POLITECNICO DI TORINO
Repository ISTITUZIONALE

Result-Biased Distributed-Arithmetic-Based Filter Architectures for Approximately Computing the DWT

(Article begins on next page)

20 April 2024

Result-biased Distributed-Arithmetic-based filter architectures for approximately computing the DWT

Maurizio Martina, *Senior Member IEEE*, Guido Masera, *Senior Member IEEE*,
Massimo Ruo Roch, Gianluca Piccinini

Abstract—**The discrete wavelet transform is a fundamental block in several schemes for image compression. Its implementation relies on filters that usually require multiplications leading to a relevant hardware complexity. Distributed arithmetic is a general and effective technique to implement multiplierless filters and has been exploited in the past to implement the discrete wavelet transform as well. This work proposes a general method to implement a discrete wavelet transform architecture based on distributed arithmetic to produce approximate results. The novelty of the proposed method relies on the use of result-biasing techniques (inspired by the ones used in fixed-width multiplier architectures), which cause a very small loss of quality of the compressed image (average loss of 0.11 dB and 0.20 dB in terms of PSNR for the 9/7 and 10/18 wavelet filters, respectively). Compared with previously proposed distributed-arithmetic-based architectures for the computation of the discrete wavelet transform, this technique saves from about 20% to 25% of hardware complexity.**

Index Terms—**low-complexity, FIR filters, Distributed Arithmetic, JPEG2000, DWT**

I. INTRODUCTION

In the last years the Discrete Wavelet Transform (DWT) has gained a wide diffusion. Thanks to its excellent decorrelation properties the DWT has been included into JPEG2000 [1], the standard recently adopted for Digital Cinema [2]. This has fostered researchers and led to efficient VLSI architectures to implement the DWT, [3], [4]. As shown in [5], the computational kernel of the DWT is a Filter Bank (FB). Thus, several efforts have been spent to obtain multiplierless architectures of the FB structure. As an example in [6], [7] the B-spline factorization [8], [9] is exploited to design multiplierless FB architectures. Recently, other approaches have been proposed as well, e.g. algebraic integer quantization [10], [11], coefficient rationalization [12], polymorphic implementation [13] and half-band polynomial factorization [14].

Unfortunately, the aforementioned techniques require not only to know the values of the filter taps but also the mathematical derivation of the filters or at least some specific factorizations. On the contrary, Distributed Arithmetic (DA) is a systematic methodology to design multiplierless architectures for digital filters. Indeed, it has been recently employed to design low complexity and high throughput architectures for i) Finite-Impulse-Response (FIR) filters [15], [16], ii) Discrete-Cosine-Transform (DCT) based architectures [17], [18], iii) multiplierless FB implementations of the DWT [19] [20].

The authors are with the Electronics and Telecommunications Department - Politecnico di Torino
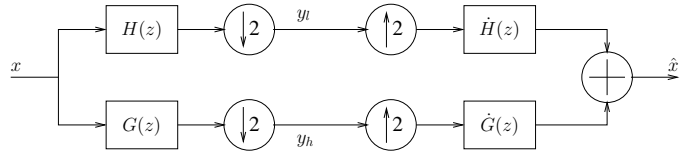
Figure 1: Block diagram of the filter bank scheme.

Inspired by result-biased techniques proposed in [21]–[24] for fixed-width multipliers, this work aims to show that the complexity of DA-based architectures for DWT computation can be further reduced by applying result-biasing techniques. It is relevant to remark that the proposed approach is agnostic, i.e. it can be applied independently of the design criterion adopted for the addressed filters. In particular, in this work we show that i) the complexity of DA-based architectures for wavelet filters can be reduced by about 20% to 25% with a very limited performance degradation (thus result-biasing compensation can be avoided); ii) the implemented DA-based architecture for the 9/7 wavelet filters features almost the same performance and complexity as other multiplierless solutions, which have been optimized by taking advantage of the specific properties of these filters (see [25]). Furthermore, the proposed solution features a large complexity reduction compared to state-of-art architectures when applied to the 10/18 wavelet filters.

The paper is structured as follows. Section II summarizes the general computational scheme of DA-based architectures for wavelet filters and Section III introduces concepts and definitions for implementing result-biasing techniques. In Section IV result-biasing is applied to two important cases of study: the 9/7 and 10/18 wavelet filters. In Section V experimental results and comparisons are shown. Finally, conclusions are drawn in Section VI.

II. DA-BASED FBS FOR DWT COMPUTATION

Let us consider the FB shown in Fig. 1 where $H(z) = \sum_{i=0}^{k-1} h[i]z^{-i}$ and $G(z) = \sum_{i=0}^{l-1} g[i]z^{-i}$ are the low pass and high pass analysis filters with length k and l, respectively, and $\dot{H}(z) = \sum_{i=0}^{\dot{k}-1} \dot{h}[i]z^{-i}$ and $\dot{G}(z) = \sum_{i=0}^{\dot{l}-1} \dot{g}[i]z^{-i}$ the low pass and high pass synthesis ones with length $\dot{k}$ and $\dot{l}$.

A. Analysis filters

The two analysis outputs (y_l and y_h) are obtained as: $y_l[i] = \sum_{j}^{k} h[j]x[i-j]$ and $y_h[i] = \sum_{j}^{l} g[j]x[i-j]$, where $x[i]$ is the input signal. Let us assume that the taps of the filters are
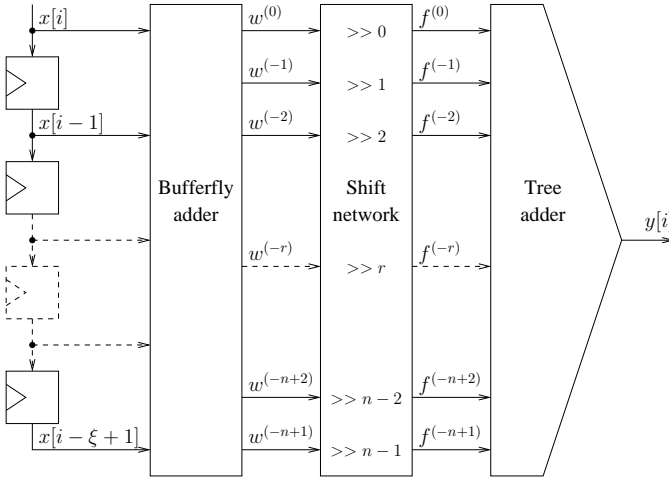
Figure 2: Block scheme of the general DA-based architecture.

amplitude normalized, i.e. $h[j], g[j] \in [-1, 1)$, and represented as 2's complement values using n bits. Then, we can re-write $y_l[i]$ and $y_h[i]$ as:

$$y_l[i] = (h[0] \cdots h[k-1]) \cdot \begin{pmatrix} x[i] \\ \cdots \\ x[i-k+1] \end{pmatrix} \quad (1)$$

$$= \underline{\sigma}_n \cdot \begin{pmatrix} h^{(0)}[0] & \cdots & h^{(0)}[k-1] \\ h^{(-1)}[0] & \cdots & h^{(-1)}[k-1] \\ \vdots & \cdots & \vdots \\ h^{(-n+1)}[0] & \cdots & h^{(-n+1)}[k-1] \end{pmatrix} \cdot \underline{x}_k$$

and

$$y_h[i] = (g[0] \cdots g[l-1]) \cdot \begin{pmatrix} x[i] \\ \cdots \\ x[i-l+1] \end{pmatrix} \quad (2)$$

$$= \underline{\sigma}_n \cdot \begin{pmatrix} g^{(0)}[0] & \cdots & g^{(0)}[l-1] \\ g^{(-1)}[0] & \cdots & g^{(-1)}[l-1] \\ \vdots & \cdots & \vdots \\ g^{(-n+1)}[0] & \cdots & g^{(-n+1)}[l-1] \end{pmatrix} \cdot \underline{x}_l,$$

where $\underline{\sigma}_n = (-2^0 \ 2^{-1} \ \cdots \ 2^{(-n+1)})$, each element $h^{(-r)}[j]$, $g^{(-r)}[j]$ represents bit 2^{-r} of $h[j]$ and $g[j]$, respectively, the $()^t$ operator stands for transposed, $\underline{x}_\xi = (x[i] \cdots x[i-\xi+1])^t$ and ξ can be either the length of the low pass or high pass filter (k or l).

For a generic filter, the DA-based architecture is obtained by computing the product between $h^{(-r)}[j]$ (or $g^{(-r)}[j]$) and $\underline{x}_k$ (or $\underline{x}_l$) first, then, the result is multiplied by $\underline{\sigma}_n$. Let $\mathbf{h}_{n,k}$ and $\mathbf{g}_{n,l}$ be the matrices containing $h^{(-r)}[j]$ and $g^{(-r)}[j]$, respectively, and $\underline{u} = \mathbf{h}_{n,k} \cdot \underline{x}_k$ and $\underline{v} = \mathbf{g}_{n,l} \cdot \underline{x}_l$, we then obtain $y_l[i] = \underline{\sigma}_n \cdot \underline{u}$ and $y_h[i] = \underline{\sigma}_n \cdot \underline{v}$.

This factorization leads to a 3-stage architecture:

1) a butterfly circuit made of adders to implement the $\mathbf{h}_{n,k}$ and $\mathbf{g}_{n,l}$ matrix product;
2) a hard-wired shift network to apply the $\underline{\sigma}_n$ vector;
3) a tree adder to combine partial results.

In Fig. 2 the generic DA-based architecture to implement $y[i]$ ($y_l[i]$ or $y_h[i]$) is depicted, where ξ is the filter length (k or

l), $w^{(-r)}$ terms are the results of the matrix product ($\underline{u}$ or $\underline{v}$), $>> r$ represents an r-position right-shift and $f^{(-r)} = w^{(-r)} >> r$. As detailed in Section IV, the downsampling operation at the output of the analysis filters is exploited to alternatively compute $y_l[i]$ and $y_h[i]$.

