Hindawi
Advances in Multimedia
Volume 2020, Article ID 8823689, 13 pages
https://doi.org/10.1155/2020/8823689

Research Article

Lifting-Based Fractional Wavelet Filter: Energy-Efficient DWT Architecture for Low-Cost Wearable Sensors

Mohd Tausif,[1] **Ekram Khan**,[2] **Mohd Hasan**,[2] **and Martin Reisslein**[3]

[1]*Faculdade de Engenharia, Departamento de Informática, Universidade da Beira Interior, Covilhã, Portugal*
[2]*Department of Electronic Engineering, Zakir Husain College of Engineering & Technology, Aligarh Muslim University, Aligarh 202002, India*
[3]*School of Electrical Computer and Energy Engineering, Arizona State University, Goldwater Center, East Tyler Mall 650, MC 5706, Tempe, AZ 85287-5706, USA*

Correspondence should be addressed to Martin Reisslein; reisslein@asu.edu

Received 20 April 2020; Revised 18 November 2020; Accepted 28 November 2020; Published 16 December 2020

Academic Editor: Constantine Kotropoulos

This paper proposes and evaluates the LFrWF, a novel lifting-based architecture to compute the discrete wavelet transform (DWT) of images using the fractional wavelet filter (FrWF). In order to reduce the memory requirement of the proposed architecture, only one image line is read into a buffer at a time. Aside from an LFrWF version with multipliers, i.e., the LFrWF_m, we develop a multiplier-less LFrWF version, i.e., the LFrWF_ml, which reduces the critical path delay (CPD) to the delay T_a of an adder. The proposed LFrWF_m and LFrWF_ml architectures are compared in terms of the required adders, multipliers, memory, and critical path delay with state-of-the-art DWT architectures. Moreover, the proposed LFrWF_m and LFrWF_ml architectures, along with the state-of-the-art FrWF architectures (with multipliers (FrWF_m) and without multipliers (FrWF_ml)) are compared through implementation on the same FPGA board. The LFrWF_m requires 22% less look-up tables (LUT), 34% less flip-flops (FF), and 50% less compute cycles (CC) and consumes 65% less energy than the FrWF_m. Also, the proposed LFrWF_ml architecture requires 50% less CC and consumes 43% less energy than the FrWF_ml. Thus, the proposed LFrWF_m and LFrWF_ml architectures appear suitable for computing the DWT of images on wearable sensors.

1. Introduction

1.1. Motivation. The availability of low-cost small-sized cameras attached to wearable sensors and portable imaging devices has opened up a wide range of imaging-oriented applications, including assisted living, smart healthcare, traffic monitoring, virtual sports experiences, and posture recognition [1–12]. An interconnection of visual sensor nodes (sensor nodes with attached camera) is known as visual sensor network (VSN) [13, 14] or as wireless multimedia sensor network (WMSN) [15, 16]. Wearable visual sensors may also be a part of the Internet of things (IoT) [17–21]. Low-cost IoT wearable sensors [22] enable a wide range of activities for the benefit of society, e.g., hazard avoidance systems for worker safety [23], navigation aids for visually impaired individuals [24], activity monitoring [25], smart irrigation [26], and sports [27].

In many visual applications of wearable sensors and portable imaging devices, images captured by the camera need to be transmitted wirelessly to a body-worn or nearby hub device. The wearable sensors and portable imaging devices have limited resources, and the wireless links have narrow bandwidth [28], making it impossible to directly send the raw (uncompressed) images. Thus, there is a need to compress the images before transmission [29]. Therefore, an image coder is needed in order to compress the images. In an image coder, an image is generally first transformed using the discrete cosine transform (DCT) [30] or discrete wavelet transform (DWT) [31, 32] and then it is quantized and entropy coded. The DWT, which is also used in JPEG 2000

[33], is popular in a wide variety of applications, including activity monitoring [34], fault detection in inverter circuits [35], medical imaging [36], image denoising [37], image recognition [38], image reconstruction [39], watermarking [40], computer graphics, and real-time processing [41] due to its multiresolution feature and excellent energy compaction properties [42, 43].

The hardware architectures for wearable visual sensors and portable imaging devices in the IoT and wireless multimedia sensor networks should require minimal hardware resources and consume low energy for a small form factor and long battery life [44, 45]. Generally, the computational capabilities of visual sensor nodes have been increasing in recent years [46]. Nevertheless, due to the economic pressures on visual sensor designs and despite the emergence of specialized hardware acceleration, e.g., FPGA and, components [47–49], the computational resources of visual sensors will likely remain scarce. Emerging computing and communication paradigms, such as mobile ad hoc cloud computing [50, 51], expect the nodes to not only transmit sensed images but also to participate in some service computing functions, e.g., for localized image analysis and decision making, which can be orchestrated through software-defined networking and control structures [52–54]. In order to make the economical functioning of wearable visual sensors in such networked systems feasible, the resource usage of the image coding and transform must be very low. In particular, as the DWT is an important component of an image coder for visual sensors, the DWT hardware architecture should have minimal area and energy consumption.

1.2. Related Work. The conventional convolution-based DWT computation of an image requires a huge amount of memory due to its row- and column-wise scanning [55, 56], making it unsuitable for memory-constrained wearable sensors. The different low-memory architectures reported in the literature for computation of DWT can be categorized as line-based architectures [57], stripe-based architectures [58, 59], block-based architectures [60, 61], and the fractional wavelet filter (FrWF) architecture [62]. For an image of dimension $J \times J$ pixels, the line, stripe, and block-based architectures require random access memory (which we refer to as RAM or memory for brevity) in the range of $3J$ to $5.5J$ words, while the FrWF architecture requires $2J + 22$ words of RAM [62].

Another low-memory pipeline-based architecture has been proposed in [63]. However, the design in [63] is based on the nonseparable DWT computation approach, which is unpopular because of its higher computational requirements than the conventional separable approach. It is a well-known fact that at a particular throughput, the separable 2D DWT computation approach is computationally more efficient than the nonseparable approach [64]. A dual data scanning-based DWT architecture is reported in [65]. In this architecture, several 2D DWT units are combined into a parallel multilevel architecture for computing up to six DWT levels. However, this architecture needs $13J$ words of memory. An architecture based on an interlaced read scan algorithm (IRSA) is proposed in [66] in conjunction with a lifting-based approach with a 5/3 filter-bank which requires $2J$ words of memory. However, the long critical path delay (CPD) of $2T_m + 4T_a$ (where T_m is the multiplier delay and T_a is the adder delay) of the architecture in [66] may limit its use in real-time applications.

An LUT-based lifting architecture for computing the DWT has been reported in [67]. The design [67] has low area and power requirements. However, it has a long CPD equal to $T_{\mathrm{LUT}} + (W/4 - 1)T_{\mathrm{FA}} + 2T_a$ (where T_{LUT} is the look-up table (LUT) delay, $W = 16$ bits is the word length, and T_{FA} is the full adder delay). A lifting-based architecture for computing both the 1D and 2D DWT has been presented in [68]. However, this design uses a transpose buffer of size J^2. An energy-efficient block-based DWT architecture has been proposed in [61]. However, this architecture requires a large number of multipliers, namely, 16 and 36 multipliers for 5/3 and 9/7 filters, respectively. Another energy-efficient lifting-based reconfigurable DWT architecture has been proposed in [69], mainly for medical applications. However, the frequency of operation of this architecture is limited to 20 MHz. An energy-efficient lifting-based configurable DWT architecture for neural sensing applications has been proposed in [70], requiring 12 adders and 12 multipliers. However, its operating frequency is limited to only 400 KHz and 80 KHz for the gating and interleaving architectures used in the main architecture, respectively.

A power-efficient modified form of the DWT architecture has been presented in [71], using Radix-8 booth multipliers. This architecture uses bit truncation to reduce the area and power. However, bit truncation degrades the quality of the reconstructed image when the inverse DWT is applied. There have been some DWT implementations on graphics processing units (GPUs) [72–78]; however, GPUs are relatively expensive for low-cost sensing platforms.

The recently proposed FrWF architecture requires only $2J + 22$ words of memory and has a CPD equal to the delay of a multiplier T_m [62]. A multiplier-less FrWF architecture was also reported in [62] which reduces the CPD to the delay of an adder, T_a, $T_a < T_m$. However, the FrWF architecture (with and without multipliers) has high energy consumption owing to its large number of compute cycles. The high energy consumption of the FrWF architecture may be prohibitive for wearable sensors and portable imaging devices with tight memory and energy constraints [79].

