Memory-efficient architecture for FrWF-based DWT of high-resolution images for IoMT applications

As Published: https://doi.org/10.1007/s11042-020-10258-0

Publisher: Springer US

Persistent URL: https://hdl.handle.net/1721.1/132076

Version: Author's final manuscript: final author's manuscript post peer review, without publisher's formatting or copy editing

Memory-efficient Architecture for FrWF-based DWT of High-resolution Images for IoMT Applications

Cite this article as: Mohd Tausif, Abhinandan Jain, Ekram Khan and Mohd Hasan, Memory-efficient Architecture for FrWF-based DWT of High-resolution Images for IoMT Applications, *Multimedia Tools and Applications* doi: 10.1007/s11042-020-10258-0

Multimedia Tools and Applications manuscript No.
(will be inserted by the editor)

Memory-efficient Architecture for FrWF-based DWT of High-resolution Images for IoMT Applications

Mohd Tausif[1] · Abhinandan Jain[2] · Ekram Khan[3] · Mohd Hasan[3]

Received: date / Accepted: date

Abstract This paper proposes a simple low memory architecture for computing discrete wavelet transform (DWT) of high-resolution (HR) images on low-cost memory-constrained sensor nodes used in visual sensor networks (VSN) or Internet of Multimedia Things (IoMT). The main feature of the proposed architecture is the novel data scanning technique that makes memory requirement independent of the image size. The proposed architecture needs only $(30S)$ words of memory, where S is the number of parallel processing units and a critical path delay (CPD) equal to the delay of a multiplier (T_m). Furthermore, a multiplierless version of this architecture is also proposed which reduces the CPD to $T_a<T_m$ (where T_a is the delay of an adder). In order to evaluate their effectiveness, the proposed architectures are coded in HDL and implemented on same FPGA board. Their performance is also compared with other state-of-the-art low memory DWT architectures. The experimental results show the superiority of the proposed architectures in terms of memory and CPD compared to existing DWT architectures. Moreover, the reduction in

M. Tausif
E-mail: mohdtausif32@gmail.com

A. Jain
E-mail: abyjain007@gmai.com

E. Khan
E-mail: ekhan67@gmail.com

M. Hasan
E-mail: mohdhasan097@gmail.com

[1] Department of Electronics & Communication, J. K. Institute of Applied Physics & Technology, University of Allahabad, Prayagraj, India

[2] MIT Media Lab, Massachusetts Institute of Technology, Cambridge, USA

[3] Department of Electronics Engg., Z. H. College of Engg. & Technology, Aligarh Muslim Universiry, Aligarh, India

CPD to T_a indicates that the operating frequency can be scaled up by several factors and can be chosen depending upon the application. Compared to one of the best state-of-the-art DWT architecture, proposed multiplierless architecture (with S =4) needs 57.37% less LUT's and 64.39% less flip-flops for HR image of dimension 2048×2048. Moreover, the proposed architecture needs no LUTRAM and DSP, whereas the existing architecture requires 3264 LUTRAM and 24 DSP's. Thus the proposed multiplierless architecture is superior to the existing state-of-the-art architecture and is suitable for IoMT/VSNs.

Keywords Discrete Wavelet Transform · visual sensor networks · high-resolution images · image coders · low memory architecture · Internet of Multimedia Things

1 Introduction

1.1 Motivation

Wireless sensor networks (WSN) consists of network of sensors deployed over a geographical area. The sensors also form foundation technology of Internet of Multimedia Things (IoMT) [1], [2]. The IoMT is a smart system which connects all things to the Internet for information sensing, data computing and exchange. More and more devices are likely to be linked through IoMT in future [3], [4]. WSN, therefore can be considered as a link between the cyber world and the real world [5]. The cameras can be integrated with sensor nodes to form visual sensors. Many such interconnected sensors constitute the Visual sensor networks (VSN) [6], or Wireless multimedia sensor networks (WMSNs). However, the available memory and power of these sensors are extremely low [7]. Moreover, the VSN and hand-held portable-multimedia devices, which are very popular nowadays support low bit-rates. In order to facilitate visual communication through these sensors there is need to design efficient image/video coding schemes that can be implemented over these memory and power constrained VSN.

Discrete Cosine transform (DCT) and Discrete Wavelet transform (DWT) are among the most popular transforms and are widely used in the field of image processing [8]. DCT is used in Joint Photographic Experts Group (JPEG). Moreover, the implementation of DCT requires less memory as compared to DWT. But DCT is applied on blocks within images, which leads to blocking artifacts. Also the performance of DCT is inferior in comparison to DWT at lower bit-rates [9]. The demand for good quality images at lower bit-rates is increasing day-by-day.

DWT is applied on the complete image and do not suffer with blocking artifacts. Also DWT has the feature of multi-resolution by which a signal can be analyzed in both space and frequency domain. Moreover, DWT has high energy compaction property. Due to these features, DWT is becoming popular day-by-day and is used in many applications like bio-medical signal processing, bio-medicine, computer graphics and real-time processing [10]. Also DWT is

used for transforming the image in JPEG 2000 [10]. But all the nice features of DWT come at the cost of increased memory requirement for implementing it. All the rows of the image are filtered (one-by-one) by low-pass filter (LPF) and high pass filter (HPF) followed by down-sampling by a factor of two, resulting in two sub-bands L and H respectively. Applying the filters on the row is termed as one dimensional DWT (1D-DWT). After this the columns of L and H subbands are filtered by LPF and HPF followed by downsampling by a factor of two resulting in final sub-bands LL, LH, HL and HH. This corresponds to one level of two dimensional DWT (2D-DWT). For computing higher levels of 2D-DWT, the same procedure as stated above is applied on the LL sub-band [11].