B. Synthesis filters

The computational scheme used for the analysis filters can be used to implement the synthesis filters as well. Indeed, synthesis filters can be obtained from the analysis ones [5] as

$$\dot{h}[j] = (-1)^j \cdot g[j] \qquad \dot{g}[j] = (-1)^j \cdot (-h[j]), \quad (3)$$

with $\dot{k} = l$ and $\dot{l} = k$. Moreover, the right part of Fig. 1 (synthesis filters) shows that

$$\dot{x}[i] = \left(\dot{h}[0] \cdots \dot{h}[\dot{k}-1] \right) \cdot \begin{pmatrix} \dot{y}_l[i] \\ \cdots \\ \dot{y}_l[i-\dot{k}+1] \end{pmatrix} \quad (4)$$

$$+ \left(\dot{g}[0] \cdots \dot{g}[\dot{l}-1] \right) \cdot \begin{pmatrix} \dot{y}_h[i] \\ \cdots \\ \dot{y}_h[i-\dot{l}+1] \end{pmatrix}$$

$$= \underline{\sigma}_n \cdot (\underline{\dot{u}} + \underline{\dot{v}})$$

where $\dot{y}_l[i]$ and $\dot{y}_h[i]$ are obtained by upsampling the $y_l[i]$ and $y_h[i]$ signals, $\underline{\dot{u}} = \dot{\mathbf{h}}_{n,k} \cdot \underline{\dot{y}}_{l_k}$ and $\underline{\dot{v}} = \dot{\mathbf{g}}_{n,l} \cdot \underline{\dot{y}}_{h_l}$ with $\underline{\dot{y}}_{l_k} = (\dot{y}_l[i] \cdots \dot{y}_l[i-\dot{k}+1])^t$ and $\underline{\dot{y}}_{h_l} = (\dot{y}_h[i] \cdots \dot{y}_h[i-\dot{l}+1])^t$. Besides, taking into account the *zeros* added by the upsampling blocks and the input shift register (shown on the left side of Fig. 2), we obtain

$$\underline{\dot{u}} + \underline{\dot{v}} = \begin{cases} \dot{\mathbf{hg}}_{n,\dot{\xi}} \cdot \underline{\dot{y}}_{lh_{\dot{\xi}}} & \text{if } \dot{y}_l[i] \neq 0 \uparrow \\ \dot{\mathbf{gh}}_{n,\dot{\xi}} \cdot \underline{\dot{y}}_{hl_{\dot{\xi}}} & \text{otherwise} \end{cases} \quad (5)$$

where $0 \uparrow$ means a zero added by the upsampling and

$$\dot{\mathbf{hg}}_{n,\dot{\xi}} = \begin{pmatrix} \dot{h}^{(0)}[0] & \dot{g}^{(0)}[1] & \cdots \\ \dot{h}^{(-1)}[0] & \dot{g}^{(-1)}[1] & \cdots \\ \vdots & \vdots & \vdots \\ \dot{h}^{(-n+1)}[0] & \dot{g}^{(-n+1)}[1] & \cdots \end{pmatrix}, \quad (6)$$

$$\underline{\dot{y}}_{lh_{\dot{\xi}}} = \begin{pmatrix} \dot{y}_l[i] \\ \dot{y}_h[i+1] \\ \vdots \end{pmatrix}, \quad (7)$$

$$\dot{\mathbf{gh}}_{n,\dot{\xi}} = \begin{pmatrix} \dot{g}^{(0)}[0] & \dot{h}^{(0)}[1] & \cdots \\ \dot{g}^{(-1)}[0] & \dot{h}^{(-1)}[1] & \cdots \\ \vdots & \vdots & \vdots \\ \dot{g}^{(-n+1)}[0] & \dot{h}^{(-n+1)}[1] & \cdots \end{pmatrix}, \quad (8)$$

$$\underline{\dot{y}}_{hl_{\dot{\xi}}} = \begin{pmatrix} \dot{y}_h[i] \\ \dot{y}_l[i+1] \\ \vdots \end{pmatrix} \quad (9)$$

with $\dot{\xi} = \max\{\dot{k}, \dot{l}\}$. Thus, the rth row of $\dot{\mathbf{hg}}_{n,\dot{\xi}}$ and $\dot{\mathbf{gh}}_{n,\dot{\xi}}$ contains bit 2^{-r} of the interlaced sequences of taps, namely $(\dot{h}[0], \dot{g}[1], \dot{h}[2], \dot{g}[3] \cdots)$ and $(\dot{g}[0], \dot{h}[1], \dot{g}[2], \dot{h}[3] \cdots)$, respectively. As a consequence, when $\dot{k} \leq j < \dot{\xi}$ or $\dot{l} \leq j < \dot{\xi}$ the corresponding taps ($\dot{h}[j]$ or $\dot{g}[j]$) are *zero*, leading to columns
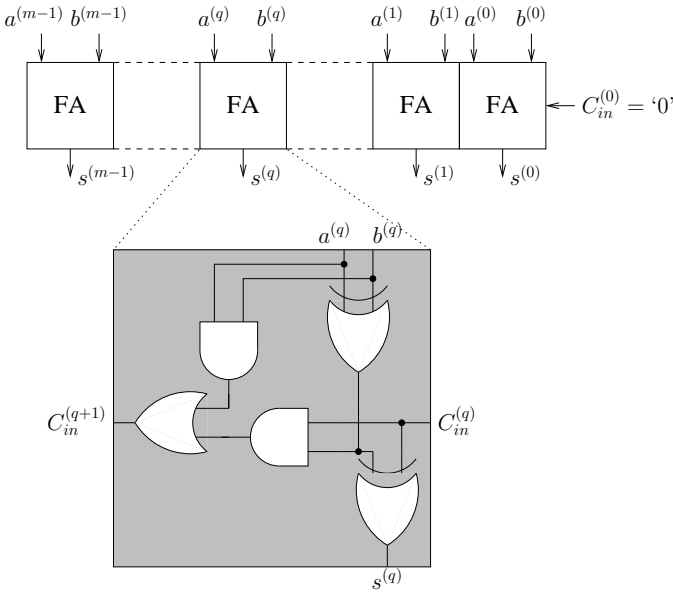
Figure 3: Ripple carry adder.

of zeros in $\dot{\mathbf{h}}\mathbf{g}_{n,\xi}$ and $\mathbf{g}\dot{\mathbf{h}}_{n,\xi}$, respectively. Unfortunately, the effectiveness of DA-based architectures applied to synthesis side of the FB strongly depends on the symmetry of the wavelet filters. Indeed, in Section IV-A we show that for the 9/7 filters the architecture for the synthesis filters is nearly the same as the one for the analysis filters. On the contrary, the architecture for the synthesis filters of the 10/18 wavelet is very different from the analysis one (see Section IV-C).

III. RESULT-BIASED CIRCUITS FOR DA-BASED ARCHITECTURES

Let us consider the circuit shown in Fig. 3 to compute $s = a + b$, where the gray shaded box highlights the circuit of a full adder (FA) and a and b are represented as 2's complement values using m bits. Let $p_{\chi^{(q)}}$ be the probability that the qth bit of χ is equal to '1', where χ is one of the signals involved in the addition, namely a, b, s, c_{in}, and c_{in} is the carry-in signal. From Fig. 3 one infers that:

$$p_{s^{(q)}} = p_{c_{in}^{(q)}}[(1 - p_{a^{(q)}})(1 - p_{b^{(q)}}) + p_{a^{(q)}}p_{b^{(q)}}] + \quad (10)$$
$$(1 - p_{c_{in}^{(q)}})[p_{a^{(q)}}(1 - p_{b^{(q)}}) + (1 - p_{a^{(q)}})p_{b^{(q)}}].$$

Let us introduce a threshold $T^{(q)}$ such that, if $p_{c_{in}^{(q)}}$ is sufficiently small, then the following approximation holds true

$$p_{s^{(q)}} \approx p_{a^{(q)}}(1 - p_{b^{(q)}}) + (1 - p_{a^{(q)}})p_{b^{(q)}} \quad \text{if } p_{c_{in}^{(q)}} < T^{(q)}. \quad (11)$$

Since this approximation biases the result of the addition, the value of $T^{(q)}$ is used to tune the bias effect. In this work we investigate two strategies to select $T^{(q)}$. These strategies are referred to as *shift-based* and *probability-based* thresholding, respectively and will be described in the following paragraphs.

A. Shift-based thresholding

As shown in Fig. 2, DA-based architectures require right shift operations at the output of the butterfly circuit. It is

Table I: Coefficients for the 9/7 analysis filters.

j	$h[j]$	$g[j]$
0	0.60294901823636	0.55754352622850
± 1	0.26686411844287	-0.29563588155713
± 2	-0.07822326652899	-0.02877176311425
± 3	-0.01686411844287	0.04563588155713
± 4	0.02674875741081	

known that, in fixed point implementation, changing the order between additions and right shift operations leads to a precision loss, i.e. $s_r = [(a + b) >> r] \neq [(a >> r) + (b >> r)]$. However, if $\hat{q}$ is the maximum value such that $\{(\hat{q} < r) \wedge (p_{c_{in}^{(\hat{q})}} < T^{(\hat{q})})\}$, then we obtain that

$$(a + b) >> r \approx [(a >> \hat{q}) + (b >> \hat{q})] >> (r - \hat{q}). \quad (12)$$

As a consequence, if a and b are represented using m bits, then we obtain an approximate version of s_r by employing $(m - \hat{q})$ instead of m FAs.

B. Probability-based thresholding

Another circuit employed in DA-based architectures is the tree adder. As it can be inferred from Fig. 2, the data combined by the tree adder come from different paths and the magnitude of two $f^{(-r)}$ terms, out of the n available ones, can be very different. Let us assume that all $x[i]$ samples have the same order of magnitude, then, the difference between $|f^{(0)}|$ and $|f^{(-n+1)}|$ can be large due to the shift operation. This idea can be exploited to predict the probability of the qth carry signal to be '1':

$$p_{c_{in}^{(q+1)}} = p_{c_{in}^{(q)}}[(1 - p_{a^{(q)}})p_{b^{(q)}} + p_{a^{(q)}}(1 - p_{b^{(q)}})] + p_{a^{(q)}}p_{b^{(q)}}. \quad (13)$$

For some of the values taken by a, if $b \geq 0$ and b is "small", then

$$\exists \, \tilde{q}_0 \mid \{p_{b^{(q)}} = 0 \quad \forall q \mid \tilde{q}_0 \leq q \leq m - 1\}. \quad (14)$$

The condition in (14) applied to (13) leads to

$$p_{c_{in}^{(q+1)}} = p_{a^{(q)}}p_{c_{in}^{(q)}} \leq p_{c_{in}^{(q)}}, \quad (15)$$

with $\tilde{q}_0 \leq q \leq m - 1$. From (15) we can infer that in many cases the probability of the carry-in signal for the most significant bits tends to 0. Analogously, if $b < 0$ and $|b|$ is "small", then

$$\exists \, \tilde{q}_1 \mid \{p_{b^{(q)}} = 1 \quad \forall q \mid \tilde{q}_1 \leq q \leq m - 1\}, \quad (16)$$

which leads to

$$p_{c_{in}^{(q+1)}} = p_{c_{in}^{(q)}} + p_{a^{(q)}}(1 - p_{c_{in}^{(q)}}) \geq p_{c_{in}^{(q)}}, \quad (17)$$

with $\tilde{q}_1 \leq q \leq m - 1$. From (17) we infer that the carry-in probability tends to 1. In order to address both cases in (15) and (17), we introduce μ_x, which is the mean value of the input signal, and we observe that the probability of carry-in signals as a function of q creates two regions: i) the Most-Significant-Bit-region (MSB-region), where $p_{c_{in}^{(q)}}$ tends to 0 or 1 depending on μ_x, ii) the Least-Significant-Bit-region (LSB-region), where $p_{c_{in}^{(q)}}$ depends on the statistic of the input signal and $\tilde{q}$ (either $\tilde{q}_0$ or $\tilde{q}_1$) is the position at the border between the MSB-region and the LSB-region.