1.3. Contributions and Structure of This Article. This paper proposes the $\mathrm{LFrWF_m}$, a novel lifting-based energy-efficient architecture to compute the DWT coefficients of an image with a 5/3 filter-bank. At the core of the proposed $\mathrm{LFrWF_m}$ architecture is a novel basic Lift_block that computes the H and L subband coefficients with only two two-input adders and one multiplier (plus two pipeline registers), thus greatly reducing the hardware requirements compared to prior convolution architectures. Moreover, a multiplier-less implementation of the proposed architecture, denoted by $\mathrm{LFrWF_{ml}}$, is designed. The multiplier-less $\mathrm{LFrWF_{ml}}$ has a shorter CPD than the proposed multiplier-based $\mathrm{LFrWF_m}$

architecture. The proposed LFrWF_{m} and LFrWF_{ml} architectures are not only efficient in terms of energy but also require fewer adders, multipliers, and registers than the state-of-the-art FrWF architectures (with multipliers (FrWF_{m}) and without multipliers (FrWF_{ml})). We compare the proposed architectures with state-of-the-art DWT computation architectures in terms of the required adders, multipliers, memory, and critical path delay. We also implement the proposed architectures and the state-of-the-art FrWF architectures on the same FPGA board. Experimental results demonstrate that the proposed LFrWF_{m} and LFrWF_{ml} architectures have lower hardware resource requirements and energy consumption than the state-of-the-art FrWF_{m} and FrWF_{ml} architectures.

The remaining part of the paper is arranged as follows. Section 2 gives a brief overview of the DWT and FrWF techniques. The proposed lifting-based LFrWF architecture is described in detail in Section 3 along with its memory requirement. The evaluation results along with related discussions are presented in Section 4. Finally, Section 5 concludes the paper.

2. Background

This section briefly reviews the DWT and FrWF techniques along with FrWF architecture. The main notations used in this article are summarized in Table 1.

2.1. Discrete Wavelet Transform (DWT). The most popular approach for computing the two-dimensional (2D) DWT of an image is the separable approach, in which the rows are filtered first, followed by column-wise filtering of the resulting coefficients. When a row is convolved (filtered) by a low-pass filter (LPF) and a high-pass filter (HPF), followed by downsampling by a factor of two, the results are known as approximation and detail coefficients, respectively. For a 1D signal of dimension J, which we consider as a preliminary step for computing the 2D DWT, there are $J/2$ approximation coefficients and $J/2$ detail coefficients. Combining the downsampling with the convolution operation, the approximation coefficients $a(i)$ and the detail coefficients $d(i)$ for $i = 0, 1, \ldots, J/2 - 1$ can be expressed mathematically as [55]

$$a(i) = \sum_{j=-\lfloor f_1/2 \rfloor}^{j=\lfloor f_1/2 \rfloor} x_{2i+j} l_j, \quad i = 0, 1, \ldots, \frac{J}{2} - 1,$$
$$d(i) = \sum_{j=-\lfloor f_2/2 \rfloor}^{j=\lfloor f_2/2 \rfloor} x_{2i+j+1} h_j, \quad i = 0, 1, \ldots, \frac{J}{2} - 1, \tag{1}$$

respectively, whereby l_j and h_j denote the j^{th} LPF and HPF coefficient, respectively, x_{2i+j} denotes the $(2i + j)^{\text{th}}$ signal sample, while f_1 and f_2 are the number of LPF and HPF coefficients, respectively. The largest integer less than or equal to x is denoted by the symbol $\lfloor x \rfloor$.

In the separable approach, all image rows are first convolved separately by a HPF and a LPF, followed by downsampling with a factor of two, resulting in the H and L subbands. Then, the columns of the H and L subbands are convolved by a HPF and a LPF, followed by downsampling with a factor of two, resulting in the HH, HL, LH, and LL subbands [80]. However, this approach needs to save the entire $J \times J$ image in the RAM on the sensor (board) system. Thus, this DWT computation approach requires a huge amount of memory, making this approach unsuitable for low-cost wearable sensors and portable imaging devices with limited RAM [55, 56].

Table 1: Summary of main notations.

$J \times J$	Image size in pixels
G	Number of segments per image line
T_m	Delay of a multiplier
T_a	Delay of an adder
f_1	Number of low-pass filter coefficients
f_2	Number of high-pass filter coefficients

The lifting scheme [81] computes the DWT of images using inplace computations which save memory. Moreover, the lifting scheme uses predict and update steps for computing the subbands. In particular, the low-pass filtered coefficients are predicted using the high-pass filtered coefficients. Thus, the lifting scheme reduces the convolution operations needed by the LPF coefficients. Hence, the lifting scheme reduces the number of arithmetic computations required for computing the image DWT [82].

The lifting scheme for a 5/3 filter-bank is shown in Figure 1. In this figure, $x_0, x_1, \ldots, x_6$ are the input signal samples. Among these samples, x_0, x_2, x_4, and x_6 are the even-indexed samples, while x_1, x_3, and x_5 are the odd-indexed samples. Also, α and β are the high-frequency and low-frequency lifting parameters, respectively; G_0 and G_1 are the scaling parameters, whereby $\alpha = -0.5$, $\beta = 0.25$, and $G_0 = G_1 = 1$ [66]; d_0, d_1, and d_2 are the high-frequency wavelet coefficients; while a_0, a_1, a_2, and a_3 are the low-frequency wavelet coefficients. The high- and low-frequency wavelet coefficients are computed following the diagram in Figure 1; for instance,

$$d_0 = [(x_0 + x_2)\alpha + x_1]G_0,$$
$$a_1 = [(d_0 + d_1)\beta + x_2]G_1. \tag{2}$$

It should be noted that the arrows without an associated symbol in Figure 1 have the unit multiplication factor, i.e., 1.

2.2. Fractional Wavelet Filter (FrWF). The FrWF is a low-memory DWT computation technique [56]. It uses a specific image data scanning technique in order to reduce the memory required for computing the DWT. It selects a vertical filter area (VFA), scanning f_1 rows of the image from an SD-card (where f_1 is the number of LPF coefficients). The rows in a VFA are read in raster scan order. Once the reading of all the image rows in a VFA is complete, the VFA is shifted by two lines in the vertical direction. This shifting of the VFA is done in order to incorporate the dyadic downsampling. One line of the HH, HL, LH, and LL subbands is computed from one VFA. All the image lines are covered by shifting the VFA. The VFA will

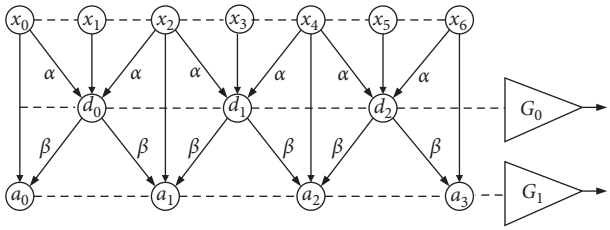

Figure 1: Lifting steps for 5/3 filter-bank [66].

be shifted $J/2$ times for an image of dimension $J \times J$. The FrWF has been combined with a low-memory image coding algorithm to design an efficient image coder for WMSNs in [83].

An FPGA architecture for the FrWF with a 5/3 filter-bank has been proposed in [62]. This FrWF architecture, which follows the FrWF data scanning order, requires $2J + 22$ words of memory and a total of $5J^2/2$ compute cycles. The large number of compute cycles results in a high energy consumption, which may be prohibitive for resource-constrained wearable visual sensors and portable imaging devices. The proposed LFrWF focuses on reducing the energy consumption for computing the DWT of images.

3. Proposed LFrWF Low Energy Architecture

This section presents the proposed LFrWF lifting-based architecture to compute the DWT of an image using the FrWF approach with a 5/3 filter-bank.

3.1. Data Scanning Order. The proposed lifting-based architecture follows the data scanning order of the FrWF algorithm [56]. It is assumed (as is common for low-memory implementations of the DWT computation) that the original image is stored on an SD-card; throughout, the SD accesses are appropriately buffered to compensate for the latencies of the SD-card accesses. Initially, a vertical filter area which spans f_1 image lines (f_1 is the number of LPF coefficients) is marked in the SD-card. The rows of the image are read in raster scan order from the VFA, one line at a time into the RAM buffer P_store (as shown in Figure 2). After the processing of all the rows of the VFA is completed, the VFA is shifted down by two lines and the new rows are again read into buffer P_store in raster scan order. The complete image is read by repeatedly shifting the VFA downwards by two lines until all the rows are read. In the proposed architecture, one complete line is read at a time and scanned in raster order; in contrast, the FrWF architecture in [62] reads only 5 coefficients of an image line at a time.

3.2. Proposed Lifting-Based LFrWF Architecture. This subsection describes the proposed lifting-based DWT architecture in detail.

