8.2 tool [15] for this purpose and all our results relate to the dynamic energy dissipation reported by the tool.

All arithmetic operations are performed in signed fixed-point representation and require (maximally) a 12-bit integer part plus a 12-bit fractional part, which ensures reconstruction peak-signal-to-noise (PSNR) values above 55dB for up to 6 levels of decomposition [16]. All our arithmetic units restrict their output to 24 bits, which is comparable or less than related designs [16] [17]. The multiplier design separates the integer and fractional parts into three 4-bit parts (IA3~IA1 and FA1~FA3) and then performs a cascade realization of the multiplication, whose intermediate operations are depicted in Figure 2(a). Since the result will always be contained within the 12 plus 12 bit output, only the intermediate operations contributing to the retained part of the output of Figure 2(a) are performed. Furthermore, each individual 4-bit multiplication of Figure 2(a) is enabled based on a zero-detection circuit. This circuit checks all inputs A[0:3] and B[0:3] of the 4-bit multiplier and raises the CE (circuit enabling) signal only when the multiplication will have non-zero output, as shown by Figure 2(b); otherwise the default zero result is obtained without activating the multiplier.

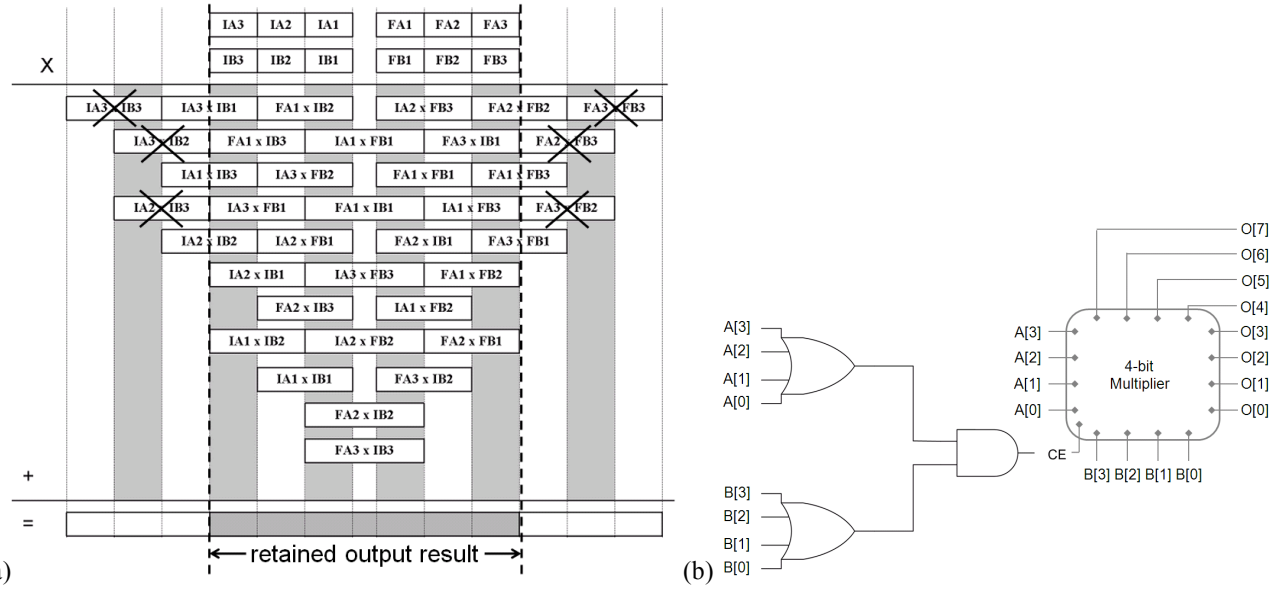

Figure 2: (a) Operations of the 24-bit multiplier. (b) Zero-detection circuit for the activation of each 4-bit multiplier.