As it is clear from above that the complete image needs to be stored in RAM for applying DWT on it, resulting in large memory requirements. For a gray-scale image of size $N \times N$, the memory required by DWT would be $8N^2$ bytes (if floating point filter coefficients are used) [12]. Furthermore, the memory requirement of DWT increases linearly with the image size. There is increasing demand of high-resolution (HR) images nowadays. Thus workstations equipped with high-end processors may also face difficulty in implementing DWT especially for HR-images. This memory issue also limits the implementation of DWT for HR images on low-cost wireless visual sensor nodes or hand-held portable multimedia-devices which have 10 kB RAM [13].

Researchers realized this problem and proposed various solutions for reducing the memory required in transforming the image by DWT. The work on memory reduction of DWT can be classified mainly in four categories namely line-based DWT, stripe-based DWT, block-based DWT and fractional wavelet filter (FrWF). There are various architectures for computing 2D-DWT. Almost all the architectures require arithmetic resources (like adders, multipliers, multiplexers, etc.) and memory element. The conventional row-column DWT requires a large amount of memory for its implementation. The memory required by the architectures of 2D-DWT can be classified into three types namely: transposition memory, temporal memory and frame memory. Transposition memory is used for storing the coefficients obtained after row-wise filtering of the image lines. Temporal memory is needed for storing the partial results generated in the column processing. Frame memory is needed for the storage of the coefficients of LL subband [14], [15]. There has been significant effort for reducing the memory required in computing 2D-DWT which are discussed in the next subsection.

1.2 Literature Survey

The first effort to reduce memory for one-dimensional (1-D) DWT is made in [16]. There are basically two approaches for implementing the 2D-DWT namely convolution scheme and lifting scheme. In comparison to convolution based methods, lifting based methods need less arithmetic resources, perform in-place computation and they have in-built parallelism. However, lifting based

architectures have the drawback that they need a long critical path delay (CPD) [15]. The various popular low memory techniques to implement 2D-DWT can be categorized into three categories namely: Line-based approaches ([9], [17–19]); block-based approaches ([20–22]); and stripe-based approaches ([23–28]). In the line-based approach, the DWT technique is applied on lines of an image. In block-based techniques, the DWT approach is applied on blocks of an image. The stripe-based approach is combines both the line-based and block-based approach and applies the DWT technique on wider image blocks. Corresponding to the three methods of low-memory DWT (i.e. line, block and stripe), there are three different scan methods (i.e. line-based scanning, block-based scanning and stripe-based scanning respectively) used in architectures for reducing the memory requirement.

Line based DWT was initially used in decoder in [29]. Its use in computing DWT coefficients of image both at the encoder and reconstruction at decoder is proposed in [9] and [17]. Architectures based on line-based scanning are proposed in [30–39]. Modified versions of line based scanning methods are used in the architectures proposed in [40–44].

Amongst the line based architectures [30–35], the design of [33] needs memory of the order of $5.5N$ words (where $N \times N$ is the dimension of gray-scale image) and is lowest among all. The transposition and temporal memory of [33] are $2.5N$ and $3N$ respectively. The architectures proposed in [31] and [43], have reduced the transposition memory to a fix size i.e., 3 and 4 words respectively. However, this reduction in transposition memory is achieved at the cost of larger temporal memory (of size $4N$). Even though the architectures proposed in [40] and [41] need a constant transposition memory of 4 words, the temporal memory increases to $5.5N$. The designs proposed in [42] and [43] also have a constant transposition memory size but they require temporal memory of size $4N$. The design in [44] has reduced the temporal memory to $3N$ and also needs less hardware resources. However, it still needs a transposition memory of size N and the reduction in hardware resources is achieved at the expense of higher computational cycles.

Among the line based and modified line based architectures the works in [30], [32], [33], [36], [41], [42] and [43] are based on lifting scheme. Different strategies have been adopted for reducing the CPD in the architectures of [30], [32], [33], [36], [41], [42], [43], [45] and [46] with trade-offs in arithmetic resources, external bandwidth and memory size. The design of [31] and [43] have also achieved a shorter critical path delay of T_m (where T_m is the delay of multiplier). But, this is achieved at the cost of large number of pipeline registers. Among the lifting based architectures ([30], [32], [33], [36], [41], [42], [43], [45] and [46]), all the architectures (except [33]) are not appropriate for parallel processing and VLSI implementation due to their irregular structure [47]. In [47], a modified stripe based scanning method is proposed and based on it a new parallel lifting based DWT architecture is designed. This architecture requires only $3N+24S$ words of memory where S is the number of parallel processing units. Also, the CPD of this architecture is T_m+T_a. Another lifting architecture of 2D-DWT using 5/3 filter based on symmetric mask-based al-

gorithm is proposed in [48]. The benefit of this architecture is that it does not require any temporal memory and its transposition memory is also less than the conventional architectures based on lifting scheme. However, this architecture needs more additions and multiplications than others. Also this method is not fit for 9/7 filter as it demands large mask [14]. A detailed survey of lifting architectures for DWT can be found in [49].