3.2.1. Top-Level Architecture. Figure 2 shows the top-level block diagram of the proposed LFrWF architecture. The LFrWF architecture works as follows. First, the input image pixels of a line are read into the register P_store. This P_store register

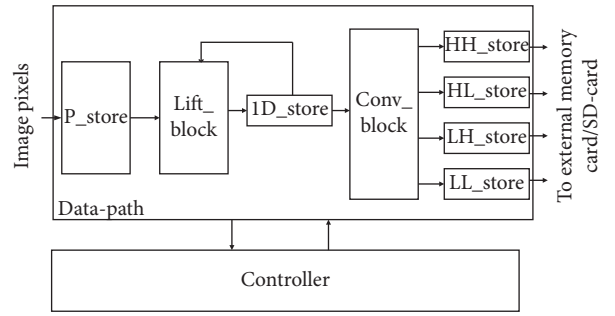

Figure 2: Block diagram of top-level structure of the proposed lifting-based LFrWF architecture; the novel Lift_block is displayed in detail in Figure 3, while the Conv_block is displayed in detail in Figure 4.

stores the original image pixels of 8 bits each. The pixels of the image from P_store are sent to the Lift_block (as detailed in Figure 3) to compute the H and L subband coefficients using the lifting scheme. The generated H and L subband coefficients are saved in the register 1D_store. The contents of the 1D_store register are used as inputs for the Conv_block (as shown in Figure 4), which generates intermediate coefficients that are saved in the HH_store, HL_store, LH_store, and LL_store registers. These intermediate values are successively updated by the next image lines. The intermediate values in the registers HH_store, HL_store, LH_store, and LL_store, after updating, will give the values of the HH, HL, LH, and LL subbands, respectively. Once the final subband coefficients of the HH, HL, LH, and LL subbands are computed, they are transferred and saved in an external SD-card. The functioning of the different blocks leading to the computation of the subbands is described next.

3.2.2. Lifting Block. In the lifting scheme with a 5/3 filter-bank, two previous high-pass filtered coefficients are used to predict a low-pass filtered coefficient. For the efficient implementation of the lifting scheme, we introduce a novel basic Lift_block. As illustrated in Figure 3, the basic Lift_block computes two H subband coefficients and one L subband coefficient from a group of five input pixels in three steps. The inputs (Input_1, Input_2, Input_3, and Lift_{par}) and output (Out_1) of the adders and multiplier to be used in Figure 3 for the different steps are shown in Table 2. The first two steps compute two coefficients of the H subband and the third step computes a coefficient of the L subband. In Table 2, P_0, P_1, P_2, P_3, and P_4 are the first five pixels of an image line. H_0 and H_1 are the first two high-pass filtered coefficients which are stored as the first two elements of the register 1D_store. L_0 is the first low-pass filtered coefficient and is stored as the third element of the register 1D_store. The high-pass filtered coefficients (H_0 and H_1) and the low-pass filtered coefficient (L_0) are computed as

$$H_0 = (P_0 + P_2)\alpha + P_1, \tag{3}$$

$$H_1 = (P_2 + P_4)\alpha + P_3, \tag{4}$$

$$L_0 = (H_0 + H_1)\beta + P_2. \tag{5}$$

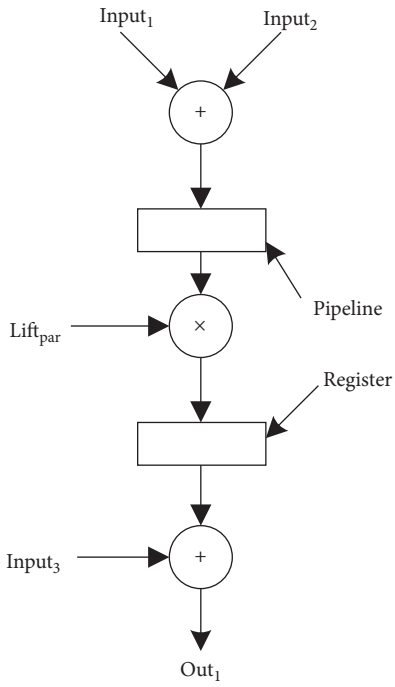

Figure 3: Schematic representation of the novel basic Lift_block employed in the proposed LFrWF architecture.

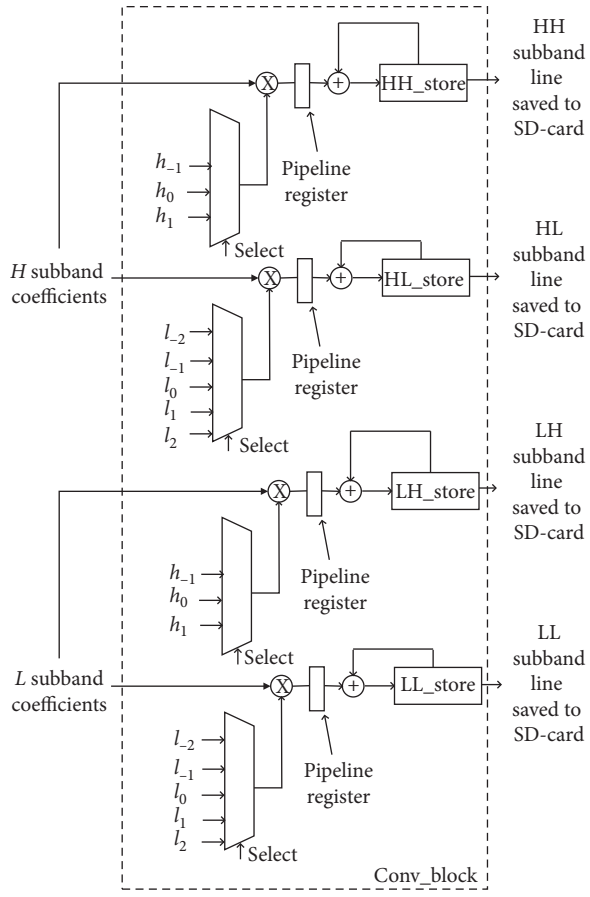

Figure 4: Schematic representation of Conv_block for computing subbands in the proposed LFrWF.

Table 2: Input and output to be used in the lift block in Figure 3.

	Input_1	Input_2	Input_3	Lift_{par}	Out_1
Step 1	P_0	P_2	P_1	α	H_0
Step 2	P_2	P_4	P_3	α	H_1
Step 3	H_0	H_1	P_2	β	L_0

where $\alpha = -0.5$ and $\beta = 0.25$ are lifting parameters [66]. Once the five pixels (P_0, P_1, P_2, P_3, and P_4) are processed, the first two pixels are discarded and two new pixels are read along with the previous last three pixels. The same procedure, in equations (3)–(5), is repeated on these new pixels to compute the H and L subband coefficients.

The basic Lift_block in Figure 3 requires two two-input adders and one multiplier. The functionality of this basic Lift_block essentially replaces the functionality of the convolution stage-1 block in the FrWF_{m} architecture, as shown in Figure 3 in [62] and elaborated in Figures 4–7 in [62]. For an LPF length of f_1 and an HPF length of f_2, the FrWF_m convolution stage-1 block in [62] requires $f_1 - 1$ two-input adders and f_1 multipliers for the low-pass filtering as well as $f_2 - 1$ two-input adders and f_2 multipliers for the high-pass filtering. Thus, for a 5/3 filter, the FrWF_{m} convolution stage-1 block requires six adders as well as eight multipliers.

3.2.3. Convolution Block. In the Conv_block in Figure 4, the H subband coefficients from the 1D_store register are multiplied by a suitable HPF and LPF coefficient (as determined by a multiplexer) and then added/stored with the previous value in the registers HH_store and HL_store, respectively. Similarly, the L subband coefficient in the 1D_store register is multiplied by a suitable HPF and LPF coefficient (as determined by a multiplexer) and then added/stored with the previous value in the registers LH_store and LL_store, respectively. The values in the registers HH_store, HL_store, LH_store, and LL_store are updated to compute the coefficients of the HH, HL, LH, and LL subbands, respectively.

We note that the Conv_block in Figure 4 is essentially equivalent to the aggregation of the FrWF convolution stage-2 blocks in Figures 4–7 in [62]. The Conv_block in Figure 4 requires four two-input adders and four multipliers. On the other hand, the aggregation of the FrWF convolution stage-2 blocks in Figures 4–7 in [62] requires two two-input adders and two multipliers.

3.2.4. Pipeline Registers. The Lift_block and the Conv_block use two and four pipeline registers, respectively, to temporarily save the intermediate results after each compute cycle. Through the use of the pipeline registers, the critical path delay (CPD) of the proposed LFrWF architecture becomes the multiplier delay T_m.

Overall, for a 5/3 filter, considering both the basic Lift_block (Figure 3) and the Conv_block (Figure 4), the proposed LFrWF_{m} requires six two-input adders and five multipliers compared to eight two-input adders and ten multipliers of the FrWF_{m} architecture (Figures 4–7 in [62]).