Table II: Values of the $\mathbf{h}_{13,9}$ and $-\mathbf{g}_{13,7}$ coefficients and corresponding $\underline{d}_j$ vectors for the 9/7 wavelet filters.

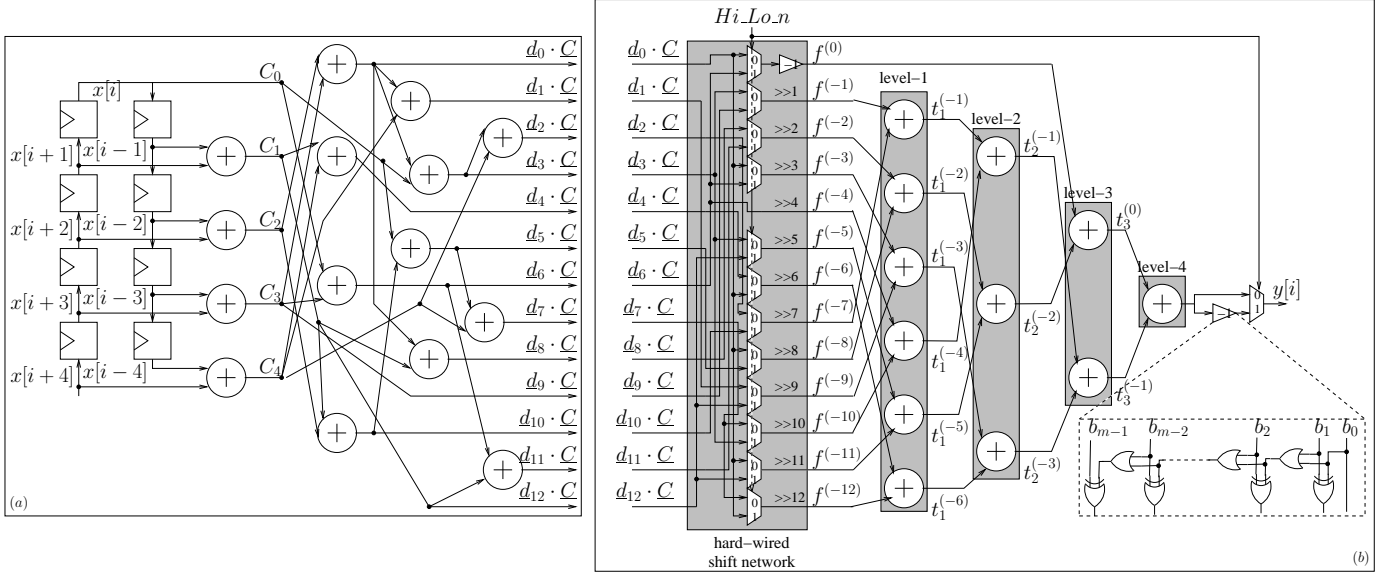| $-r$	$h[\pm 4]$	$h[\pm 3]$	$h[\pm 2]$	$h[\pm 1]$	$h[0]$	$\underline{d}_j$	$-g[\pm 3]$	$-g[\pm 2]$	$-g[\pm 1]$	$-g[0]$	$\underline{d}_j$
0	0	1	1	0	0	$\underline{d}_0$	1	0	0	1	$\underline{d}_6$
1	0	1	1	0	1	$\underline{d}_3$	1	0	0	0	$\underline{d}_9$
2	0	1	1	1	0	$\underline{d}_8$	1	0	1	1	$\underline{d}_{11}$
3	0	1	1	0	0	$\underline{d}_0$	1	0	0	1	$\underline{d}_6$
4	0	1	0	0	1	$\underline{d}_6$	1	0	0	1	$\underline{d}_6$
5	0	1	1	0	1	$\underline{d}_3$	0	0	1	0	$\underline{d}_{12}$
6	1	0	0	1	0	$\underline{d}_4$	1	1	0	0	$\underline{d}_0$
7	1	1	1	0	1	$\underline{d}_2$	0	1	1	0	$\underline{d}_{10}$
8	0	1	1	0	0	$\underline{d}_0$	0	1	1	1	$\underline{d}_5$
9	1	1	1	0	0	$\underline{d}_1$	0	0	1	0	$\underline{d}_{12}$
10	1	0	1	1	1	$\underline{d}_7$	1	1	0	1	$\underline{d}_3$
11	0	1	1	0	0	$\underline{d}_0$	0	0	1	0	$\underline{d}_{12}$
12	1	0	1	1	1	$\underline{d}_7$	1	1	0	0	$\underline{d}_0$

Figure 4: Butterfly circuit (a) and tree adder with hard-wired shift network computational scheme (b) for the 9/7 wavelet filters.

To maximize the occurrence of the conditions in (15) and (17), we add $f^{(-r)}$ values as follows:

$$t_1^{(-i)} = f^{(-i)} + f^{(-n/2-i)}, \tag{18}$$

with $0 \le i < n/2$. Then, by setting a threshold T, we can find

$$q^* = \max\{q\} \in [0, \tilde{q}] \mid p_{c_{in}^{(q^*)}} < T \tag{19}$$

and force $c_{in}^{(q)} = 0$ for $0 \le q \le q^*$. The same approach can be extended to all the levels in the tree adder. Let $T_{\zeta,\vartheta}$ be the threshold for adder ϑ at level ζ and $q^*_{\zeta,\vartheta}$ the position of the last FA such that $c_{in}^{(q)} = 0$ for $0 \le q \le q^*_{\zeta,\vartheta}$. If the input values are represented using m bits, then we can obtain an approximate version of each result by employing $(m - q^*_{\zeta,\vartheta})$ instead of m FAs.

IV. CASES OF STUDY: RESULT-BIASED DA-BASED ARCHITECTURES FOR THE 9/7 AND 10/18 ANALYSIS WAVELET FILTERS

Two important cases of study are shown in the following: the experimental results obtained by implementing result-biased DA-based architectures for the 9/7 and 10/18 wavelet filters. In order to show both the complexity reduction and the

Table III: List of $\underline{d}_j$ vectors and corresponding $\mathcal{I}_j$ sets for the 9/7 wavelet filters.

$\underline{d}_j$	$\mathcal{I}_j$	$\min_r\{\mathcal{I}_j\}$
$\underline{d}_0$	$\mathcal{I}_0 = \{u^{(0)}, u^{(-3)}, u^{(-8)}, u^{(-11)}, v^{(-6)}, v^{(-12)}\}$	0
$\underline{d}_1$	$\mathcal{I}_1 = \{u^{(-9)}\}$	9
$\underline{d}_2$	$\mathcal{I}_2 = \{u^{(-7)}\}$	7
$\underline{d}_3$	$\mathcal{I}_3 = \{u^{(-1)}, u^{(-5)}, u^{(-10)}\}$	1
$\underline{d}_4$	$\mathcal{I}_4 = \{u^{(-6)}\}$	6
$\underline{d}_5$	$\mathcal{I}_5 = \{u^{(-8)}\}$	8
$\underline{d}_6$	$\mathcal{I}_6 = \{u^{(-4)}, v^{(0)}, v^{(-3)}, v^{(-4)}\}$	0
$\underline{d}_7$	$\mathcal{I}_7 = \{u^{(-10)}, u^{(-12)}\}$	10
$\underline{d}_8$	$\mathcal{I}_8 = \{u^{(-2)}\}$	2
$\underline{d}_9$	$\mathcal{I}_9 = \{v^{(-1)}\}$	1
$\underline{d}_{10}$	$\mathcal{I}_{10} = \{v^{(-7)}\}$	7
$\underline{d}_{11}$	$\mathcal{I}_{11} = \{v^{(-2)}\}$	2
$\underline{d}_{12}$	$\mathcal{I}_{12} = \{v^{(-5)}, v^{(-9)}, v^{(-11)}\}$	5

performance achieved by the proposed result-biased DA-based architectures, we modified *openjpeg* [26], a Class-1 Profile-1 compliant open source JPEG2000 implementation[1]. To be compatible with the *openjpeg* model, we represented $h[j]$ and $g[j]$ taps with 1 bit for the integer part and 12 bits for the fractional part ($n = 13$). Internal data are represented as 16 bit

---

[1]For other profiles related to Digital Cinema the reader can refer to [27].

Table IV: Values of the $\dot{\mathbf{hg}}_{13,9}$ and $\dot{\mathbf{gh}}_{13,9}$ coefficients and corresponding $\underline{\dot{d}}_j$ vectors for the 9/7 wavelet filters.