The authors in [36–38] have proposed line based lifting scheme for both forward and inverse DWT. However, the architecture of [37] requires $9N$ storage cells and complexity of the order of $O(N^2)$ clock cycles. Combined line based architecture for 9/7 and 5/3 filter presented in [38] needs more resources.

A dual line scan technique in which two consecutive rows or columns are scanned simultaneously is proposed in [43] and [50] to reduce the internal memory size. The architecture proposed in [43] and [50] need memory only for 2×2 transposing registers. However, for an image of size $N\times N$, these architectures require internal memory size of $4N$.

Block based architectures for DWT are presented in [27], [51–53]. Though the design proposed in [51] has high throughput, it requires large transposition and temporal memories. Furthermore, this design demands a large number of arithmetic resources. Despite the design presented in [52] uses only two row processors, two column processors and two memory modules, yet its memory control logic is complex, which is $O(N^2)$ clock cycles. The architecture proposed in [53] results in reduced internal memory size but at the cost of increased computation time and external bandwidth.

Stripe based DWT architectures are proposed in [14], [15] and [54]. The temporal memory is eliminated in the architecture proposed in [14], but for storing the subband coefficients it requires a large line buffer. Moreover, the memory of this design increases as the length of filter increases. Also for the same throughput, this design requires more arithmetic resources than the designs of [33] and [44]. Moreover, the architecture has a complex control scheme and its level 2 and level 3 processing units have irregular structure. Furthermore, the design of [14] needs more memory and arithmetic resources than the one proposed in [15]. The stripe based architecture of [54] is only for 1D transform. A modified stripe based design is presented in [55], which has overlapping of 8 columns per stripe for 9/7 filter. This results in longer computation time. Another approach for implementing stripe-based DWT is discussed in [26] which does not require overlapping of pixels. However, this method needs a large memory.

For implementing multi-level or higher levels of DWT there are two options namely; folded architecture [51] and pipelined architecture [14], [33]. The folded architectures require a frame memory of $\frac{N^2}{4}$ [15]. This memory size is large and so an external RAM is normally used for on-chip implementations [14]. On the other hand the pipelined architectures need lesser or no frame memory at all. Moreover their throughput is also high. However, the above benefits of pipelined architectures comes at the cost of increased arithmetic resources [15]. A pipeline architecture, without any transposition memory and frame buffer is proposed in [56]. However, the design is based on

non-separable approach for computing 2-D DWT, which is not popular owing to its higher computational requirements than the separable approach of implementing 2-D DWT. Given the same throughput rate, the non-separable approach needs $\frac{M}{2}$ times more computations than the separable 2-D DWT, where $M \times M$ is the order of 2D wavelet filter.

Recently in [57], the authors have developed an architecture for computing multi-level 2-D DWT using dual data scanning scheme (which doubles the throughput per cycle). The authors have combined several 2-D DWT units to make a parallel multi-level architecture, for computing up to six-levels of 2-D DWT. Although this architecture does not require any frame buffer words, it requires $13N$ on-chip storage words. Also the design is slightly larger than that of [35]. The folded architectures of [34] and [39], also compute multi-level 2D-DWT level-by-level. These approaches use serial processing and hence the computation time increases with the increase in level of transform.

Another scanning technique named as interlaced read scan algorithm (IRSA) is proposed in [58], with the aim of decreasing the transpose memory-size of the dual mode lifting DWT structure. The transposing memory of this method with 5/3 filter is $2N$. However, this architecture has a long CPD which may limit its use in real time applications. Recently another multiplier-less 2-D DWT architecture using lifting scheme is proposed in [59]. It proposes a new dual Z-scanning technique which helps in reducing the transposition buffers and latency. This architecture also has the merit that it has a simple data path and control path as compared to the design presented in [34], which needs $4N$ temporal line buffers to compute 2-D DWT of $N \times N$ image.

The work in [60] proposes an architecture for implementing DWT utilizing both canonical signed digit (CSD) and distributed arithmetic. The architecture is multiplierless and requires only 7 adders for its implementation, but it has complex controller. The authors in [61] have proposed a new DWT architecture in which they have replace the CSD multiplier with 16-bit radix 8 booth multiplier. The proposed architecture is fast but requires larger area compared to existing DWT architectures. In [62], an internal folded architecture for implementing multi-level DWT is presented. This architecture is tailored for applications requiring hardware acceleration and optimization. However, this design requires approximately $6N$ words of memory. There are some other efficient architectures of DWT [63, 64], but they are for its three dimensional implementation which is out pf the scope of this work.

Recently Fractional wavelet filter (FrWF) [12], [13] is proposed to compute 2D-DWT of images with minimum memory requirements. For a gray scale image of size 512× 512, FrWF requires only 4.6 kB of random access memory (RAM). Recently, architectures of FrWF [65] (both with multipliers and multiplierless). Although these architectures have reduced the memory requirement to $2N+14$, they are not suitable for calculating transform of HR-images on low-cost sensors nodes. This is because the memory requirement of HR-images by these architectures exceed 10 kB (memory available on low-cost sensor nodes [13]). Moreover, there is repetitive reading of image lines in FrWF which increases its complexity. In this paper we have modified FrWF

and proposed an architecture targeted for computing DWT of HR-images on low-cost sensor nodes used in WMSNs/IoMT.