The proposed LFrWF architecture stores the original image and the subbands in the SD-card. Thus, higher wavelet decomposition levels can be computed with the same architecture, whereby the LL subband coefficients are taken as input.

3.3. Proposed Multiplier-less LFrWF$_{ml}$ Implementation. The 5/3 filter-bank coefficients (shown in Table 3) and the 5/3 filter-bank lifting parameters involve integer division and multiplication. Thus, they can be implemented using the shift and add method. More specifically, the convolution with the 5/3 filter-bank requires only integer multiplication and division and can therefore be implemented with only shift and add operations. For example, $z \cdot 0.25 = z \cdot 2^{-2}$, i.e., shifting the number z two times to the right is equivalent to dividing z by 4. The shift and add concept, as applied to the 5/3 filter coefficients, operates as follows:

(1) The filter coefficient $l_{-2} = l_2 = -1/8 = -1/2^3$ can be implemented by three right shift operations, followed by a complement operation

(2) The filter coefficient $l_{-1} = l_1 = 2/8 = 1/2^2$ can be implemented by two right shift operations

(3) The filter coefficient $l_0 = 6/8 = 1/2^2 + 1/2$ can be implemented by two right shift operations, followed by addition with one right shift

(4) The filter coefficient $h_0 = h_2 = -1/2$ can be implemented by one right shift operation, followed by a complement operation

(5) The coefficient $h_1 = 1$, thus, no shifting is required

With these specified shifting operations, the convolution block can be simplified and implemented using only shifters and adders. Multiplier-less computation blocks for the 5/3 LPF and HPF coefficients are given in Figures 5 and 6, respectively. One benefit of the multiplier-less implementation over the multiplier-based architecture in Section 3.2 is that the multiplier-less implementation reduces the CPD from the multiplier delay T_m down to the adder delay T_a.

3.4. Memory Requirement. In order to compute the DWT coefficients, the proposed LFrWF architecture uses four registers (HH_store, HL_store, LH_store, and LL_store), two register arrays (P_store and 1D_store), and six pipeline registers. The register array P_store (of size J words) is used to store an image line. The H and L subband coefficients computed by the Lift_block are saved in the register array 1D_store of 3 words. The four registers HH_store, HL_store, LH_store, and LL_store are of $J/2$ words each. The total memory requirement of the proposed architecture is equal to the sum of all registers, i.e.,

$$\text{Mem}_{\text{LFrWF}} = 3J + 9 \text{ words}. \tag{6}$$

3.5. Line Segmentation. Equation (6) indicates that LFrWF memory requirement grows with the image dimension J and thus will be significantly greater than the FrWF memory requirement of $2J + 22$ words for large images. In order to reduce the memory requirement of the proposed LFrWF architectures, each image line may be segmented, as illustrated in Figure 7, with overlapping of $\lfloor f_1/2 \rfloor$ coefficients at both boundaries of the second to the last, but one segment (the first and last segments only require overlapping at one boundary) (Appendix E in reference [88]). In this approach, only one line segment needs to be read into the register array P_store. Thus, the memory requirement of the LFrWF with G line segments is

$$\text{Mem}_{\text{LFrWF\_seg}} = \frac{J}{G} + 2J + 2\lfloor \frac{f_1}{2} \rfloor + 9 \text{ words}. \tag{7}$$

For the 5/3 filter-bank with a VFA of $f_1 = 5$ lines, the memory requirement is

$$\text{Mem}^{\text{5/3 filt.}}_{\text{LFrWF\_seg}} = \frac{J}{G} + 2J + 13 \text{ words}. \tag{8}$$

The other resource requirements are independent of line segmentation and remain unchanged.

The line segmentation reduces the memory requirement of the proposed LFrWF architectures so that their memory requirement can be reduced below the memory required by FrWF architectures of [62]. The FrWF architecture does not include a line segmentation provision; therefore, its memory requirement cannot be reduced further. We observe from Table 4 that the memory requirements of the proposed LFrWF architectures are greater than the FrWF memory requirements. However, by incorporating the line segmentation approach, the memory requirement of the LFrWF architectures can be reduced below that of the FrWF architectures. In case of the 5/3 filter-bank, we observe from Table 4 that the memory requirement of the FrWF architectures is $2J + 22$, while the memory requirement of LFrWF architecture with G line segments is $J/G + 2J + 13$, see equation (8). Therefore, the LFrWF memory requirement is less than the FrWF memory requirement if $G > J/9$.

Table 3: Coefficients of 5/3 filter-bank [84].

LPF coeff.	Value	HPF coeff.	Value
l_{-2}	$-1/8$	h_{-2}	0
l_{-1}	2/8	h_{-1}	0
l_0	6/8	h_0	$-1/2$
l_1	2/8	h_1	1
l_2	$-1/8$	h_2	$-1/2$

4. Results and Discussion

This section presents the implementation of the proposed LFrWF architecture and its comparison with state-of-the-art architectures. First, we compare the proposed LFrWF architecture with several state-of-the-art architectures in terms of the required numbers of adders and multipliers, as well as the critical path delay (CPD) and required memory. Next, the postimplementation results of the proposed LFrWF architectures are compared with the state-of-the-art FrWF

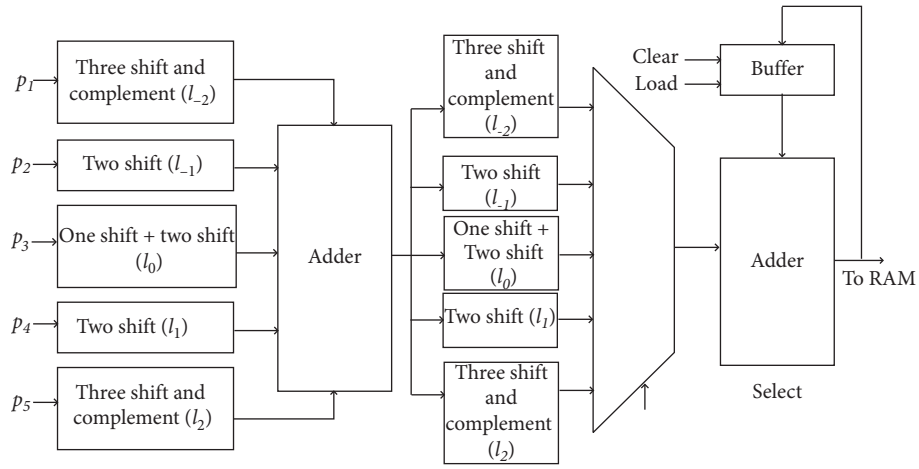

Figure 5: Multiplier-less block for LPF.

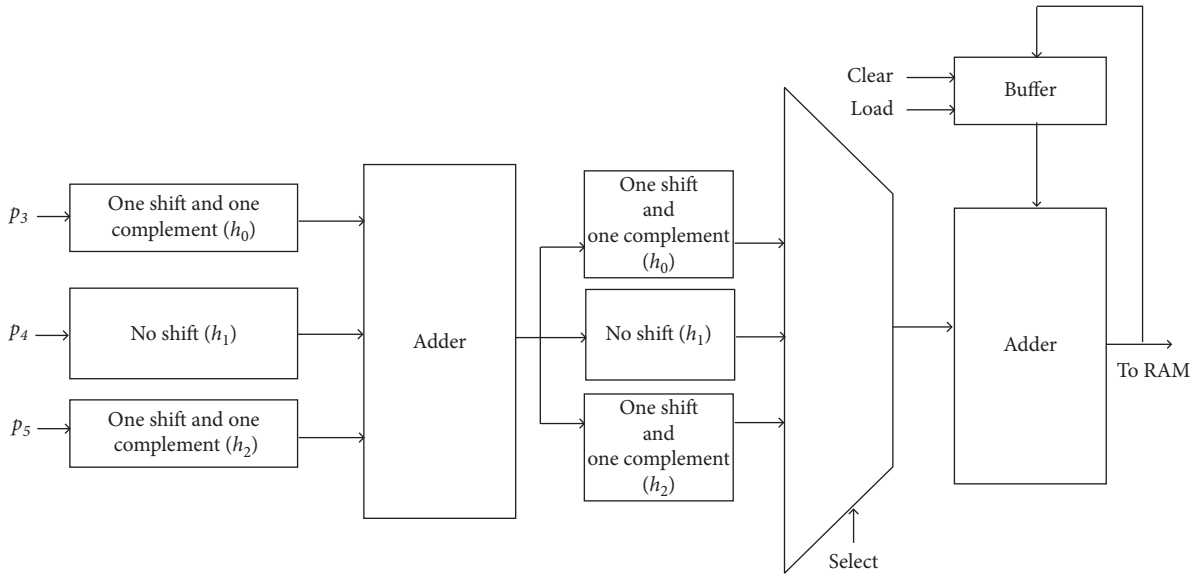

Figure 6: Multiplier-less block for HPF.