$-r$	$\dot{h}[\pm 4]$	$\dot{g}[\pm 3]$	$\dot{h}[\pm 2]$	$\dot{g}[\pm 1]$	$\dot{h}[0]$	$\underline{\dot{d}}_j$	$\dot{g}[\pm 4]$	$\dot{h}[\pm 3]$	$\dot{g}[\pm 2]$	$\dot{h}[\pm 1]$	$\dot{g}[0]$	$\underline{\dot{d}}_j$
0	0	1	1	0	0	$\underline{\dot{d}}_0$	1	1	0	0	1	$\underline{\dot{d}}_{10}$
1	0	1	1	0	1	$\underline{\dot{d}}_3$	1	1	0	0	0	$\underline{\dot{d}}_{11}$
2	0	1	1	1	0	$\underline{\dot{d}}_8$	1	1	0	1	1	$\underline{\dot{d}}_{13}$
3	0	1	1	0	0	$\underline{\dot{d}}_0$	1	1	0	0	1	$\underline{\dot{d}}_{10}$
4	0	1	1	0	0	$\underline{\dot{d}}_0$	1	1	1	0	0	$\underline{\dot{d}}_1$
5	0	1	1	0	1	$\underline{\dot{d}}_3$	1	0	0	1	0	$\underline{\dot{d}}_4$
6	0	0	0	1	1	$\underline{\dot{d}}_5$	0	1	1	0	1	$\underline{\dot{d}}_3$
7	0	1	0	0	1	$\underline{\dot{d}}_6$	0	0	0	1	0	$\underline{\dot{d}}_{12}$
8	0	1	0	0	0	$\underline{\dot{d}}_9$	1	0	0	1	1	$\underline{\dot{d}}_{14}$
9	0	1	1	0	1	$\underline{\dot{d}}_3$	0	0	0	1	1	$\underline{\dot{d}}_{15}$
10	0	0	0	1	1	$\underline{\dot{d}}_5$	0	1	0	0	0	$\underline{\dot{d}}_9$
11	0	1	1	0	0	$\underline{\dot{d}}_0$	1	0	0	1	1	$\underline{\dot{d}}_{14}$
12	0	0	1	1	0	$\underline{\dot{d}}_7$	1	1	1	0	1	$\underline{\dot{d}}_2$

fixed point values ($m = 16$) as in other works, e.g. [4], [9]. For our simulations five standard images (256 gray levels), namely 'Lena' 512×512, 'Barbara' 512×512, 'Boat' 512×512, 'Goldhill' 512×512 and 'Fingerprint' 512×512 [28], have been employed[2]. The number of DWT decomposition levels (L) has been varied from 1 to 4. This corresponds to $\Lambda = L+1$, where Λ is the number of DWT resolution levels required by *openjpeg*. Different compression ratios (ρ) have been imposed, namely 1:1, 8:1, 16:1, 32:1 and 64:1, precinct and code-block size are the encoder default values. Simulations shown in this work have been obtained by modifying the encoder, namely we implemented the forward DWT with the DA-based solution proposed in [20] for the 9/7 DWT. Then, the DA-based solution has been extended to support the 10/18 wavelet filters as well. Finally, we implemented the proposed result-biasing techniques.

A. DA-based architecture for the 9/7 wavelet filters

As argued in [20], it is more convenient to consider the binary representation of $h[j]$ and $-g[j]$, instead of $h[j]$ and $g[j]$, to find terms that are common to both the low pass and the high pass taps. Given that the 9/7 wavelet filters are symmetric (see Table I), we can further reduce the complexity of the butterfly circuit. These two considerations permit to write $\mathbf{h}_{n,k}$ and $-\mathbf{g}_{n,l}$ for the 9/7 filters, as shown in Table II, where repeated common-term-vectors ($\underline{d}_j$) are gray-shaded. Moreover, to exploit filter symmetry, we introduce the column vector $\underline{C}$, which elements are

$$C_\omega = \begin{cases} x[i] & \omega = 0 \\ x[i+\omega] + x[i-\omega] & \omega = 1, \dots, 4 \end{cases} . \quad (20)$$

Then, we produce the $w^{(-r)}$ values, as shown in Fig. 4 (a), by combining $\underline{C}$ with the 13 possible $\underline{d}_j$ vectors. As an example, Table II shows that $\underline{d}_0 \cdot \underline{C}$, where $\underline{d}_0 = [0\ 1\ 1\ 0\ 0]$, is used to calculate $u^{(0)}$, $u^{(-3)}$, $u^{(-8)}$, $u^{(-11)}$ for the low pass branch and $v^{(-6)}$, $v^{(-12)}$ for the high pass branch. In general, every product $\underline{d}_j \cdot \underline{C}$ defines a set ($\mathcal{I}_j$) made of the proper $\underline{u}$ and $\underline{v}$ elements, e.g. $\mathcal{I}_0 = \{u^{(0)}\ u^{(-3)}\ u^{(-8)}\ u^{(-11)}\ v^{(-6)}\ v^{(-12)}\}$, as shown in Table III. Furthermore, as argued in [20], a

[2]Other images have been tested as well. Since the results we obtained are similar to ones presented in this paper, we are not showing them for the sake of brevity.

Reduced-Adder-Graph-like technique [29], where common sub-expressions are extracted and calculated only once, reduces the number of adders required by the butterfly circuit. As an example, sub-expression $C_3 + C_2$, which is common to several $\underline{d}_j \cdot \underline{C}$ products, is computed only once and then reused multiple times.

A similar approach can be employed for the synthesis filters, where odd filter lengths and the symmetry of $\dot{\mathbf{hg}}_{13,9}$ and $\dot{\mathbf{gh}}_{13,9}$ matrices can be exploited to define

$$\dot{C}_\omega = \begin{cases} \dot{y}[i] & \omega = 0 \\ \dot{y}[i+\omega] + \dot{y}[i-\omega] & \omega = 1, \dots, 4 \end{cases} , \quad (21)$$

where $\dot{y}[i]$ can be either $\dot{y}_l[i]$ or $\dot{y}_h[i]$ (see the notation introduced in Section II-B). The corresponding butterfly circuit is very similar to the one shown in Fig. 4 (a) and can be derived from the $\underline{\dot{d}}_j$ vectors summarized in Table IV. Finally, both analysis and synthesis architectures rely on a shift network and a tree adder to compute the results, as shown in Fig. 2 for a general case.

B. Result-biased DA-based architecture for the 9/7 analysis wavelet filters

1) Implementation of the result-biased butterfly circuit: As described in section III-A, we can reduce the number of FAs required to compute the $w^{(-r)}$ terms as follows. Since the $\underline{d}_j \cdot \underline{C}$ products define 13 different sets ($\mathcal{I}_j$), we have 13 possible $\hat{q}_j$ values. Every $\hat{q}_j$ is the maximum value that satisfies $\{(q < \min_r\{\mathcal{I}_j\}) \wedge (p_{c_{in}^{(q)}} < T^{(q)})\}$, where $\min_r\{\mathcal{I}_j\}$ is the minimum among the possible shift amounts (r) in $\mathcal{I}_j$. As an example, $\min_r\{\mathcal{I}_0\} = 0$ implies that the elements in $\mathcal{I}_0$ are not affected by result-biasing. On the contrary, $\underline{d}_{10} \cdot \underline{C}$, where $\underline{d}_{10} = [0\ 0\ 1\ 1\ 0]$, is employed to calculate only $v^{(-7)}$. Thus, $\min_r\{\mathcal{I}_{10}\} = 7$ so $v^{(-7)}$ can be approximated by finding $\hat{q}_{10} = \max_q\{(q < 7) \wedge (p_{c_{in}^{(q)}} < T^{(q)})\}$.

To set each $T^{(q)}$, we simulated the proposed DA-based result-biased DWT in the *openjpeg* model with the test conditions detailed at the beginning of section IV. In Table V we show the results obtained by choosing $T^{(q)}$ such that $\hat{q}_j = \min_r\{\mathcal{I}_j\} - 1$. As an example, $\hat{q}_0 = \min_r\{\mathcal{I}_0\} - 1 = -1$ means that the elements in $\mathcal{I}_0$ are not biased. Experimental results show that the Peak Signal to Noise Ratio (PSNR)

Table V: PSNR comparison between the DA-based DWT [20] (column $\mathrm{PSNR_{DA}}$) and the proposed DA-based DWT with result-biased butterfly circuit (column $\Delta\mathrm{PSNR_{BB}} = \mathrm{PSNR_{DA}} - \mathrm{PSNR_{BB}}$) for the 9/7 filters.

| Image | L | $\mathrm{PSNR_{DA}}$ [dB] | | | | | $\Delta\mathrm{PSNR_{BB}}$ [dB] | | | | |
		1:1	8:1	16:1	32:1	64:1	1:1	8:1	16:1	32:1	64:1
Lena	1	49.30	38.94	34.24	30.01	24.13	0.00	0.00	0.00	0.00	0.00
	2	48.79	39.94	36.68	33.20	29.59	0.00	0.00	0.00	0.00	0.00
	3	48.48	40.05	37.19	34.05	30.87	0.00	0.00	0.00	0.00	0.00
	4	48.21	40.05	37.20	34.11	30.98	0.00	0.01	0.00	0.00	0.00
Barbara	1	49.49	35.79	29.42	24.04	20.43	0.00	0.00	0.00	0.00	0.00
	2	48.97	37.59	32.16	27.77	24.17	0.02	0.00	0.00	0.00	0.00
	3	48.62	37.83	32.78	28.75	25.30	-0.01	0.00	0.00	0.00	0.00
	4	48.34	37.85	32.77	28.80	25.78	-0.03	-0.01	0.00	0.00	0.00
Boat	1	49.37	37.63	32.62	28.23	23.39	0.00	0.00	-0.02	0.00	0.00
	2	48.76	38.88	33.99	30.14	26.96	-0.06	0.00	0.00	0.00	0.00
	3	48.37	39.02	34.44	30.91	27.86	0.01	0.00	0.00	0.00	0.00
	4	48.05	39.01	34.52	30.97	28.05	0.01	0.01	0.00	0.00	0.00
Goldhill	1	49.68	35.79	31.69	27.55	23.13	0.00	0.00	0.00	0.00	0.00
	2	49.19	36.34	32.88	30.16	27.37	0.00	0.00	0.00	0.00	0.00
	3	48.78	36.40	33.14	30.52	28.44	0.00	-0.04	0.00	0.00	0.00
	4	48.32	36.41	33.18	30.49	28.46	-0.02	0.00	0.00	0.00	-0.01
Fingerprint	1	49.47	35.83	31.74	27.76	17.76	0.00	0.00	0.00	0.00	0.00
	2	48.85	36.19	32.36	29.14	25.99	-0.01	0.00	0.00	0.00	0.00
	3	48.18	36.22	32.43	29.47	26.78	-0.01	0.00	0.00	0.00	0.00
	4	47.42	36.15	32.45	29.50	26.87	-0.02	0.00	0.00	0.00	0.00

(a) lp tree adder first level: low pass generation.