The use of the zero-detection circuit ensures that the overall multiplier design is adjusted to: *(i)* each lifting coefficient's active bitwidth without customizing the multiplier design to a particular lifting scheme; *(ii)* the input's varying statistics and varying dynamic range; *(iii)* the output's range (24 bits). This design was found to consume less energy than the standard 24-bit cascade multiplier with the same throughput and latency.

Concerning the adder's design, even though we designed a bitwidth-adaptive adder unit, our measurements indicated that the energy consumption of this design is comparable to that of a standard 24-bit adder, and overall more than an order of magnitude smaller than the average energy consumption of the adaptive multiplier design. Hence, we resorted to using a standard 24-bit adder for all additions performed.

Both the multiplier and adder designs operate in one clock cycle. Representative results for the dynamic energy consumption measured for each combination of inputs when clocked at 75MHz, as well as the layout of the multiplier design are provided online [13].

V. Experimental Results

We examine the energy consumption of the proposed approach for individual cases of forward and inverse DWT applied to YUV intra frames as well as to YUV error frames produced by motion-compensated prediction. Two schemes are used for comparison purposes: conventional (non-refinable) computation and incremental refinement of DWT computation as proposed in our previous work [4]. All 300 frames of the CIF sequences *Stefan*, *Coastguard* and *Foreman* were used for the experiments of this section. Image distortion is measured using the PSNR across luminance (Y) and chrominance (U and V) channels of each video frame: $\text{mean\_PSNR}=\frac{4\text{PSNR(Y)}+\text{PSNR(U)}+\text{PSNR(V)}}{6}$ in order to produce one metric including all channels. The PSNR of each channel $\text{C}\in\{\text{Y},\text{U},\text{V}\}$ is measured as: $\text{PSNR(C)}=10\log_{10}\frac{255^2}{\text{MSE(C)}}$, where MSE(C) is the mean squared error. For the forward DWT, PSNR is measured by inverting the produced decomposition via a software implementation of the inverse DWT with double-precision floating-point representation [13].

Energy consumption is measured via the testbed presented in Section IV, which includes one adder and one multiplier clocked at 35MHz for the conventional approach and at 75MHz for the incremental approaches. The clock frequencies were chosen so as to allow for processing of 25 frames/sec when using all input bitplanes: since the incremental approaches are performing more arithmetic operations (multiple increments but with smaller bitwidth), we increased the clock frequency for these cases in order to ensure they complete the frame processing at the same time with the conventional approach. The 9/7 [11] and 7/5 [14] filter-pairs were chosen for our comparisons since they present state-of-the-art compression performance with moderate complexity. The lifting coefficients of the 9/7 and 7/5 filter-pairs can be found in [11] and [14], respectively. The conventional approach is using all input bitplanes simultaneously, while the incremental approaches are breaking the computation into two

layers: for the forward DWT, the four MSBs are inserted first, followed by the 4 LSBs; for the inverse DWT, the 8 MSBs are inserted first, followed by the 4 LSBs. This creates two layers of computation labeled as "half-precision" and "full-precision" results. All utilized error frames in our experiments were produced by a spatial-domain motion-compensated prediction scheme [18] using successive (frame-by-frame) prediction and within a group-of-pictures (GOP) of 16 frames. Four wavelet decomposition levels were performed. No wavelet-coefficient coding was performed in order to avoid artificial thresholding of wavelet coefficients before the inverse DWT.

The results are reported in Table 1. As seen from the top half of the table, when used for intra frames, the proposed approach provides benefits in comparison to the original algorithm for incremental refinement of computation [4] by decreasing its energy consumption and making it comparable to conventional computation. In addition, the proposed scheme brings significant benefits in terms of energy reduction for half-precision computation. However, when used for error frames, the proposed approach does not provide any improvement in comparison to incremental refinement [4]; instead, energy consumption is increased, sometimes by a significant amount. This is attributed to the failure of the prediction scheme due to reduced correlation between neighboring coefficients of error frames. These observations hold for both filter-pairs and both forward and inverse DWT.