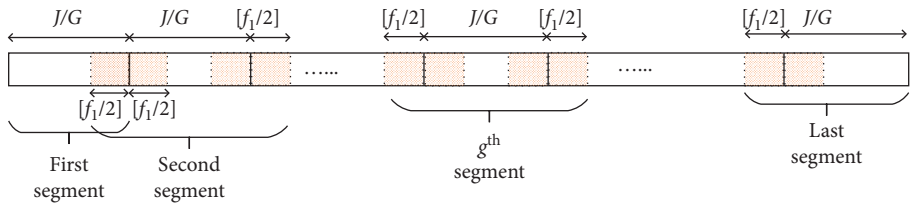

Figure 7: Illustration of segmentation of an image line of J pixels into G segments with overlapping of half the VFA, i.e., of $f_1/2$ lines at each end of a segment.

architecture [62] by implementing both architectures on the Xilinx Artix-7 FPGA platform.

4.1. Adders, Multipliers, CPD, and Memory. Table 4 compares the numbers of required adders and multipliers, as well as the CPD and the required RAM of the proposed LFrWF architectures with state-of-the-art architectures. The numbers of adders and multipliers of the existing state-of-the-art architectures shown in Table 4 have been taken from the corresponding papers. We observe from Table 4 that the proposed LFrWF_m architecture requires the least number of adders (namely, only six adders, see Figures 3 and 4) among the state-of-the-art architectures. While the proposed LFrWF_m reduces the number of required adders only by two compared to the FrWF_m architecture, the proposed LFrWF_m

Table 4: Comparison of required adders and multipliers, as well as critical path delay and required RAM of proposed LFrWF_{m} and LFrWF_{ml} vs. state-of-the-art architectures for one wavelet decomposition level with 5/3 filter-bank. (S = number of parallel proc. units; T_s = delay of shifter).

Architecture	Add.	Mul.	CPD	Mem.
PMA [85]	56	28	$T_a + T_s$	$4.8J$
RMA [85]	12	6	$2T_a + T_m$	$6.5J$
Savic and Rajovic [86]	22	17	$T_m + T_a$	$4J$
Aziz [87]	20	10	$2T_a$	$4J$
FrWF_{m} [62]	8	10	T_m	$2J + 22$
FrWF_{ml} [62]	10	0	T_a	$2J + 22$
LFrWF_{m}	6	5	T_m	$3J + 9$
LFrWF_{ml}	11	0	T_a	$3J + 9$

reduces the number of adders down to less than half of the other prior architectures. Among the architectures using multipliers, the proposed LFrWF_{m} architecture also requires the least number of multipliers, namely, only five multipliers, see Figures 3 and 4. Only the RMA [85] has a similarly low multiplier requirement with six multipliers (but requires approximately twice the memory compared to LFrWF). The other prior architectures require twice or more multipliers than the proposed LFrWF_m architecture.

We also observe from Table 4 that the CPD of the proposed LFrWF_{m} architecture and the FrWF_{m} architecture [62] are T_m, which is less than the architectures in [85, 86]. We note from Table 4 that the multiplier-less LFrWF_{ml} and FrWF_{ml} have reduced the CPD to T_a, which is less than the CPD of other state-of-the-art architectures. The CPD of T_a achieved by the proposed LFrWF_{ml} architecture cuts the shortest CPD of any existing architecture of $2T_a$ achieved by the Aziz architecture [87] down to half. Note that the shifter delay T_s is commonly larger than the adder delay T_a, i.e., $T_s > T_a$; thus, the PMA architecture [85] has a longer CPD than the Aziz architecture. The benefit of the reduction in CPD is that the architectures can be operated at higher frequencies, since maximum operations frequency = 1/CPD. As the CPD decreases, the maximum operating frequency increases.

Table 4 furthermore indicates that the FrWF architecture has the lowest memory requirement. However, the memory requirement of the proposed LFrWF architecture is less than the memory requirement of the other state-of-the-art architectures in Table 4. As noted in Section 4.3, with segmentation of a line of J words (pixels) into G segments (of J/G words each), the LFrWF memory requirement drops below the FrWF memory requirement if more than $J/9$ segments are used.

4.2. FPGA Implementation. The proposed LFrWF architecture computes the DWT coefficients of images based on lifting while following the FrWF approach. As observed from Table 4, the FrWF architecture [62] requires the least memory among the state-of-the-art architectures. Thus, we implemented the FrWF architectures [62] and the proposed LFrWF architectures (initially without segmentation, i.e., $G = 1$) on an Artix-7 FPGA (family: Artix-7, device: xc7a15t, package: csg324, speed: $-2L$). The implementations used identical multipliers, adders, and other components provided by the Xilinx Artix-7 FPGA family. All architectures used an input pixel width of 8 bits and a data-path width of 16 bits. Table 5 summarizes the FPGA implementation comparison. We report averages for evaluations with seven popular 512×512 (8 bits/pixel) test images, namely, "lena," "barbara," "goldhill," "boat," "mandrill," "peppers," and "zelda," obtained from the Waterloo Repertoire (http://links.uwaterloo.ca). The energy consumption is evaluated by multiplying the number of compute cycles with the average power consumption and the compute (clock) cycle durations of 5.0 ns and 1.5 ns for the architectures with multipliers and without multipliers, respectively. These clock cycle durations have been selected to satisfy the CPD constraint, as given in Table 5, namely, a CPD of 4.8 ns for the design with multipliers and a CPD of 1.45 ns for the multiplier-less design. The number of compute cycles and the average power consumption were evaluated by simulation with the Xilinx Vivado software suite, version 2018.2.

We observe from Table 5 that the proposed LFrWF_{m} architecture requires approximately 22% less LUTs, 34% less FFs, and 50% less compute cycles, and consumes 65% less energy than the FrWF_{m} architecture. Due to the reduced number of hardware components (LUTs and FFs), the area occupied by the LFrWF_{m} architecture will be less than the area of the corresponding FrWF_{m} architecture. Moreover, the proposed multiplier-less LFrWF_{ml} architecture requires 2.6% less FFs and 50% less cycles and consumes 43% less energy than the multiplier-less FrWF_{ml} architecture [62]. The proposed LFrWF_{ml} architecture requires slightly more LUTs than the multiplier-less FrWF_{ml} architecture.

We also observe from Table 5 that the proposed LFrWF reduces the number of required compute cycles to roughly half the compute cycles required by the FrWF. More specifically, while the FrWF requires on the order of 10 million compute cycles for a 512×512 image, the proposed LFrWF requires only a little more than 5 million compute cycles. This substantial reduction is primarily due to the computational efficiency of the novel Lift_block (see Section 3.2.2) for computing the decomposition subband coefficients.

Moreover, we observe from Table 5 that the power consumption of the proposed LFrWF architecture with multipliers is less than the power consumption of the corresponding FrWF architecture with multipliers, while the multiplier-less LFrWF and FrWF have approximately the same power consumption. The energy consumption is evaluated by multiplying clock cycle duration (which is based on the CPD) with the number of clock cycles and the consumed power. Due to the reduced (almost half) number of compute cycles and the lower (or same) power consumption, the energy consumption levels of the proposed LFrWF architectures are substantially lower than the energy consumption levels of the FrWF architectures. We further observe from Table 5 that compared to the designs with multipliers, the multiplier-less designs of both the LFrWF and the FrWF have the same numbers of clock cycles, but shorter CPD and (slightly) reduced power levels; thus, the multiplier-less designs have substantially reduced energy consumption levels.

Table 5: Comparison of FPGA implementation resource utilization of proposed LFrWF vs. FrWF [62] with 5/3 filter-bank for one wavelet decomposition level for 512×512 image.

Param.	$FrWF_m$	$LFrWF_m$	Multiplier-less	
			$FrWF_{ml}$	$LFrWF_{ml}$
LUT	215	168	119	135
FF	305	201	190	185
CC	10,485,760	5,242,880	10,485,760	5,242,880
CPD (ns)	4.80	4.80	1.45	1.45
Power (W)	0.162	0.114	0.110	0.109
Energy (mJ)	8.545	3.014	1.741	0.860

LUT, look-up tables; FF, flip-flops; CC, compute cycles.

We also observe from Table 5 that both architectures have the same CPD. We note that the numbers of hardware components, e.g., adders, multipliers, LUT, and FF, and other parameters, such as the number of clock cycles, memory, and CPD (T_m or T_a), are independent of the platform on which the design is implemented and the test image. Among the results presented in Tables 4–6, only the energy consumption, the power consumption, and the energy delay product (EDP) depend on the platform and image.