1.3 Contribution of this paper

In this paper, we propose a novel architecture with low-memory requirements for computing DWT coefficients of an image. To the best of our knowledge, the proposed architecture is one of the best (in terms of power consumption, CPD and area) among the existing state-of-the-art low memory techniques for computing DWT coefficients in which the memory requirement is independent of the image size. A new data scanning technique of reading the pixels is used in this work which exploits the spatial redundancies in the calculation of the transform coefficients. The CPD of proposed architecture is equal to the delay of multiplier (T_m). Furthermore, a multiplierless version of the proposed architecture for 5/3 filter is also presented in this paper. The advantage of the multiplierless version of the proposed architecture is that the CPD is reduced to the delay of an adder (T_a, where $T_a < T_m$). The architecture is implemented in FPGA and the results show that the proposed architecture is better than the other state-of-the-art low memory DWT architectures in terms of CPD and on chip power.

The rest of the paper is organized as follows. In Section 2, conventional DWT and FrWF are briefly reviewed in order to explain the memory issues in their implementation. In Section 3, the proposed architecture along with the scanning technique is discussed in detail. Results and its related discussions are presented in Section 4. Finally the paper is concluded in Section 5.

2 Memory issues in existing DWT

In this section, the conventional approaches of computing the 2D-DWT of images are briefly described and then the memory issue in their implementation is highlighted.

2.1 Conventional DWT

The one dimensional DWT (1D-DWT) of a signal is computed by separately filtering the signal by LPF and HPF followed by downsampling by a factor of 2. The coefficients obtained after filtering by LPF and HPF followed by downsampling are known as approximations and detailed coefficients respectively. For a 1D signal of dimension N, there are $\frac{N}{2}$ approximation coefficients and $\frac{N}{2}$ detail coefficients. For computing 1D-DWT, the downsampling combined with convolution operation can be mathematically represented as given in eqn. (1) and eqn. (2) respectively [13].

$$a(i) = \sum_{j=-\lfloor \frac{n_l}{2} \rfloor}^{j=\lfloor \frac{n_l}{2} \rfloor} x_{2i+j}.l_j \qquad i = 0, 1, ..., \frac{N}{2} - 1 \tag{1}$$

$$d(i) = \sum_{j=-\lfloor \frac{n_h}{2} \rfloor}^{j=\lfloor \frac{n_h}{2} \rfloor} x_{2i+j+1}.h_j \qquad i = 0, 1, ..., \frac{N}{2} - 1 \tag{2}$$

Where l_j and h_j denote the j^{th} element in the impulse response of LPF and HPF coefficients respectively, x_{2i+j} denotes the $(2i+j)^{th}$ sample of the signal and n_l and n_h are the number of coefficients in LPF and HPF respectively. The symbol $\lfloor x \rfloor$ denotes the largest integer less than or equal to x. The symbols $a(i)$ and $d(i)$ denote the i^{th} approximation and detail coefficient respectively. For computing 1D-DWT of an image, all the rows are filtered by LPF and HPF followed by downsampling operation, resulting in two sub-bands namely L and H respectively. If the original image is of dimension $N \times N$, then both the L and H sub-bands are of dimensions $N \times \frac{N}{2}$.

For computing 2D-DWT, 1D-DWT is further applied on all the columns of the L and H sub-bands to obtain the LL, LH, HL and HH sub-bands; each of dimension $\frac{N}{2} \times \frac{N}{2}$. Although the above procedure of computing 2D-DWT of an image is simple, it requires the complete image to be stored in system's RAM. As such conventional DWT needs $8N^2$ bytes (with floating point filter coefficients) for calculating transform of gray-scale image of size $N \times N$ [12]. Thus the conventional approach cannot be used for computing DWT of images on low-cost handheld multimedia devices and low cost visual sensors. Recently an alternate technique named as Fractional wavelet filter (FrWF) is proposed in literature [13], which reduces the memory requirement in computing the DWT of images to such an extent, that an image of size 256×256 (8 bits/pixel) would require only 2.304 kB of RAM. The FrWF technique and the memory requirement in it's implementation is briefly discussed in subsequent subsection.

2.2 Fractional Wavelet Filter (FrWF)

Fractional wavelet filter (FrWF) is an alternative way of computing DWT of images, which uses the concept of Vertical filter area (VFA) in order to reduce the computation memory. The VFA (shown in Fig. 1) is an area selected in the image which covers the same number of rows of the image as the number of coefficients in LPF. The original image and the sub-bands are assumed to be stored in SD-card. For a gray-scale image of dimension $N \times N$, the FrWF uses only three buffers each of size N. From the VFA, one image line at a time is only stored in buffer 's'. Then fractional subband wavelet coefficients $ll(i,j,k)$, $lh(i,j,k)$, $hl(i,j,k)$ and $hh(i,j,k)$ are computed from current image line stored in buffer 's'. For example, the $ll(i,j,k)$ coefficient is calculated as given in eqn. 3 [13].