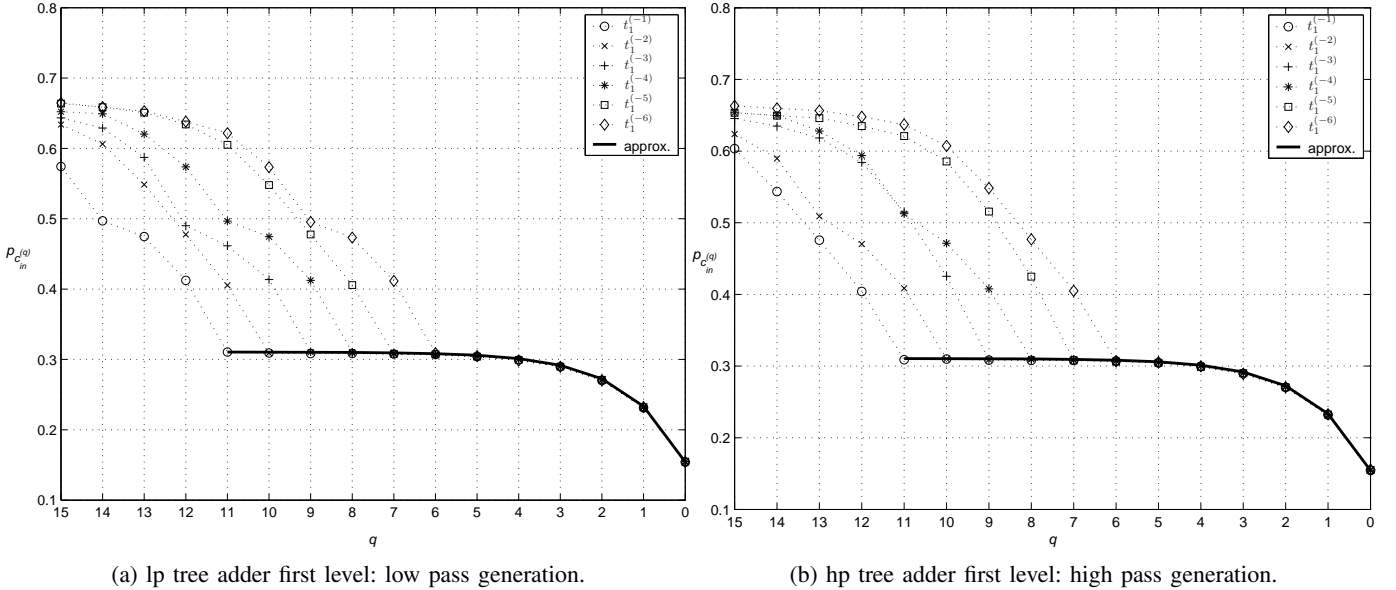(b) hp tree adder first level: high pass generation.

Figure 5: Values of $p_{c_{in}^{(q)}}$ vs q in the low pass (lp) and high pass (hp) computation of the tree adder first level with result-biased butterfly circuit for the 'Goldhill' image.

difference ($\Delta\mathrm{PSNR_{BB}}$) between the original DA-based DWT ($\mathrm{PSNR_{DA}}$) and the proposed one is negligible, when result-biasing is applied to the butterfly circuit (BB, $\mathrm{PSNR_{BB}}$). Moreover, the standard butterfly circuit [20] requires $15 \times m$ FAs, where $4 \times m$ FAs are required to compute $\underline{C}$. On the other hand, the proposed result-biased butterfly saves $\sum_{j=0}^{12} \Delta_j = 55$ FAs, where $\Delta_j = \hat{q}_j + 1 = \min_r\{\mathcal{I}_j\}$. As an example, since $\mathcal{I}_{10} = v^{(-7)}$, the computation of $v^{(-7)}$ requires only $m-7$ FAs. Since $m = 16$, the standard butterfly circuit requires 240 FAs, whereas the proposed one requires 240-55=185 FAs.

2) Result-biased tree adder implementation: Stemming from the computational scheme defined in the previous section, the 13 different $f^{(-r)}$ values are added together. As detailed

Table VI: Parameters used to approximate $p_{c_{in}^{(q)}}$ in the LSB-region.

level	adder	α	β	γ	δ	τ
1	all	0.31	0.15	0	3	$\sqrt{2}$
2	$t_2^{(-1)}$	0.31	0.11	2	4	$\sqrt{2}$
2	$t_2^{(-2)}$	0.31	0.09	1	4	$\sqrt{2}$
2	$t_2^{(-3)}$	0.31	0.06	0	4	$\sqrt{2}$
3	$t_3^{(0)}$	0.31	0.11	3	10	$\sqrt{2}$
3	$t_3^{(-1)}$	0.31	0.11	2	3	$\sqrt{2}$

in section III-B, we combine $f^{(-r)}$ values as in (18). Fig. 4 (b) shows the tree adder and the hard-wired shift network used in the architecture for the 9/7 wavelet filters. As it can be observed, the Hi_Lo_n signal produces $y_h[i]$ ($Hi\_Lo\_n =$'1')

Table VII: PSNR loss with respect to the original DA-based architecture for the 9/7 filters. Results are obtained by enabling result-biased butterfly circuit and result-biasing at the first level of the tree adder $\Delta\text{PSNR}_{\text{BB+TB1}} = \text{PSNR}_{\text{DA}} - \text{PSNR}_{\text{BB+TB1}}$; similarly for the second and third level with $\Delta\text{PSNR}_{\text{BB+TB1/2}} = \text{PSNR}_{\text{DA}} - \text{PSNR}_{\text{BB+TB1/2}}$ and $\Delta\text{PSNR}_{\text{BB+TB1/2/3}} = \text{PSNR}_{\text{DA}} - \text{PSNR}_{\text{BB+TB1/2/3}}$, respectively.

Image	L	$\Delta\text{PSNR}_{\text{BB+TB1}}$ [dB]					$\Delta\text{PSNR}_{\text{BB+TB1/2}}$ [dB]					$\Delta\text{PSNR}_{\text{BB+TB1/2/3}}$ [dB]				
		1:1	8:1	16:1	32:1	64:1	1:1	8:1	16:1	32:1	64:1	1:1	8:1	16:1	32:1	64:1
Lena	1	0.10	0.02	-0.06	0.00	0.00	0.11	0.01	-0.08	0.00	0.00	0.11	0.01	-0.09	0.00	0.00
	2	0.38	0.04	-0.04	0.01	0.00	0.32	0.04	-0.04	0.01	0.00	0.32	0.04	-0.04	0.01	0.00
	3	0.43	0.07	0.04	0.02	0.01	0.48	0.08	0.05	0.02	0.01	0.49	0.08	0.05	0.03	0.01
	4	0.87	0.15	0.07	0.05	0.00	0.84	0.16	0.08	0.05	0.00	0.97	0.16	0.08	0.05	0.01
Barbara	1	0.07	0.00	-0.06	-0.01	-0.21	0.08	-0.01	-0.06	0.00	-0.21	0.08	-0.01	-0.06	0.00	-0.21
	2	0.17	0.00	0.00	0.09	0.01	0.20	0.00	0.00	0.09	0.01	0.18	0.00	0.00	0.09	0.01
	3	0.29	0.04	0.01	0.02	0.00	0.33	0.03	0.00	0.01	-0.28	0.34	0.03	0.00	0.01	-0.28
	4	0.46	0.06	-0.01	-0.01	0.00	0.55	0.07	0.00	-0.02	0.00	0.55	0.08	0.00	-0.02	0.00
Boat	1	0.14	-0.02	-0.01	0.02	-0.15	0.21	-0.01	0.00	0.02	-0.15	0.15	-0.01	0.00	0.02	-0.15
	2	0.43	0.03	0.02	0.01	0.01	0.46	0.02	0.01	0.01	0.01	0.46	0.02	0.01	0.01	0.01
	3	0.83	0.08	0.00	0.02	0.01	0.89	0.09	0.03	0.02	0.01	0.91	0.09	0.00	0.02	0.01
	4	1.29	0.18	0.04	0.02	0.06	1.38	0.18	0.05	0.03	0.06	1.40	0.19	0.05	0.03	0.06
Goldhill	1	0.08	0.02	0.00	0.08	0.00	0.08	0.02	0.00	0.08	0.00	0.08	0.02	0.00	0.08	0.00
	2	0.18	0.02	0.05	0.00	0.00	0.20	0.02	0.05	0.01	0.00	0.20	0.02	0.05	0.01	0.00
	3	0.31	-0.01	0.02	0.00	0.00	0.36	-0.01	0.02	0.00	0.00	0.37	-0.01	0.02	0.00	0.00
	4	0.47	0.04	0.02	0.01	0.00	0.54	0.05	0.04	0.00	0.00	0.57	0.05	0.04	0.00	0.00
Fingerprint	1	-0.09	0.02	0.01	-0.06	0.04	-0.08	0.02	0.01	-0.06	0.04	-0.08	0.02	0.01	-0.06	0.04
	2	-0.31	0.01	0.00	0.00	0.00	-0.30	0.01	0.00	0.00	0.00	-0.26	0.01	0.00	0.00	0.01
	3	-0.64	0.02	-0.02	0.00	0.00	-0.63	0.02	-0.02	0.00	0.00	-0.49	-0.02	-0.01	0.00	0.00
	4	-1.01	-0.07	0.03	-0.01	-0.01	-1.01	-0.07	-0.02	-0.01	-0.01	-1.01	-0.06	-0.01	-0.01	-0.01

or $y_l[i]$ ($Hi\_Lo\_n =$ '0') by selecting the proper $\underline{d}_j \cdot \underline{C}$ input to the hard-wired shift network.

As discussed in Section III-B, the delay and the complexity of the tree adder can be reduced by cutting the carry chains, namely by fixing the value of $T_{\zeta,\vartheta}$ and by finding the corresponding $q^*_{\zeta,\vartheta}$. Simulations were performed on the modified *openjpeg* model in the test conditions described in the first paragraph of Section IV and including the result-biased butterfly circuit described in Section IV-B1. As an example, Fig. 5 shows the values of $p_{c^{(q)}_{in}}$ obtained with the 'Goldhill' image by varying $q \in [0, 15]$ for low pass (lp) and high pass (hp) filters, respectively, at the first level of the tree adder (referred to as $t_1^{(-i)}$ in Fig. 4 (b), with $i = 1, \ldots, 6$). Since the *openjpeg* model converts image pixels from $[0, 255]$ to $[-128, 127]$, then $\mu_x < 0$ for the 'Goldhill' image. Indeed, the values of $p_{c^{(q)}_{in}}$ in the MSB-region (Fig. 5) tend to 1, whereas in the LSB-region $p_{c^{(q)}_{in}} \in [0.15, 0.31]$. Simulations show that in both low pass and high pass computation the following approximation holds true:

$$p_{c^{(q)}_{in}} \approx \alpha - \beta e^{-(q-\gamma)/\tau}, \qquad (22)$$

where the values of the coefficients are summarized in Table VI. As shown in Fig. 5, the curve defined in (22) is a good approximation of $p_{c^{(q)}_{in}}$ in the LSB-region.