4.3. Line Segmentation. We observe from Table 6 that increasing the number of line segments G reduces the memory requirement while increasing the number of compute cycles and the energy consumption. The compute cycle and energy consumption increases are mainly due to the overlapping of $\lfloor f_1/2 \rfloor$ coefficients at the line segment boundaries which need to be read twice. However, for all line segmentations ($G = 2, 4, 8$), the number of compute cycles and energy consumption are less than for the corresponding FrWF architectures, see Table 5. We observe from Tables 5 and 6 that even with $G = 8$ segments per line, the number of compute cycles and the energy consumption of the proposed LFrWF architectures are less than those for the corresponding FrWF architectures. Since the FrWF architectures of [62] read only 5 pixels at a time, the segmentation approach cannot be incorporated into the FrWF architecture. Hence, the memory of the FrWF architectures cannot be further reduced by incorporating line segmentation.

The EDPs of the LFrWF and FrWF architectures with and without multipliers are compared in Figures 8 and 9, respectively. The EDP, which characterizes both the consumed energy and the computational performance, is evaluated by multiplying the consumed energy with the corresponding clock cycle duration. We observe from Figures 8 and 9 that the EDPs of the proposed LFrWF architectures (with and without multipliers) are less than the EDPs of the corresponding FrWF architectures (with and without multipliers). The EDP of the proposed $LFrWF_m$ architecture with multipliers ($G = 1$) is approximately 65% less than that for the FrWF architecture with multipliers, and the EDP of the proposed $LFrWF_{ml}$ multiplier-less architecture ($G = 1$) is approximately 43% less than that for the multiplier-less FrWF architecture. We observe from

Table 6: Line segmentation evaluation: memory requirement and number of compute cycles for proposed LFrWF as well as energy consumption of proposed $LFrWF_m$ and $LFrWF_{ml}$ for different line segment numbers G with 5/3 filter-bank for 512×512 image.

Parameters	$G = 1$	$G = 2$	$G = 4$	$G = 8$
Memory (words)	1545	1293	1165	1101
# cycles	5242880	5243904	5244928	5245952
E. (mJ) $LFrWF_m$	3.014	3.040	3.073	3.122
E. (mJ) $LFrWF_{ml}$	0.990	0.997	1.006	1.017

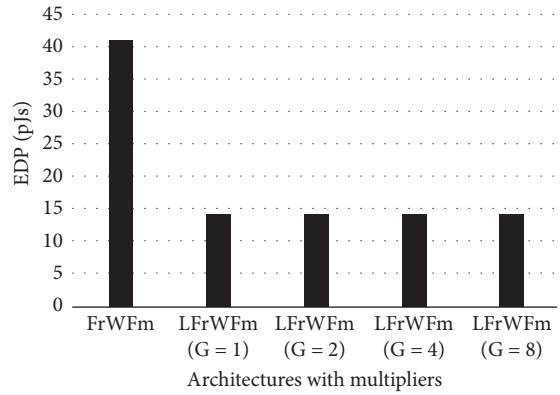

Figure 8: Energy delay product (EDP) of architectures with multipliers: LFrWF with G line segments and FrWF.

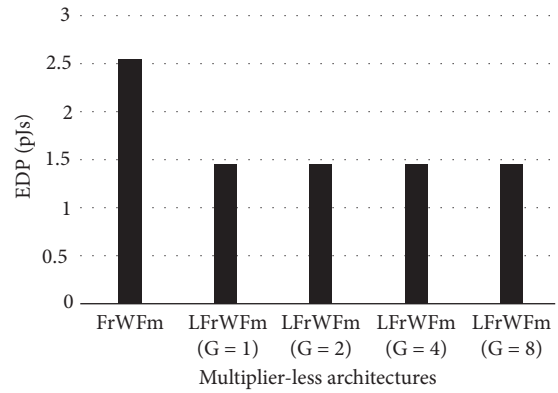

Figure 9: Energy delay product (EDP) of architectures without multipliers: $LFrWF_{ml}$ with G line segments and $FrWF_{ml}$.

Figures 8 and 9 that the EDPs of the proposed LFrWF architectures increase with the number G of segments. However, even with $G = 8$ segments per image line, the EDPs of the proposed LFrWF architectures are less than those for the corresponding FrWF architectures.

5. Conclusion

This paper proposed and evaluated a lifting-based architecture to compute the DWT coefficients of an image based on the FrWF approach with a 5/3 filter-bank. The proposed architecture requires fewer adders and multipliers than state-of-the-art architectures. The proposed architecture

with multipliers ($LFrWF_m$) and without multipliers ($LFrWF_{ml}$) and the state-of-the-art FrWF architecture (with and without multipliers) [62] have been implemented on the same FPGA board and compared.

The experimental results show that the proposed $LFrWF_m$ architecture requires less hardware components (and thus less area) and consumes 65% less energy than the $FrWF_m$ architecture. Moreover, the proposed $LFrWF_{ml}$ architecture consumes 43% less energy with only a slight increase in area compared to the $FrWF_{ml}$ architecture. The lower energy consumption with minimal area overhead makes the proposed architectures promising candidates for computing the DWT of images on resource-constrained wearable sensors.

An important direction for future research is to integrate the LFrWF architecture with efficient architectures of state-of-the-art wavelet-based image coding algorithms to design FPGA-based image coders for real-time applications on wearable visual sensors and IoT platforms. Another interesting future research direction is the examination of the use of our proposed approach in the context of compressive sensing [15, 89].

Data Availability

The evaluation data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

References

[1] Y. Gu, Y. Tian, and E. Ekici, "Real-time multimedia processing in video sensor networks," *Signal Processing: Image Communication*, vol. 22, no. 3, pp. 237–251, 2007.

[2] T. Kamal, R. Watkins, Z. Cen, J. Rubinstein, G. Kong, and W. M. Lee, "Design and fabrication of a passive droplet dispenser for portable high resolution imaging system," *Scientific Reports*, vol. 7, pp. 1–13, 2017.

[3] R. LeMoyne and T. Mastroianni, "Wearable and wireless gait analysis platforms: smartphones and portable media devices," in *Wireless MEMS Networks and Applications*, D. Uttamchandani, Ed., pp. 129–152, Woodhead Publishing, Cambridge, UK, 2017.

[4] A. Nag, S. C. Mukhopadhyay, and J. Kosel, "Wearable flexible sensors: a review," *IEEE Sensors Journal*, vol. 17, no. 13, pp. 3949–3960, 2017.

[5] H. Ng, W.-H. Tan, J. Abdullah, and H.-L. Tong, "Development of vision based multiview gait recognition system with MMUGait database," *The Scientific World Journal*, vol. 2014, 2014.

[6] S. Plangi, A. Hadachi, A. Lind, and A. Bensrhair, "Real-time vehicles tracking based on mobile multi-sensor fusion," *IEEE Sensors Journal*, vol. 18, no. 24, pp. 10077–10084, 2018.

[7] S. Seneviratne, Y. Hu, T. Nguyen et al., "A survey of wearable devices and challenges," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2573–2620, 2017.

[8] L. Tsao, L. Li, and L. Ma, "Human work and status evaluation based on wearable sensors in human factors and ergonomics: a review," *IEEE Transactions on Human-Machine Systems*, vol. 49, no. 1, pp. 72–84, 2019.

[9] B. Velusamy and S. C. Pushpan, "An enhanced channel access method to mitigate the effect of interference among body sensor networks for smart healthcare," *IEEE Sensors Journal*, vol. 19, no. 16, pp. 7082–7088, 2019.

[10] G. Yang, W. Tan, H. Jin, T. Zhao, and L. Tu, "Review wearable sensing system for gait recognition," *Cluster Computing*, vol. 22, no. S2, pp. 3021–3029, 2019.

[11] M. Zheng, P. X. Liu, R. Gravina, and G. Fortino, "An emerging wearable world: new gadgetry produces a rising tide of changes and challenges," *IEEE Systems, Man, and Cybernetics Magazine*, vol. 4, no. 4, pp. 6–14, 2018.

[12] J. Wang and S. Payandeh, "Hand motion and posture recognition in a network of calibrated cameras," *Advances in Multimedia*, vol. 2017, Article ID 2162078, 2017.

[13] C.-H. Hsia, J.-M. Guo, and J.-S. Chiang, "A fast Discrete Wavelet Transform algorithm for visual processing applications," *Signal Processing*, vol. 92, no. 1, pp. 89–106, 2012.

[14] S. Soro and W. Heinzelman, "A survey of visual sensor networks," *Advances in Multimedia*, vol. 2009, Article ID 640386, 2009.

[15] A. S. Unde and P. P. Deepthi, "Rate-distortion analysis of structured sensing matrices for block compressive sensing of images," *Signal Processing: Image Communication*, vol. 65, pp. 115–127, 2018.

[16] S. Heng, C. So-In, and T. G. Nguyen, "Distributed image compression architecture over wireless multimedia sensor networks," *Wireless Communications and Mobile Computing*, vol. 2017, Article ID 5471721, 2017.