There are some minor discrepancies in PSNR between the different approaches caused by the fixed-point precision chosen for the example cases presented, in the order of 0.03dB. All visual artifacts observed in the reconstructed images and error frames at each terminating quantization precision (bitplane) are typical of low-bitrate wavelet-based quantization and coding approaches studied in the literature [19].

Frame Type	**Full Precision**				**Half Precision**		
	Conventional	Incremental [4]	Proposed	mean_PSNR	Incremental [4]	Proposed	mean_PSNR
9/7 forward DWT							
Intra	**22.3**	+24.8%	+14.6%	56.7dB	-34.4%	-43.5%	29.5dB
Error	**16.8**	-5.2%	+9.2%	57.7dB	-74.5%	-63.1%	37.2dB
9/7 inverse DWT							
Intra	**21.9**	+14.7%	+10.4%	57.6dB	-39.4%	-45.7%	33.3dB
Error	**16.5**	-37.7%	+10.3%	57.7dB	-82.4%	-52.3%	36.7dB
7/5 forward DWT							
Intra	**17.0**	+25.5%	+16.6%	56.7dB	-34.8%	-43.6%	29.5dB
Error	**13.4**	-08.0%	-04.0%	57.7dB	-77.5%	-68.3%	37.2dB
7/5 inverse DWT							
Intra	**15.9**	+8.7%	+2.7%	57.6dB	-36.2%	-48.1%	33.3dB
Error	**12.9**	-39.8%	-30.1%	57.7dB	-84.4%	-50.7%	36.7dB

GOP with: 1 Intra (I) & 15 Error (P) frames	Conventional	Proposed (I)+Incremental (P)	mean_PSNR	Proposed (I)+Incremental (P)	mean_PSNR
	9/7 forward DWT				
	18.3	-13.3%	57.4dB	-69.4%	33.7dB
	9/7 inverse DWT				
	17.1	-17.1%	57.7dB	-72.8%	35.4dB

Table 1. Energy consumption (in micro-Joule per frame) and PSNR results per intra frame and error frame. The results of incremental refinement [4] and the proposed prediction-based incremental refinement approaches are presented as the percentile difference to the energy consumption reported for the conventional approach.

The bottom half of Table 1 shows the energy consumption in a video coding scenario where 1 intra frame and 15 error frames are transformed, when the proposed approach is used for the intra frames and incremental refinement [4] is used for the error frames. This "hybrid" approach provides for lower energy consumption for full-precision computation in comparison to the conventional (non-refinable) computation. Moreover, in such video applications, energy consumption is reduced significantly (by more than 70% on average) by terminating at half-precision (first increment layer), as shown by the bottom half of Table 1.

VI. Conclusions

We propose a prediction-based method for incremental computation of the forward and inverse discrete wavelet transform (DWT) under a bitplane-based formulation of the 2D lifting decomposition. The proposed approach applies prediction of neighboring pixels or wavelet coefficients in the direction of the lifting-based filtering. This is performed for each of the 2D polyphase components of the direct 2D computation of the multilevel DWT decomposition. Based on FPGA-based comparisons with conventional computation, we verified that the proposed approach and incremental refinement has comparative energy consumption in full-precision computation, with the proposed approach providing an improvement for intra frames. At the same time, these approaches can terminate at intermediate energy-distortion points (e.g. half-precision) with significant decrease in energy consumption.

References

[1] *JPEG2000: Image Compression Fundamentals, Standards and Practice*, D. Taubman, M. Marcellin, Kluwer, 2002.

[2] S. Mallat, *A wavelet tour of signal processing*. Academic Press, San Diego CA, 1998.
V. K. Goyal, and M. Vetterli, "Manipulating rates, complexity and error-resilience with discrete transforms," *Proc. IEEE Asilomar Conf. on Signals, Syst. and Comput.*, vol. 1, pp. 457-461, Nov. 1998.

[3] Z. Wang, "Pruning the fast discrete cosine transform," *IEEE Trans. on Comm.*, vol. 39, no. 5, May 1991.