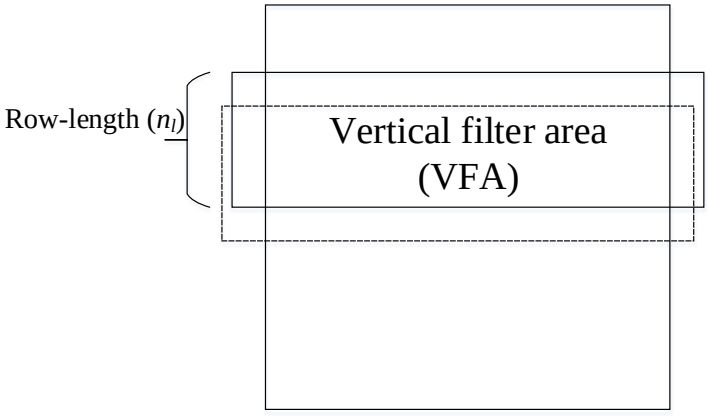

Fig. 1 Schematic representation of Vertical filter area in an image.

$$ll(i,j,k) = l_j . \sum_{m=-\lfloor \frac{n_l}{2} \rfloor}^{m=\lfloor \frac{n_l}{2} \rfloor} s_{2k+m}.l_m \tag{3}$$

Where $i = 0, 1, ..., \frac{N}{2}$-1; is the index of $(i+1)^{th}$ VFA (each VFA contains n_l lines of image) with vertical position of VFA as $2i$; $j = -\lfloor \frac{n_l}{2} \rfloor, ..., \lfloor \frac{n_l}{2} \rfloor$; determines the current input line to be read in buffer 's' as $2i + j$, from the present VFA; $k = 0, 1, ..., \frac{N}{2}$-1 indicates the present position of the filter horizontally within the current VFA; s_{2k+m} is the $(2k+m)^{th}$ sample of the signal; l_m and h_j are the m^{th} and j^{th} LPF and HPF coefficients respectively and n_l is the number of coefficients in LPF.

The fractional sub-band wavelet coefficients of each line in VFA are iteratively summed up to compute the final coefficients of four sub-bands. For example, LL sub-band is obtained by iteratively summing the fractional sub-band wavelet coefficient $ll(i,j,k)$, for all j in VFA according to eqn. 4.

$$LL(i,k) += ll(i,j,k), \forall j \tag{4}$$

Each VFA helps to compute the coefficients of one line of each of the four sub-bands. The VFA is then shifted vertically by two lines to achieve vertical downsampling by a factor of 2, until entire image is covered. For a gray-scale image of dimension $N \times N$, FrWF requires a memory of $9N$ bytes for filters with floating point coefficients and $5N$ bytes for filters with fixed point coefficients [13]. Despite the FrWF requires much smaller memory for its implementation compared to conventional DWT, it's memory requirement depends on image size. Therefore, FrWF may not be suitable for computing DWT of high-resolution images on low-memory platforms. An alternative approach of computing DWT coefficients is to apply the FrWF on segments of an image line as proposed in Segmented FrWF (SFrWF) in [66]. This approach

reduces the memory requirement further below that needed by FrWF but the complexity of this approach is greater than that of FrWF. From complexity point of view SFrWF may not be suitable for low-cost handheld multimedia devices which are limited in terms of memory as well as computation power. Also, in [66] no architecture for SFrWF is presented.

As discussed earlier, the memory requirement of FrWF depends on image size, as it processes and stores one complete image line at a time. One of the objective of the proposed work is that the memory requirement to compute DWT coefficients of image should be independent of image size, so that high-resolution images may be processed in a memory limited environment. In this paper, we propose a new scanning method to scan the pixels within a VFA, such that the DWT coefficients can be computed using fixed size (independent of image size) memory. The novel scanning technique and related discussions are presented in next section.

3 Proposed Architecture

The conventional approach of computing DWT scans the pixels of an image in mortan scan order, i.e. the image lines are read successively one after the other (all pixels of an image line are read at a time), filtered and downsampled to obtain L and H subband coefficients. After this column wise scanning of L and H subband coefficients are done (i.e. columns are read consecutively). This scanning order requires complete image to be stored in system's RAM. In order to reduce the memory requirement, FrWF uses a different scan order. It selects a VFA and within the current VFA, the rows are read in mortan scan order. As discussed earlier, this approach requires one complete image line to be kept in buffer at any time. In order to further reduce the memory requirement and to make it independent of the image size, a novel data scanning technique is proposed in this paper which is discussed next. Also, the architecture of proposed method with and without multipliers are presented subsequently.

3.1 Data Scanning and basic concept

In the proposed method, first a VFA which encloses n_l (equal to the number of coefficients in LPF) number of rows is selected in the image stored in secure digital card (SD-card) as shown in Fig. 1. This VFA is same as that in FrWF. Within this VFA, a horizontal sliding window (HSW) of size $n_l \times n_l$ is selected. In Fig. 2, the scanning order is shown for first HSW (I-HSW). The pixels of the other HSW are also scanned in the same fashion. In order to explain the proposed scanning method, we have considered DWT computation with 5/3 biorthogonal filter ($n_l = 5$) as depicted in Fig. 2. The initial position of HSW is shown as first HSW (I-HSW). Within a HSW, filtering is performed row-wise by aligning the central coefficient of LPF with central pixel of each row and the central coefficient of HPF is aligned with one pixel offset within HSW