[17] F. Sun, C. Mao, X. Fan, and Y. Li, "Accelerometer-based speed-adaptive gait authentication method for wearable IoT devices," *IEEE Internet of Things Journal*, vol. 6, no. 1, pp. 820–830, 2019.

[18] H. Baali, H. Djelouat, A. Amira, and F. Bensaali, "Empowering technology enabled care using IoT and smart devices: a review," *IEEE Sensors Journal*, vol. 18, no. 5, pp. 1790–1809, 2018.

[19] S. Hiremath, G. Yang, and K. Mankodiya, "Wearable Internet of Things: concept, architectural components and promises for person-centered healthcare," in *Proceedings of the 7th International Conference on Wireless Mobile Communication and Healthcare (MOBIHEALTH)*, pp. 304–307, Vienna, Austria, 2014.

[20] M. Manas, A. Sinha, S. Sharma, and M. R. Mahboob, "A novel approach for IoT based wearable health monitoring and messaging system," *Journal of Ambient Intelligence and Humanized Computing*, vol. 10, no. 7, pp. 2817–2828, 2019.

[21] L. E. Lima, B. Y. L. Kimura, and V. Rosset, "Experimental environments for the Internet of Things: a review," *IEEE Sensors Journal*, vol. 19, no. 9, pp. 3203–3211, 2019.

[22] F. Javed, M. K. Afzal, M. Sharif, and B. Kim, "Internet of Things (IoTs) operating systems support, networking technologies, applications, and challenges: a comparative review," *IEEE Sensors Journal*, vol. 20, no. 3, pp. 2062–2100, 2018.

[23] K. Kim, H. Kim, and H. Kim, "Image-based construction hazard avoidance system using augmented reality in wearable device," *Automation in Construction*, vol. 83, pp. 390–403, 2017.

[24] Z. Bauer, A. Dominguez, E. Cruz, F. Gomez-Donoso, S. Orts-Escolano, and M. Cazorla, "Enhancing perception for the visually impaired with deep learning techniques and low-cost wearable sensors," *Pattern Recognition Letters*, vol. 137, pp. 27–36, 2020.

[25] U. Lee, K. Han, H. Cho et al., "Intelligent positive computing with mobile, wearable, and IoT devices: literature review and research directions," *Ad Hoc Networks*, vol. 83, pp. 8–24, 2019.

[26] M. Ayaz, M. Ammad-uddin, I. Baig, and E.-H. M. Aggoune, "Wireless sensor's civil applications, prototypes, and future integration possibilities: a review," *IEEE Sensors Journal*, vol. 18, no. 1, pp. 4–30, 2018.

[27] A. Kos, V. Milutinović, and A. Umek, "Challenges in wireless communication for connected sensors and wearable devices used in sport biofeedback applications," *Future Generation Computer Systems*, vol. 92, pp. 582–592, 2019.

[28] M. Cagnazzo, F. Delfino, L. Vollero, and A. Zinicola, "Trading off quality and complexity for a HVQ-based video codec on portable devices," *Journal of Visual Communication and Image Representation*, vol. 17, no. 3, pp. 564–572, 2006.

[29] A. Chefi, A. Soudani, and G. Sicard, "Hardware compression scheme based on low complexity arithmetic encoding for low power image transmission over WSNs," *AEU-International Journal of Electronics and Communications*, vol. 68, no. 3, pp. 193–200, 2014.

[30] M. Chen, Y. Zhang, and C. Lu, "Efficient architecture of variable size HEVC 2D-DCT for FPGA platforms," *AEU-International Journal of Electronics and Communications*, vol. 73, pp. 1–8, 2017.

[31] A. Madanayake, R. J. Cintra, V. Dimitrov et al., "Low-power VLSI architectures for DCT\/DWT: precision vs approximation for HD video, biomedical, and smart antenna applications," *IEEE Circuits and Systems Magazine*, vol. 15, no. 1, pp. 25–47, 2015.

[32] T. K. Araghi, A. A. Manaf, A. Alarood, and A. B. Zainol, "Host feasibility investigation to improve robustness in hybrid DWT+SVD based image watermarking schemes," *Advances in Multimedia*, vol. 2018, Article ID 1609378, 2018.

[33] H. Persson, A. Brunstrom, and T. Ottosson, "Utilizing cross-layer information to improve performance in JPEG2000 decoding," *Advances in Multimedia*, vol. 2007, Article ID 024758, 2007.

[34] B. Yan, T. Pei, and X. Wang, "Wavelet method for automatic detection of eye-movement behaviors," *IEEE Sensors Journal*, vol. 19, no. 8, pp. 3085–3091, 2019.

[35] B. D. E. Cherif and A. Bendiabdellah, "Detection of two-level inverter open-circuit fault using a combined DWT-NN approach," *Journal of Control Science and Engineering*, vol. 2018, Article ID 1976836, 2018.

[36] A. F. R. Guarda, J. M. Santos, L. A. da Silva Cruz, P. A. A. Assunção, N. M. M. Rodrigues, and S. M. M. de Faria, "A method to improve HEVC lossless coding of volumetric medical images," *Signal Processing: Image Communication*, vol. 59, pp. 96–104, 2017.

[37] M. L. L. De Faria, C. E. Cugnasca, and J. R. A. Amazonas, "Insights into IoT data and an innovative DWT-based technique to denoise sensor signals," *IEEE Sensors Journal*, vol. 18, no. 1, pp. 237–247, 2018.

[38] F. Han, X. Qiao, Y. Ma, W. Yan, X. Wang, and X. Pan, "Grass leaf identification using dbN wavelet and CILBP," *Advances in Multimedia*, vol. 2020, Article ID 1909875, 8 pages, 2020.

[39] J. G. Escobedo and K. B. Ozanyan, "Tiled-block image reconstruction by wavelet- based, parallel-filtered back-projection," *IEEE Sensors Journal*, vol. 16, no. 12, pp. 4839–4846, 2016.

[40] D. Bhowmik and C. Abhayaratne, "2D + t wavelet domain video watermarking," *Advances in Multimedia*, vol. 2012, Article ID 973418, 19 pages, 2012.

[41] P. R. Hill, N. Anantrasirichai, A. Achim, M. E. Al-Mualla, and D. R. Bull, "Undecimated dual-tree complex wavelet transforms," *Signal Processing: Image Communication*, vol. 35, pp. 61–70, 2015.

[42] T. Brahimi, F. Laouir, L. Boubchir, and A. Ali-Chérif, "An improved wavelet-based image coder for embedded greyscale and colour image compression," *AEU-International Journal of Electronics and Communications*, vol. 73, pp. 183–192, 2017.

[43] H. Liu, K.-K. Huang, C.-X. Ren, Y.-F. Yu, and Z.-R. Lai, "Quadtree coding with adaptive scanning order for space-borne image compression," *Signal Processing: Image Communication*, vol. 55, pp. 1–9, 2017.

[44] R. Banerjee and S. Das Bit, "An energy efficient image compression scheme for wireless multimedia sensor network using curve fitting technique," *Wireless Networks*, vol. 25, no. 1, pp. 167–183, 2019.

[45] M. Tükel, A. Yurdakul, and B. Örs, "Customizable embedded processor array for multimedia applications," *Integration*, vol. 60, pp. 213–223, 2018.

[46] D. G. Costa, "Visual sensors hardware platforms: a review," *IEEE Sensors Journal*, vol. 20, no. 8, pp. 4025–4033, 2020.

[47] L. Linguaglossa, S. Lange, S. Pontarelli et al., "Survey of performance acceleration techniques for network function virtualization," *Proceedings of the IEEE*, vol. 107, no. 4, pp. 746–764, 2019.

[48] G. S. Niemiec, L. M. S. Batista, A. E. Schaeffer-Filho, and G. L. Nazar, "A survey on FPGA support for the feasible execution of virtualized network functions," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 1, pp. 504–525, 2020.

[49] P. Shantharama, A. S. Thyagaturu, and M. Reisslein, "Hardware-accelerated platforms and infrastructures for network functions: a survey of enabling technologies and research studies," *IEEE Access*, vol. 8, pp. 132021–132085, 2020.

[50] A. J. Ferrer, J. M. Marquès, and J. Jorba, "Towards the decentralised cloud: survey on approaches and challenges for mobile, ad hoc, and edge computing," *ACM Computing Surveys (CSUR)*, vol. 51, no. 6, 2019.

[51] M. Mehrabi, D. You, V. Latzko, H. Salah, M. Reisslein, and F. H. Fitzek, "Device-enhanced MEC: multi-access edge computing (MEC) aided by end device computation and caching: a survey," *IEEE Access*, vol. 7, pp. 166 079–166 108, 2019.