[4] Y. Andreopoulos and M. van der Schaar, "Incremental refinement of computation for the discrete wavelet transform," *IEEE Trans. on Signal Processing*, vol. 56, no. 1, pp. 140-157, Jan. 2008.

[5] J. Winograd and S. H. Nawab, "Incremental refinement of DFT and STFT approximations," *IEEE Signal Proc. Letters*, vol. 2, no. 2, pp. 25-27, Feb. 1995.

[6] D. Anastasia and Y. Andreopoulos, "Software designs of image processing tasks with incremental refinement of computation," *Proc. IEEE Workshop on Signal Process. Systems* (SIPS), Tampere, Finland, Oct. 2009, pp. 249-254.

[7] H. Meng and Z. Wang, "Fast spatial combinative lifting algorithm of wavelet transform using the 9/7 filter for image block compression," *IEE Electronics Letters*, vol. 36, no. 21, pp. 1766-1767, Oct. 2000.

[8] T. Xanthopoulos, "Low power data-dependent transform video and still image coding," *Ph.D. Thesis, Massachusetts Institute of Technology*, Feb. 1999, [Online]. Available: http://mtlweb.mit.edu/researchgroups/icsystems/theses.html.

[9] P.-C. Tseng, Y.-C. Chang, Y.-W. Huang, H.-C. Fang, C.-T. Huang, and L.-G. Chen, "Advances in hardware architectures for image and video coding – a survey," *Proc. of the IEEE*, vol. 93, no. 1, 184-197, Jan. 2005.

[10] M. Angelopoulou, K. Masselos, P. Y. K. Cheung, and Y. Andreopoulos, "Implementation and comparison of the 5/3 lifting 2-D discrete wavelet transform computation schedules on FPGAs," *VLSI Signal Processing*, vol. 51, no. 1, pp. 3-21, April 2008.

[11] I. Daubechies and W. Sweldens, "Factoring wavelet transforms into lifting steps," *J. Fourier Anal. Appl.*, vol.4, pp.247-269, 1998.

[12] C.-T. Huang, P.-C. Tseng and L.-G. Chen, "Efficient VLSI architectures of lifting-based discrete wavelet transform by systematic design method," *Proc. IEEE Int. Symp. on Circ. and Syst.,* ISCAS, vol. 5, pp. 565-568, 2002.

[13] [Online]. Available: http://www.ee.ucl.ac.uk/~iandreop/ORIP.html

[14] C. Brislawn, "Interim report on (Part 2) core experiment CodEff07, 7-tap/5-tap Filter Bank option," *ISO/IEC JTC1/SC29/WG1*, N1761, JPEG, July 2000.

[15] [Online]. Available: http://www.xilinx.com/itp/xilinx10/help/iseguide/mergedProjects/xpower/html/xp_b_overview.htm

[16] V. Spiliotopoulos, *et al*, "Quantization effect on VLSI implementations for the 9/7 DWT Filters," *Proc. IEEE International Conf. on Acoustics, Speech and Signal Processing*, ICASSP'01, Salt Lake City, UT, vol. 2, pp. 1197-1200, May 2001.

[17] S. L. Bishop, S. Rai, B. Gunturk, J. L. Trahan, and R. Vaidyanathan, "Reconfigurable implementation of wavelet integer lifting transforms for image compression," *Proc. IEEE Internat. Conf. on Reconfig. Comput. and FPGA's*, pp. 1-9, 2006.

[18] Y. Andreopoulos, *et al*, "Scalable wavelet video-coding with in-band prediction - Implementation and experimental results," *Proc. IEEE International Conf. on Image Processing*, ICIP'02, vol. 3, pp. 729-732, Sept. 2002.

[19] A. B. Watson, G. Y. Yang, J. A. Solomon and J. Villasenor, "Visibility of wavelet quantization noise," *IEEE Trans. on Image Process.*, vol. 6, no. 8, pp. 1164-1175, Aug. 1997.