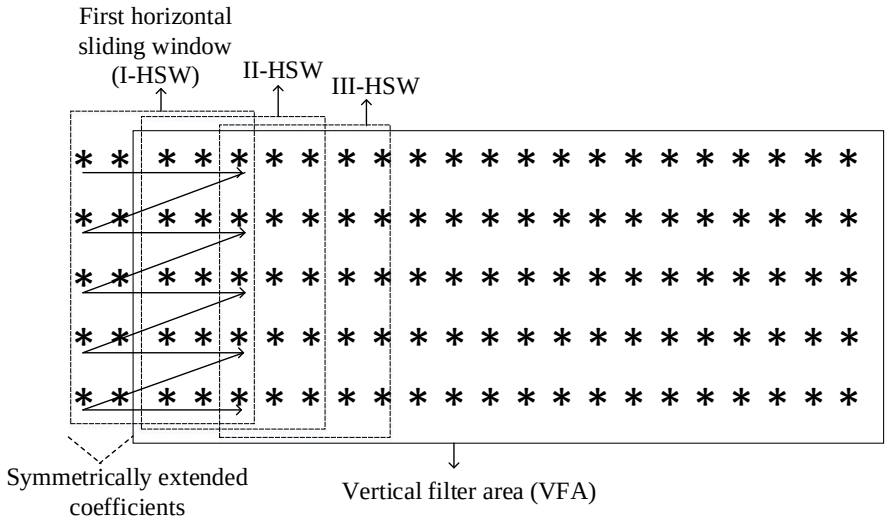

Fig. 2 Data Scanning method for 5/3 filter in a VFA ($*$ represents a pixel).

as shown in Fig. 3. The filtered coefficient of the row thus obtained are then multiplied by corresponding LPF and HPF coefficients and saved in a buffer. The process is repeated on each row in HSW and the values in buffer are successively updated to obtain one coefficient of each of LL, LH, HL and HH subbands according to equations (5)-(8).

$$LL(k,v) = \sum_{j=1}^{j=n_l} l_{j-3}. \sum_{m=-\lfloor \frac{n_l}{2} \rfloor}^{m=\lfloor \frac{n_l}{2} \rfloor} x_{(j,m+2)}.l_m \tag{5}$$

$$LH(k,v) = \sum_{j=1}^{j=n_h} h_{j-3}. \sum_{m=-\lfloor \frac{n_l}{2} \rfloor}^{m=\lfloor \frac{n_l}{2} \rfloor} x_{(j,m+2)}.l_m \tag{6}$$

$$HL(k,v) = \sum_{j=1}^{j=n_l} l_{j-3}. \sum_{m=-\lfloor \frac{n_h}{2} \rfloor}^{m=\lfloor \frac{n_h}{2} \rfloor} x_{(j,m+3)}.h_m \tag{7}$$

$$HH(k,v) = \sum_{j=1}^{j=n_h} h_{j-3}. \sum_{m=-\lfloor \frac{n_h}{2} \rfloor}^{m=\lfloor \frac{n_h}{2} \rfloor} x_{(j,m+3)}.h_m \tag{8}$$

In the above equations l_m and h_m are the m^{th} coefficient of LPF and HPF respectively; n_l and n_h are the number of coefficients in LPF and HPF

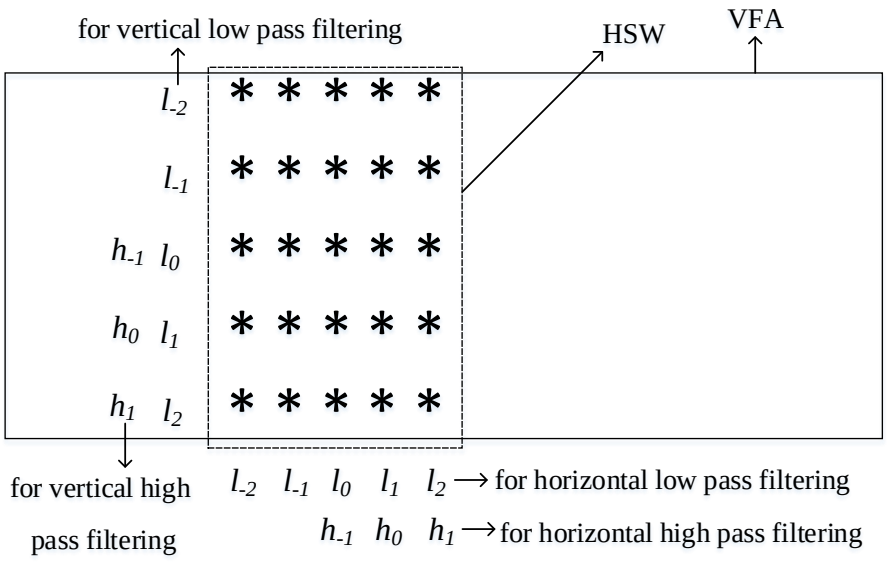

Fig. 3 Alignment of filter coefficients for a HSW.

respectively; $x_{(j,m+2)}$ is the $(m+2)^{th}$ coefficient of j^{th} line of current HSW; $LL(k,v)$, $LH(k,v)$, $HL(k,v)$ and $HH(k,v)$ are the coefficients of LL, LH, HL and HH subbands respectively at the indices (k,v); where $k = 0, 1, ..., \frac{N}{2} - 1$ represents the $(k+1)^{th}$ HSW in a VFA and $v = 0, 1, ..., \frac{N}{2} - 1$ represents the $(v+1)^{th}$ VFA. It should be noted that symmetric extension needs to be done at the image boundaries in order to avoid border effects (as shown in Fig. 2). For example a signal $\mathbf{q}$=$[q_0, q_1, q_2, q_4, q_5, q_6, q_7]$ would be symmetrically extended at boundaries for filtering with LPF of length 5 as $\mathbf{q}_{ext}$=$[q_2, q_1, q_0, q_1, q_2, q_4, q_5, q_6, q_7, q_6, q_5]$.