[52] N. Karakoc, A. Scaglione, A. Nedic, and M. Reisslein, "Multi-layer decomposition of network utility maximization problems," *IEEE/ACM Transactions on Networking*, vol. 28, no. 5, pp. 2077–2091, 2020.

[53] T. Luo, H.-P. Tan, and T. Q. S. Quek, "Sensor OpenFlow: enabling software-defined wireless sensor networks," *IEEE Communications Letters*, vol. 16, no. 11, pp. 1896–1899, 2012.

[54] P. Shantharama, A. S. Thyagaturu, N. Karakoc et al., "LayBack: SDN management of multi-access edge computing (MEC) for network access services and radio resource sharing," *IEEE Access*, vol. 6, pp. 57 545–557 561, 2018.

[55] S. Rein and M. Reisslein, "Low-memory wavelet transforms for wireless sensor networks: a tutorial," *IEEE Communications Surveys & Tutorials*, vol. 13, no. 2, pp. 291–307, 2011.

[56] S. Rein and M. Reisslein, "Performance evaluation of the fractional wavelet filter: a low-memory image wavelet transform for multimedia sensor networks," *Ad Hoc Networks*, vol. 9, no. 4, pp. 482–496, 2011.

[57] B. K. Mohanty, A. Mahajan, and P. K. Meher, "Area- and power-efficient architecture for high-throughput

implementation of lifting 2-D DWT," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 59, no. 7, pp. 434–438, 2012.

[58] R. K. Bhattar, K. R. Ramakrishnan, and K. S. Dasgupta, "Strip based coding for large images using wavelets," *Signal Processing: Image Communication*, vol. 17, no. 6, pp. 441–456, 2002.

[59] Y. Hu and C. C. Jong, "A memory-efficient scalable architecture for lifting-based discrete wavelet transform," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 60, no. 8, pp. 502–506, 2013.

[60] L. Ye and Z. Hou, "Memory efficient multilevel discrete wavelet transform schemes for JPEG2000," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 25, no. 11, pp. 1773–1785, 2015.

[61] Y. Hu and V. K. Prasanna, "Energy- and area-efficient parameterized lifting-based 2-D DWT architecture on FPGA," in *Proceedings of the IEEE High Performance Extreme Computing Conference (HPEC)*, pp. 1–6, MA, USA, 2014.

[62] M. Tausif, A. Jain, E. Khan, and M. Hasan, "Low memory architectures of fractional wavelet filter for low-cost visual sensors and wearable devices," *IEEE Sensors Journal*, vol. 20, no. 13, pp. 6863–6871, 2020.

[63] C. Zhang, C. Wang, and M. O. Ahmad, "A pipeline VLSI architecture for fast computation of the 2-D discrete wavelet transform," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 59, no. 8, pp. 1775–1785, 2012.

[64] B. K. Mohanty and P. K. Meher, "Memory-efficient high-speed convolution-based generic structure for multilevel 2-D DWT," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 23, no. 2, pp. 353–363, 2013.

[65] Y. Zhang, H. Cao, H. Jiang, and B. Li, "Memory-efficient high-speed VLSI implementation of multi-level discrete wavelet transform," *Journal of Visual Communication and Image Representation*, vol. 38, pp. 297–306, 2016.

[66] C.-H. Hsia, J.-S. Chiang, and J.-M. Guo, "Memory-efficient hardware architecture of 2-D dual-mode lifting-based discrete wavelet transform," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 23, no. 4, pp. 671–683, 2013.

[67] G. Hegde, K. S. Reddy, and T. K. Shetty Ramesh, "A new approach for 1-D and 2-D DWT architectures using LUT based lifting and flipping cell," *AEU - International Journal of Electronics and Communications*, vol. 97, pp. 165–177, 2018.

[68] M. M. A. Basiri and S. N. Mahammad, "An efficient VLSI architecture for lifting based 1D/2D discrete wavelet transform," *Microprocessors and Microsystems*, vol. 47, pp. 404–418, 2016.

[69] C. Wang, J. Zhou, L. Liao et al., "Near-threshold energy- and area-efficient reconfigurable DWPT/DWT processor for healthcare-monitoring applications," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 62, no. 1, pp. 70–74, 2015.

[70] T. Wang, P. Huang, K. Chen et al., "Energy-efficient configurable discrete wavelet transform for neural sensing applications," in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1841–1844, Melbourne, Australia, June 2014.

[71] G. Kumar, N. Balaji, N. Balaji, K. Reddy, and V. Thanuja, "Power and area efficient radix-8 booth multiplier for 2-D DWT architecture," *International Journal of Intelligent Engineering and Systems*, vol. 12, no. 3, pp. 148–155, 2019.

[72] J. Franco, G. Bernabé, J. Fernández, and M. Ujaldón, "The 2D wavelet transform on emerging architectures: GPUs and multicores," *Journal of Real-Time Image Processing*, vol. 7, no. 3, pp. 145–152, 2012.

[73] V. Galiano, O. López, M. P. Malumbres, and H. Migallón, "Parallel strategies for 2D Discrete Wavelet Transform in shared memory systems and GPUs," *The Journal of Supercomputing*, vol. 64, no. 1, pp. 4–16, 2013.

[74] T. Ikuzawa, F. Ino, and K. Hagihara, "Reducing memory usage by the lifting-based discrete wavelet transform with a unified buffer on a GPU," *Journal of Parallel and Distributed Computing*, vol. 93-94, pp. 44–55, 2016.

[75] W. J. van der Laan, A. C. Jalba, and J. B. T. M. Roerdink, "Accelerating wavelet lifting on graphics hardware using CUDA," *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, no. 1, pp. 132–146, 2011.

[76] M. Kucis, D. Barina, M. Kula, and P. Zemcik, "2-D discrete wavelet transform using GPU," in *Proceedings of the International Symposium on Computer Architecture and High Performance Computing Workshop*, pp. 1–6, Paris, France, 2014.

[77] T. M. Quan and W.-K. Jeong, "A fast discrete wavelet transform using hybrid parallelism on GPUs," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 11, pp. 3088–3100, 2016.

[78] R. Khemiri, F. Sayadi, M. Atri, and R. Tourki, "MatLab acceleration for DWT "daubechies 9/7" for JPEG2000 standard on GPU," in *Proceedings of the Global Summit on Computer & Information Technology (GSCIT)*, pp. 1–4, Sousse, Tunisia, 2014.

[79] G. Suseela and Y. Asnath Victy Phamila, "Energy efficient image coding techniques for low power sensor nodes: a review," *Ain Shams Engineering Journal*, vol. 9, no. 4, pp. 2961–2972, 2018.

[80] H. Sun and Y. Q. Shi, *Image and Video Compression for Multimedia Engineering: Fundamentals, Algorithms, and Standards*, CRC Press, Boca Raton, FL, USA, 2nd edition, 2008.

[81] H. ZainEldin, M. A. Elhosseini, and H. A. Ali, "A modified listless strip based SPIHT for wireless multimedia sensor networks," *Computers & Electrical Engineering*, vol. 56, pp. 519–532, 2016.

[82] W. Sweldens, "The lifting scheme: a construction of second generation wavelets," *SIAM Journal on Mathematical Analysis*, vol. 29, no. 2, pp. 511–546, 1998.

[83] S. A. Rein, F. H. P. Fitzek, C. Gühmann, and T. Sikora, "Evaluation of the wavelet image two-line coder: a low complexity scheme for image compression," *Signal Processing: Image Communication*, vol. 37, pp. 58–74, 2015.

[84] A. Skodras, C. Christopoulos, and T. Ebrahimi, "The JPEG 2000 still image compression standard," *IEEE Signal Processing Magazine*, vol. 18, no. 5, pp. 36–58, 2001.

[85] A. D. Darji, S. S. Kushwah, S. N. Merchant, and A. N. Chandorkar, "High-performance hardware architectures for multi-level lifting-based discrete wavelet transform," *EURASIP Journal on Image and Video Processing*, vol. 2014, 2014.

[86] G. Savić and V. Rajović, "Novel memory efficient hardware architecture for 5/3 lifting-based 2D Inverse DWT," *Journal of Circuits, Systems and Computers*, vol. 28, no. 7, 2018.

[87] S. M. Aziz and D. M. Pham, "Efficient parallel architecture for multi-level forward discrete wavelet transform processors," *Computers & Electrical Engineering*, vol. 38, no. 5, pp. 1325–1335, 2012.

[88] M. Tausif, E. Khan, M. Hasan, and M. Reisslein, "SMFrWF: segmented modified fractional wavelet filter: fast low-memory discrete wavelet transform (DWT)," *IEEE Access*, vol. 7, pp. 84 448–84 467, 2019.

[89] R. Li, H. Liu, Y. Zeng, and Y. Li, "Block compressed sensing of images using adaptive granular reconstruction," *Advances in Multimedia*, vol. 2016, 2016.