The approximation in (22) can be used for setting the threshold of each adder in the tree adder. As an example, simulations show that if one sets the threshold to the highest probability in the LSB-region ($T = p_{c^{(\tilde{q})}_{in}}$), which corresponds to $q^* = \tilde{q}$, then there is a PSNR loss of up to 5 dB. As a consequence, we set $q^* = \tilde{q} - \delta < \tilde{q}$ with $\delta > 0$. It is worth pointing out that, since we force $c^{(q)}_{in} =$ '0' for $0 \leq q \leq q^*$, the corresponding bits of $s = a + b$ are not correct. However, in JPEG2000 the results of the DWT are quantized, so it is unnecessary to compensate the

Table VIII: Coefficients for the analysis 10/18 filters.

j	$h[j]$	$\phi(j)g[j]$
0,-1	5.366288017916415e-001	-4.407818292932527e-001
1,-2	5.429907539425682e-002	1.155190028604326e-001
2,-3	-1.113880188246157e-001	6.057160715369129e-002
3,-4	5.829726464040216e-005	-9.733420187993370e-003
4,-5	2.040184437407670e-002	-2.180274267317332e-002
5,-6		-1.787592313637589e-003
6,-7		6.683900685043967e-003
7,-8		-1.928418995893536e-006
8,-9		-6.748739325063276e-004

approximation caused by result-biasing, as discussed in the next paragraphs. Thus, to save complexity we approximate $s^{(q)} \approx a^{(q)}$ for $0 \leq q \leq q^*$. Through extensive simulations we found the values for δ (see Table VI) that minimize the PSNR loss. These values lead to the results detailed in Table VII as $\Delta\text{PSNR}_{\text{BB+TB1}} = \text{PSNR}_{\text{DA}} - \text{PSNR}_{\text{BB+TB1}}$, where $\text{PSNR}_{\text{BB+TB1}}$ is the PSNR obtained by performing result-biasing both in the butterfly circuit and the first level of the adder tree. From the complexity point of view, the number of FAs required for the six adders at the first level of the tree adder decreases from 96 to 57 (39 FAs saved).

The approach used for the first level of adders in the tree adder is applied to the other levels as well and the value of each parameter is summarized in Table VI. As it can be observed, the performance loss caused by result-biasing at the second and third level of adders in the tree adder ($\Delta\text{PSNR}_{\text{BB+TB1/2}}$ and $\Delta\text{PSNR}_{\text{BB+TB1/2/3}}$, respectively) is nearly the same as $\Delta\text{PSNR}_{\text{BB+TB1}}$, where $\Delta\text{PSNR}_{\text{BB+TB1/2}} = \text{PSNR}_{\text{DA}} - \text{PSNR}_{\text{BB+TB1/2}}$, $\Delta\text{PSNR}_{\text{BB+TB1/2/3}} = \text{PSNR}_{\text{DA}} - \text{PSNR}_{\text{BB+TB1/2/3}}$ and $\text{PSNR}_{\text{BB+TB1/2}}$, $\text{PSNR}_{\text{BB+TB1/2/3}}$ are the PSNR values obtained by introducing result-biasing in the butterfly circuit and at the first, second (BB + TB1/2) and first, second and third (BB + TB1/2/3) levels in the tree adder. When the

Table IX: Values of the $\mathbf{h}_{13,10}$ and $\mathbf{g}_{13,18}$ coefficients and corresponding $\underline{d}_j$ vectors for the 10/18 wavelet filters.

$-r$	$h[4]$	$h[3]$	$h[2]$	$h[1]$	$h[0]$	$\underline{d}_j$	$g[8]$	$g[7]$	$g[6]$	$g[5]$	$g[4]$	$g[3]$	$g[2]$	$g[1]$	$g[0]$	$\underline{d}_j$
0	0	0	1	0	0	$\underline{d}_0$	0	0	1	0	0	0	1	1	0	$\underline{d}_7$
1	0	0	1	0	1	$\underline{d}_1$	0	0	1	0	0	0	1	1	0	$\underline{d}_7$
2	0	0	1	0	0	$\underline{d}_0$	0	0	1	0	0	0	1	1	1	$\underline{d}_8$
3	0	0	1	0	0	$\underline{d}_0$	0	0	1	0	0	0	1	1	1	$\underline{d}_8$
4	0	0	0	0	0	**0**	0	0	1	0	0	0	1	0	1	$\underline{d}_9$
5	0	0	0	1	1	$\underline{d}_2$	0	0	1	0	0	0	0	0	0	$\underline{d}_{10}$
6	1	0	0	1	0	$\underline{d}_3$	0	0	1	0	1	0	0	0	0	$\underline{d}_{11}$
7	0	0	1	0	0	$\underline{d}_0$	0	0	1	0	0	1	0	1	0	$\underline{d}_{12}$
8	1	0	1	1	1	$\underline{d}_5$	0	0	0	0	1	0	0	0	0	$\underline{d}_{13}$
9	0	0	1	1	0	$\underline{d}_6$	0	0	0	0	1	1	1	0	1	$\underline{d}_{14}$
10	1	0	0	1	1	$\underline{d}_4$	0	0	1	1	0	0	0	1	1	$\underline{d}_{15}$
11	0	0	0	1	1	$\underline{d}_2$	1	0	0	1	0	0	0	1	0	$\underline{d}_{16}$
12	0	0	0	0	0	**0**	1	0	1	1	1	0	0	1	1	$\underline{d}_{17}$

Table X: Values of $\dot{\mathbf{hg}}_{13,18}$ and $\dot{\mathbf{gh}}_{13,18}$ coefficients.

	$-r$ \ j	8	7	6	5	4	3	2	1	0	-1	-2	-3	-4	-5	-6	-7	-8	-9
$\dot{\mathbf{hg}}_{13,18}$	0	1	0	0	0	1	0	0	1	1	1	1	0	0	1	0	0	0	0
	1	1	0	0	0	1	0	0	1	1	0	1	0	0	1	0	0	0	0
	2	1	0	0	0	1	0	0	1	0	1	1	0	0	1	0	0	0	0
	3	1	0	0	0	1	0	0	1	0	1	1	0	0	1	0	0	0	0
	4	1	0	0	0	1	0	0	1	0	1	0	1	0	1	0	0	0	0
	5	1	0	0	0	1	0	1	0	1	0	0	1	0	1	0	0	0	0
	6	1	0	0	0	0	0	1	0	1	1	0	1	0	0	0	0	0	0
	7	1	0	0	0	1	0	1	1	1	1	1	0	1	1	0	0	0	0
	8	1	0	1	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0
	9	1	0	1	0	0	0	1	0	0	1	0	1	1	1	0	0	0	0
	10	1	0	0	0	1	0	0	0	0	0	1	0	0	1	1	0	0	0
	11	0	0	1	0	1	0	0	1	1	1	1	0	0	0	1	0	0	0
	12	1	0	1	0	1	0	0	0	1	0	1	0	0	0	1	0	0	0
$\dot{\mathbf{gh}}_{13,18}$	0	0	0	0	0	0	0	1	1	0	1	0	0	0	1	0	0	0	1
	1	0	0	0	0	0	0	1	1	1	1	0	0	0	1	0	0	0	1
	2	0	0	0	0	0	0	1	1	0	0	0	0	0	1	0	0	0	1
	3	0	0	0	0	0	0	1	1	0	0	0	0	0	1	0	0	0	1
	4	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1
	5	0	0	0	0	0	0	0	0	1	1	1	1	0	1	0	0	0	1
	6	0	0	0	0	1	0	0	0	0	1	1	1	0	0	0	0	0	1
	7	0	0	0	0	0	0	1	1	1	0	1	0	1	0	1	0	0	1
	8	0	0	0	0	1	0	1	0	1	1	1	1	0	0	1	0	1	1
	9	0	0	0	0	0	1	1	0	0	0	1	1	0	0	0	1	0	1
	10	0	0	0	1	1	0	0	1	1	0	1	0	0	1	0	0	0	1
	11	0	0	0	1	0	0	0	1	1	1	1	0	0	1	0	1	0	0
	12	0	0	0	1	0	0	0	1	0	1	0	0	0	1	0	1	0	1

result-biasing technique is applied at the second and third level of the tree adder, the number of required FAs decreases from 48 to 36 (12 FAs saved) and from 32 to 26 (6 FAs saved), respectively. Thus, the proposed result-biased tree adder saves 57 FAs on levels 1 to 3 of the tree adder. Further experiments have shown that there is no advantage in implementing result-biasing in the adder at the fourth level.

C. DA-based architecture for the 10/18 wavelet filters

As shown in Table VIII, the 10/18 wavelet filters are symmetric. This property can be exploited to design a reduced complexity architecture. However, since in this case k and l are even values, we introduce

$$\phi(j) = \begin{cases} 1 & j \geq 0 \\ -1 & j < 0 \end{cases} \qquad (23)$$

and derive the common-term-vectors $\underline{d}_j$, which are summarized in Table IX. The corresponding butterfly circuit is depicted in Fig. 6, where $C_\omega = x[i + \omega + 1] \pm x[i - \omega]$, $\omega = 1,\dots,8$ and the Hi_Lo_n signal is used to add or subtract input samples in the low-pass and high-pass filter implementations, respectively. Then, as for the 9/7 case, the architecture relies on a hard-wired shift network and a tree adder.

On the other hand, it is not possible to exploit the symmetry of the filters in the implementation of the architecture for synthesis filters as $\dot{k}$ and $\dot{l}$ are even values. Indeed, since $\dot{\mathbf{hg}}_{13,10}$ and $\dot{\mathbf{gh}}_{13,18}$ are non-symmetric matrices, $\underline{\dot{C}}$ can not be defined. As a consequence, there is no simplification in (6) and (8) and these equations are implemented as summarized in Table X. Unfortunately, the content of $\dot{\mathbf{hg}}_{13,10}$ and $\dot{\mathbf{gh}}_{13,18}$ shows only partial common-term-vectors, thus the DA approach is more effective for the analysis filters than for the synthesis ones.

D. Result-biased DA-based architecture for the 10/18 analysis wavelet filters

1) Implementation of the result-biased butterfly circuit: In order to trim the result-biasing for the butterfly circuit, we

Table XI: Values of $\underline{d}_j$ vectors and corresponding $\mathcal{I}_j$ sets for the 10/18 wavelet filters.