The best part of the proposed method is that vertical filtering need not be done separately. Rather the values obtained after horizontal filtering within HSW are multiplied by corresponding LPF and HPF coefficients (as shown in Fig. 3). The HSW is slided by 2 pixels after computing the coefficients of the subbands related to that HSW each time, till all the pixels of a VFA are covered. For an image of dimension $N \times N$, the HSW would be shifted $\frac{N}{2}$ times within a VFA. Since for each HSW, one coefficient of each of the four subbands will be computed. Therefore for each VFA, one row of $\frac{N}{2}$ coefficients of each of the four subbands would be calculated. Once the process is completed for a VFA, it is shifted down by two lines (to incorporate vertical downsampling by a factor of 2). The above procedure is repeated for each VFA. Since VFA is also shifted $\frac{N}{2}$ times, applying the above process for each VFA, the $\frac{N}{2}$ rows of each of the four subbands are obtained.

The number of computation cycles depends on the scanning order and is estimated as follows. It is assumed that in one cycle one line from HSW is

read. For reading n_l (n_l is the number of coefficients in LPF) lines of HSW, n_l cycles are needed. The HSW is shifted in the VFA $N/2$ times. This requires a total of $\frac{n_l.N}{2}$ number of clock cycles are required for reading all the pixels in a VFA. The VFA is shifted down $N/2$ times, so a total of $\frac{n_l.N^2}{4}$ number of clock cycles are needed for reading all the pixels of the image. Thus, for 5/3 filter bank, $\frac{5N^2}{4}$ number of cycles are needed.

It should be noted that the image and VFA are assumed to be stored in SD-card similar to the approach used in [12], [13], [15], [27]. A SD-card is a reasonable extension for use in portable devices and sensor nodes, since the image and transformed coefficients has to be stored somewhere. Also, it is hardly feasible to instantly send out the coded transformed coefficients to the network, since there may be network congestion or some other internal ongoing operation with higher priority, therefore there may be a need of storing the coefficients [67]. Only the pixels of HSW are read individually in the scanning order shown in Fig. 2, and multiplied by filter coefficient and then stored in a buffer. The data in the buffer is updated until the final sub-band coefficients are calculated. With each shift of the HSW, subband coefficients are generated which need to be stored. Thus dedicated update buffers for each subband are only required. In this way the temporal memory of the length of the row is not required in our proposed architecture. Moreover, for faster computation the pixels can be read in parallel. However this would result in more memory requirement. The architecture based on the new data scanning technique is discussed in the next subsection.

For better understanding of the proposed architecture, we have added a high-level design flow of the complete process used in this work which is shown in Fig. 4. It should be noted that the original image is stored in external SD-card from which the pixels are read as and when needed. At a time only a row of pixels from the HSW is read, which makes the proposed architecture's memory independent of the image dimension. The pixels are fed to the top level of the architecture as shown in Fig. 5. The subband coefficients computed are saved in the SD-card. The computation of the subband coefficients and detailed discussion of the top level of the architecture are described in next section.

3.2 Proposed low memory Architecture

The top level architecture for 5/3 filter is shown in Fig. 5. The architecture consists of multiple tree blocks, a multiplexer block and a control block. The number of tree blocks depends on the filter size. For a LPF with number of coefficients n_l, $(n_l + 1)/2$ tree blocks will be used. In Fig. 5, T1, T2 and T3 are three tree blocks shown for 5/3 filter. The input data is fed to all the blocks concurrently. The computation inside each block is determined by the control block. The outputs of all the tree blocks are multiplexed. Once the computation of a subband coefficient inside a tree block buffer is complete, it is relayed to the output for storage. Although, the architecture is explained

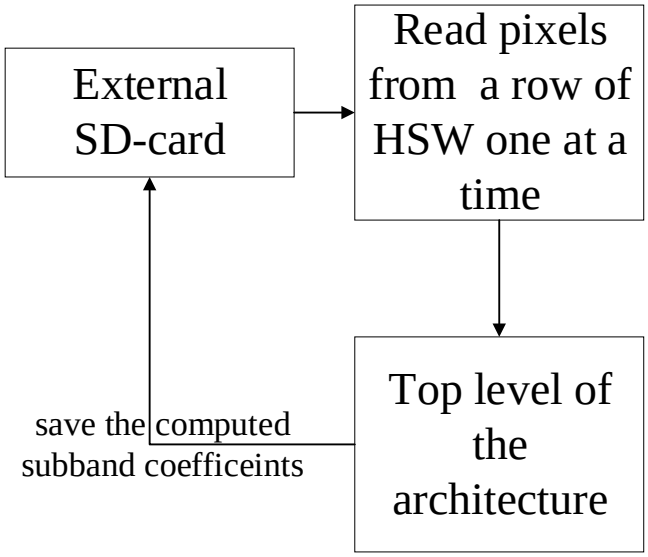

Fig. 4 High level design flow of the complete process.