$\underline{d}_j$	$\mathcal{I}_j$	$\min_r\{\mathcal{I}_j\}$
$\underline{d}_0$	$\mathcal{I}_0 = \{u^{(0)}, u^{(-2)}, u^{(-3)}, u^{(-7)}\}$	0
$\underline{d}_1$	$\mathcal{I}_1 = \{u^{(-1)}\}$	1
$\underline{d}_2$	$\mathcal{I}_2 = \{u^{(-5)}, u^{(-11)}\}$	5
$\underline{d}_3$	$\mathcal{I}_3 = \{u^{(-6)}\}$	6
$\underline{d}_4$	$\mathcal{I}_4 = \{u^{(-10)}\}$	10
$\underline{d}_5$	$\mathcal{I}_5 = \{u^{(-8)}\}$	8
$\underline{d}_6$	$\mathcal{I}_6 = \{u^{(-9)}\}$	9
$\underline{d}_7$	$\mathcal{I}_7 = \{v^{(0)}, v^{(-1)}\}$	0
$\underline{d}_8$	$\mathcal{I}_8 = \{v^{(-2)}, v^{(-3)}\}$	2
$\underline{d}_9$	$\mathcal{I}_9 = \{v^{(-4)}\}$	4
$\underline{d}_{10}$	$\mathcal{I}_{10} = \{v^{(-5)}\}$	5
$\underline{d}_{11}$	$\mathcal{I}_{11} = \{v^{(-6)}\}$	6
$\underline{d}_{12}$	$\mathcal{I}_{12} = \{v^{(-7)},\}$	7
$\underline{d}_{13}$	$\mathcal{I}_{12} = \{v^{(-8)},\}$	8
$\underline{d}_{14}$	$\mathcal{I}_{12} = \{v^{(-9)},\}$	9
$\underline{d}_{15}$	$\mathcal{I}_{12} = \{v^{(-10)},\}$	10
$\underline{d}_{16}$	$\mathcal{I}_{12} = \{v^{(-11)},\}$	11
$\underline{d}_{17}$	$\mathcal{I}_{12} = \{v^{(-12)},\}$	12

built $\mathcal{I}_j$, the sets defined by each of the $\underline{d}_j \cdot \underline{C}$ products, as shown in Table XI. Then, we set $\hat{q}_j = \min_r\{\mathcal{I}_j\} - 1$ (as for the 9/7 filters). In this case, the proposed result-biased butterfly circuit saves 113 FAs. Since $m = 16$, the standard butterfly circuit requires 448 FAs, whereas the proposed one requires 448-113=335 FAs.
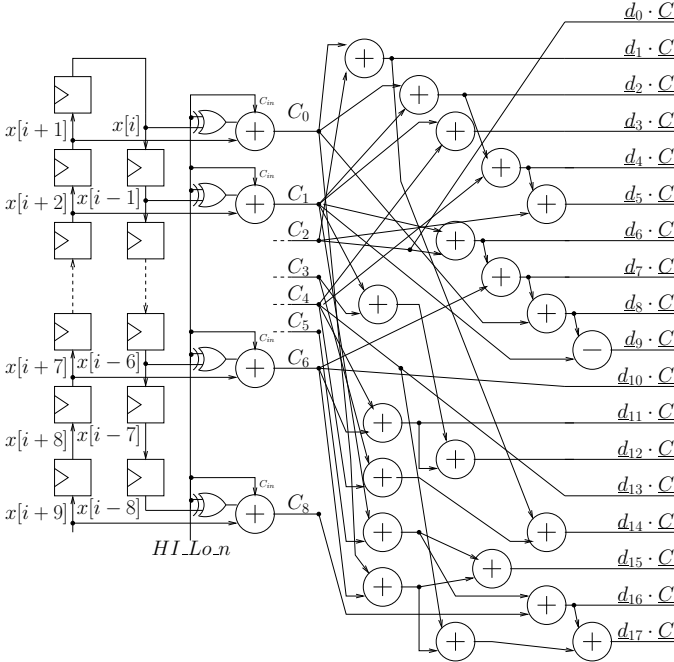

Figure 6: Butterfly circuit for the 10/18 filters.

2) Result-biased tree adder implementation: The architecture of the result-biased tree adder is nearly the same one employed for the 9/7 filters and described in Section IV-B2. The only differences with respect to the circuit shown in Fig. 4 (b) are: i) the hard-wired shift network, where the inputs to the multiplexers are the ones summarized in Table IX, ii) the -1 multiplication at the output. Since we observed similar carry signal probabilities for both 9/7 and 10/18

filters, the same result-biasing strategy has been employed for the implementation of the tree adder. In particular, the approximation in (22) with the parameters shown in Table VI, has been exploited. As a consequence, the number of saved FAs is the same one obtained for the 9/7 filters, namely 57 FAs. Experimental results, achieved by implementing result-biasing in the tree adder, are shown in Table XII. As it can be observed, the performance loss in terms of PSNR ($\Delta\text{PSNR}_{\text{BB+TB1/2/3}} = \text{PSNR}_{\text{DA}} - \text{PSNR}_{\text{BB+TB1/2/3}}$) of the proposed result-biased variant, with respect to the original DA-based architecture, is limited to few fractions of dB, where PSNR_{DA} and $\text{PSNR}_{\text{BB+TB1/2/3}}$ are the PSNRs of the original and proposed solution respectively.

V. HARDWARE IMPLEMENTATION AND COMPARISON

The proposed result-biased architectures for the computation of the 9/7 and 10/18 wavelet filters have been implemented using a 90 nm standard cell technology library for a 200 MHz target clock frequency, leading to areas of 7621 μm^2 and 12602 μm^2, which correspond to about 2.7 and 4.5 equivalent kgates for the 9/7 and 10/18 filters, respectively. Moreover, with the same technology the proposed architectures can achieve maximum clock frequencies of 450 MHz and 360 MHz with areas of 8087 μm^2 (2.85 eq. kgates) and 13845 (4.91 eq. kgates) for the 9/7 and 10/18 filters, respectively. These results are shown in Table XIII, where the proposed architectures are compared with other solutions available in the literature in terms of PSNR, number of FAs, number of Flip-Flops (FFs), clock frequency (f_{clk}) and area.

The proposed architecture for the 9/7 filters offers a relevant complexity reduction with respect to previously published DA-based implementations for DWT computation [19], [20], with a very small PSNR loss (see Tables V and VII). When compared with multiplierless solutions, which were specifically optimized for the 9/7 wavelet filters, the proposed architecture shows a PSNR loss of few fractions of dB as the variants described in [6], [7], [13], [30]. To enable complexity comparison of the proposed architecture with the other works, in particular with the ones described in [13] and [9] for the 9/7 and 10/18 filters, we introduced the normalized area A_n=Area·(90/Tech)^2 (last column of Table XIII), where Tech is the technology process used for the implementation, namely 90 nm for the architectures proposed in this work, 45 nm for [13] and 130 nm for [9].

From the data in Table XIII we observe that the proposed architecture for the 9/7 filters features almost the same complexity as the lowest complexity implementations, i.e. [7], [30]. Comparison with [13] in terms of circuit speed is not straightforward as it relies on a technology more scaled than the one employed in this work. As a consequence, [13] features higher maximum clock frequency than our implementation. Beside, the architecture in [13] requires more FFs but less FAs than the result-biased DA-based variant, leading to a slightly higher normalized area than the proposed solution. It is worth noting that all the considered architectures for the 9/7 filters have a throughput of one sample per clock cycle. Furthermore, Table XIII shows that the area of DA-based architectures

Table XII: PSNR comparison between the DA-based DWT (column $\mathrm{PSNR_{DA}}$) and the proposed DA-based DWT with result-biasing applied to the butterfly circuit and to the tree adder (first, second and third level) (column $\Delta\mathrm{PSNR_{BB+TB1/2/3}} = \mathrm{PSNR_{DA}} - \mathrm{PSNR_{BB+TB1/2/3}}$) for the 10/18 filters.

Image	L	PSNR$_{DA}$ [dB]					ΔPSNR$_{BB+TB1/2/3}$ [dB]				
		1:1	8:1	16:1	32:1	64:1	1:1	8:1	16:1	32:1	64:1
Lena	1	49.35	39.46	34.77	29.89	22.97	0.15	0.00	-0.04	0.01	0.03
	2	49.00	40.49	37.18	33.64	30.07	0.40	0.07	0.01	0.06	0.00
	3	48.92	40.66	37.52	34.34	31.21	0.83	0.14	0.07	0.03	-0.01
	4	48.87	40.67	37.54	34.44	31.34	1.34	0.23	0.12	0.07	0.01
Barbara	1	49.43	36.92	30.08	24.39	20.60	0.11	-0.01	-0.02	0.00	0.00
	2	49.14	38.50	33.06	28.44	24.47	0.41	0.02	0.02	0.00	0.01
	3	49.07	38.74	33.55	29.28	25.70	0.84	0.08	0.03	0.04	0.00
	4	49.02	38.72	33.61	29.34	26.19	1.35	0.11	0.04	0.02	0.05
Boat	1	49.37	38.23	33.13	28.01	21.61	0.11	0.03	-0.03	0.00	-0.15
	2	49.04	39.40	34.52	30.64	27.40	0.41	0.02	0.01	0.19	0.09
	3	48.97	39.57	34.90	31.16	28.14	0.82	0.10	0.00	0.02	0.01
	4	48.92	39.59	34.95	31.22	28.17	1.35	0.22	0.04	0.06	0.02
Goldhill	1	49.75	36.06	31.93	27.70	22.90	0.17	0.00	-0.02	0.02	0.00
	2	49.51	36.68	33.17	30.32	27.93	0.50	0.02	0.06	-0.03	0.29
	3	49.46	36.75	33.34	30.62	28.54	0.97	0.05	0.07	0.01	0.01
	4	49.42	36.78	33.35	30.62	28.57	1.53	0.11	0.03	-0.04	0.02
Fingerprint	1	49.57	36.24	31.92	26.91	15.86	0.17	0.00	0.00	0.01	-0.20
	2	49.39	36.60	32.74	29.58	26.21	0.47	0.02	0.02	0.01	0.00
	3	49.36	36.67	32.80	29.73	27.15	0.94	0.05	0.04	0.01	0.02
	4	49.33	36.68	32.82	29.80	27.19	1.51	0.08	0.03	-0.01	0.01

Table XIII: Architecture comparison in terms of performance (PSNR), complexity (FAs, FFs, Eq. kgate, Area, A_n), technology (Tech) and speed (f_{clk}).

Filter	DA	Arch.	ΔPSNR	FAs	FFs	Eq. kgate	Tech [nm]	f_{clk} [MHz]	Area μm^2	A_n μm^2
9/7	N	[6]	Table VII, ΔPSNR$_{BB+TB1/2/3}$	512	144	-	130	200	-	-
		[7]	Table VII, ΔPSNR$_{BB+TB1/2/3}$	336	144	2.81	130	200	-	-
		[13]	Table VII, ΔPSNR$_{BB+TB1/2/3}$	192	213	-	45	500	2135	8540
		[30]	Table VII, ΔPSNR$_{BB+TB1/2/3}$	304	144	2.69	130	200	-	-
	Y	[19]	0 dB	688	144	5.39	130	200	-	-
		[20]	0 dB	432	144	4.17	130	200	-	-
		Prop.	Table VII, ΔPSNR$_{BB+TB1/2/3}$	320	144	2.71/2.85	90	200/450	7621/8087	7621/8087
10/18	N	[8]	0 dB	640[a]	432	23.16	250	78	-	-
		[9]	0 dB	832[a]	464	11.27	130	200	67612	32406
	Y	-[b]	0 dB	640	144	5.62/6.22	90	200/365	15868/17559	15868/17559
		Prop.	Table XII, ΔPSNR$_{BB+TB1/2/3}$	470	144	4.47/4.91	90	200/366	12602/13845	12602/13845

[a] The architecture contains also multipliers.
[b] Since no reference is available in the literature it has been implemented.

for the 10/18 filters is from 39% (result-biased DA-based architecture) to 49% (DA-based architecture) the area of other optimized variants based on B-spline factorization, such as [8], [9]. Even if the architectures in [8], [9] have a throughput of two samples per clock cycle, the low area required by DA-based implementations makes them superior in terms of throughput to area ratio. Finally, the proposed result-biasing technique reduces the complexity of the architecture for the 10/18 DWT computation as well as for the 9/7 one. These figures of merit highlight the effectiveness of the proposed result-biasing technique as a general method to reduce the complexity of DA-based architectures for the approximate computation of the DWT.

VI. CONCLUSIONS

In this work a result-biased DA-based filter architecture for the approximate computation of the DWT has been presented. The proposed idea has been applied to the well known 9/7 and 10/18 wavelet filters, respectively, to reduce the complexity of DA-based architectures for the DWT computation, with a very

small loss in terms of PSNR. Experimental results show that i) the proposed technique is effective in reducing the complexity of DA-based architectures for the DWT computation; ii) the performance and complexity of the variant derived for the 9/7 filters are comparable with the ones of previously proposed architectures, which are specifically optimized for the 9/7 wavelet filters; iii) the performance and complexity of the proposed architecture for the 10/18 wavelet filters are better than those of previously published works.

REFERENCES

[1] M. Boliek, "JPEG 2000 Final Committee Draft," 2000.
[2] A. Biligin and M. W. Marcellin, "JPEG2000 for digital cinema," in *IEEE International Conference on Circuits and Systems*, 2006.
[3] B. K. Mohanty and P. K. Meher, "Memory-efficient high-speed convolution-based generic structure for multilevel 2-D DWT," *IEEE Tran. on Circuits and Systems for Video Technology*, vol. 23, no. 2, pp. 353–363, Feb 2013.
[4] Y. Hu and C. C. Jong, "A memory-efficient high-throughput architecture for lifting-based multi-level 2-D DWT," *IEEE Tran. on Signal Processing*, vol. 61, no. 20, pp. 4975–4987, Oct 2013.
[5] G. Strang and T. Q. Nguyen, *Wavelets and Filter Banks*. Wellesley-Cambridge, MA: Wellesley, 1996.

[6] K. A. Kotteri, A. E. Bell, and J. E. Carletta, "Design of multiplierless, high-performace, wavelet filter banks with image compression applications," *IEEE Tran. on Circuits and Systems-I*, vol. 51, no. 3, pp. 483–494, Mar. 2004.

[7] M. Martina and G. Masera, "Low-complexity, efficient 9/7 wavelet filters VLSI implementation," *IEEE Tran. on Circuits and Systems-II*, vol. 53, no. 11, pp. 1289–1293, Nov 2006.

[8] C. T. Huang, P. C. Tseng, and L. G. Chen, "VLSI architecture for forward discrete wavelet transform based on B-spline factorization," *Journal of VLSI Signal Processing*, vol. 40, no. 3, pp. 343–353, Jul. 2005.

[9] M. Martina, G. Masera, and G. Piccinini, "Scalable low-complexity B-spline discrete wavelet transform architecture," *IET Circuits, Devices and Systems*, vol. 4, no. 2, pp. 159–167, Feb 2010.

[10] M. A. Islam and K. A. Wahid, "Area- and power-efficient design of Daubechies wavelet transforms using folded AIQ mapping," *IEEE Tran on Circuits and Systems-II*, vol. 57, no. 9, pp. 716–720, Sep 2010.

[11] S. K. Madishetty, A. Madanayake, R. J. Cintra, and V. S. Dimitrov, "Precise VLSI architecture for AI based 1-D/ 2-D Daub-6 wavelet filter banks with low adder-count," *IEEE Tran. on Circuits and Systems-I*, to appear.

[12] S. Murugesan and D. B. H. Tay, "New techniques for rationalizing orthogonal and biorthogonal wavelet filter coefficients," *IEEE Tran. on Circuits and Systems-I*, vol. 59, no. 3, pp. 628–637, Mar 2012.

[13] A. Pande and J. Zambreno, "Poly-DWT: Polymorphic wavelet hardware support for dynamic image compression," *ACM Tran. on Embedded Computing Systems*, vol. 11, no. 1, pp. 1–26, Mar 2012.

[14] A. K. Naik and R. S. Holambe, "Design of low-complexity high-performance wavelet filters for image analysis," *IEEE Tran. on Image Processing*, vol. 22, no. 5, pp. 1848–1858, May 2013.

[15] S. Y. Park and P. K. Meher, "Low-power, high-throughput, and low-area adaptive FIR filter based on distributed arithmetic," *IEEE Tran. on Circuits and Systems-II*, vol. 60, no. 6, pp. 346–350, Jun 2013.

[16] M. S. Prakash and R. A. Shaik, "Low-area and high-throughput architecture for an adaptive filter using distributed arithmetic," *IEEE Tran. on Circuits and Systems-II*, vol. 60, no. 11, pp. 781–785, Nov 2013.

[17] J. Xie, P. K. Meher, and J. He, "Hardware-efficient realization of prime-length DCT based on distributed arithmetic," *IEEE Tran. on Computers*, vol. 62, no. 6, pp. 1170–1178, Jun 2013.

[18] Y. H. Chen, J. N. Chen, T. Y. Chang, and C. W. Lu, "High-throughput multistandard transform core supporting MPEG/H.264/VC-1 using common sharing distributed arithmetic," *IEEE Tran. on VLSI Systems*, vol. 22, no. 3, pp. 463–474, Mar 2014.

[19] M. Alam, C. Rahman, W. Badawy, and G. Jullien, "Efficient distributed arithmetic based DWT architecture for multimedia applications," in *IEEE International Workshop on System-on-Chip for Real-Time Applications, Calgari, 30 June - 2 July*, 2003, pp. 333–336.

[20] X. Cao, Q. Xie, C. Peng, Q. Wang, and D. Yu, "An efficient VLSI implementation of distributed architecture for DWT," in *IEEE Workshop on Multimedia Signal Processing*, 2006, pp. 364–367.

[21] S. S. Kidambi, F. El-Guibaly, and A. Antoniou, "Area-efficient multipliers for digital signal processing applications," *IEEE Tran. on Circuits and Systems-II*, vol. 43, no. 2, pp. 90–95, Feb. 1996.

[22] K. J. Cho, K. C. Lee, J. G. Chung, and K. K. Parhi, "Design of low-error fixed-width modified Booth multiplier," *IEEE Tran. on VLSI Systems*, vol. 12, no. 5, pp. 522–531, May 2004.

[23] N. Petra, D. De Caro, V. Garofalo, E. Napoli, and A. G. M. Strollo, "Design of fixed-width multipliers with linear compensation function," *IEEE Tran. on Circuits and Systems-I*, vol. 58, no. 5, pp. 947–960, May 2011.

[24] D. De Caro, N. Petra, A. G. M. Strollo, F. Tessitore, and E. Napoli, "Fixed-width multipliers and multipliers-accumulators with min-max approximation error," *IEEE Tran. on Circuits and Systems-I*, vol. 60, no. 9, pp. 2375–2388, Sep 2013.

[25] M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies, "Image coding using the wavelet transform," *IEEE Tran. on Image Processing*, vol. 1, no. 2, pp. 205–220, Apr. 1992.

[26] "http://www.openjpeg.org."

[27] ISO/IEC 15444-1:2004/FDAM 1, "Information technology - JPEG 2000 image coding system: Core coding system, amendment 1: profiles for digital cinema applications," ISO/IEC, Tech. Rep., 2004.

[28] M. Martina, "Low Complexity 9/7 Wavelet: Modified OpenJPEG model," downloadable at http://personal.delen.polito.it/maurizio.martina/wavelet.html.

[29] A. G. Dempster and M. D. Macleod, "Use of minimum-adder multiplier blocks in FIR digital filters," *IEEE Tran. on Circuits and Systems-II*, vol. 42, no. 9, pp. 569–577, Sep. 1995.

[30] M. Martina and G. Masera, "Multiplierless, folded 9/7-5/3 wavelet VLSI architecture," *IEEE Tran. on Circuits and Systems-II*, vol. 54, no. 9, pp. 770–774, Sep 2007.

Maurizio Martina (S'98-M'94-SM'15) was born in Pinerolo, Italy, in 1975. He received the M.Sc. and Ph.D. in electrical engineering from Politecnico di Torino, Italy, in 2000 and 2004, respectively. He is currently an Associate Professor of the VLSI-Lab group, Politecnico di Torino. His research activities include VLSI design and implementation of architectures for digital signal processing and communications.

Guido Masera (SM'07) received the Dr. Ing. Degree (summa cum laude) in 1986 and the Ph.D. degree in electronic engineering from the Politecnico di Torino, Torino, Italy, in 1992. Since 1992, he has been an Assistant Professor and then Associate Professor with the Electronic Department, where he is member of the VLSI-Lab group. His research interests include several aspects in the design of digital integrated circuits and systems, with special emphasis on high-performance architecture development and on-chip interconnect modeling and optimization. He is an associate editor of IEEE Transactions on Circuits and Systems II.

Massimo Ruo Roch joined the Department of Electronics, Politecnico di Torino, Turin, Italy, in 1998, where he has been a Full-Time Researcher since 1995. His research interests include digital design of application specific computing architectures, high speed telecommunications, and digital television. Recent activities include design and modeling of MPSoCs, embedded systems for bioapplications, and cloud-based systems for e-learning.

Gianluca Piccinini received the Dr. Ing. and the Ph.D. degrees in electronics engineering, in 1986 and 1990, respectively. He is a Full Professor from 2006 at the Department of Electronics, Politecnico di Torino, Torino, Italy. His current research interest includes the use of nanotechnologies in integrated systems, and he is working on molecular transport for beyond CMOS structures and on molecules interaction in molecular QCA.