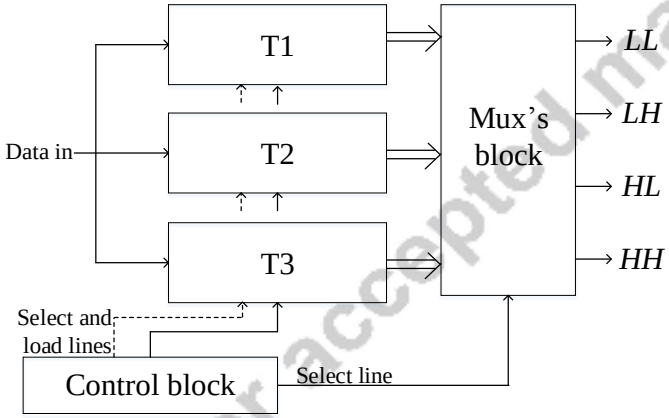

Fig. 5 Top Level of the Architecture

for 5/3 filter, a slight modification in the architecture needs to be done for implementing the proposed architecture with other filters. For 9/7 filter, the number of Tree blocks would be 5. However, the number of B-blocks used in a T tree will be same and the layout of B blocks will also be the same for any filter length.

The various blocks used in the proposed architecture are as follows:

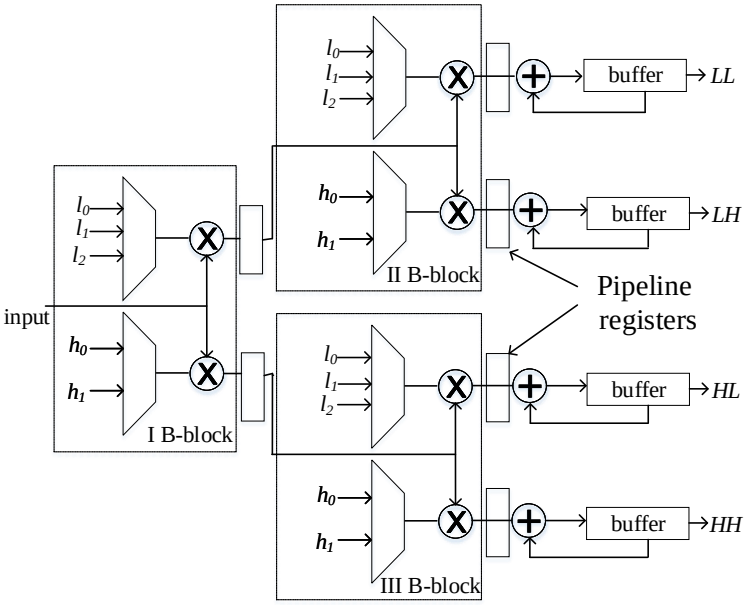

Fig. 6 Schematic view of a Tree block (T-block)

3.2.1 Tree Block

A tree block (T-block) consists of three basic computation block units along with four adders and four registers as shown in Fig. 6. The input data is multiplied twice through the data path and stored in the buffer. By inserting pipeline registers in the path, the critical path delay (CPD) is T_m (delay of a multiplier). Each T block requires 4 words of memory in the form of buffer registers and 6 words for pipeline registers. With each incoming pixel coefficient, the intermediate result is stored in the buffer and updated till the final coefficient of the sub-band is obtained, as discussed in the data scanning method.

3.2.2 Basic computation block

The basic computation block (B Block) consists of multipliers and multiplexers (Mux's) as shown in Fig. 7. A Mux is used to select which filter coefficient is to be multiplied by the current input. (The 5/3 biorthogonal filter is symmetric, hence $l_{-2} = l_2$, $l_{-1} = l_1$, $h_{-1} = h_1$). Two such units are required for concurrent low-pass and high-pass filtering for each input. In Fig. 6, the I B-block is used to obtain the L and H sub-band coefficients. This corresponds to the row-wise 1-D filtering in case of conventional DWT. These L and H sub-band coefficients are fed as inputs to II B-block and III B-block respectively. The input of II B-block is multiplied by LPF and HPF coefficients and updated to compute a coefficient of LL and LH subband (eqn. 5, eqn.6). Likewise the

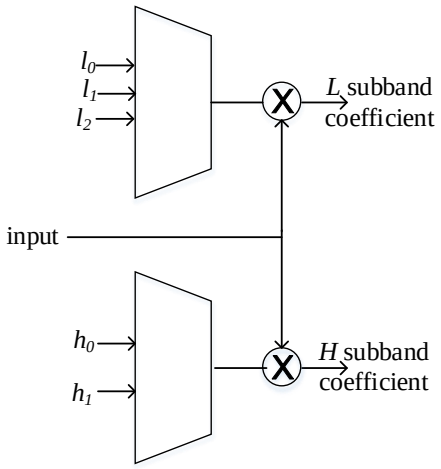

Fig. 7 Schematic view of a basic computation block (B-block)

input of III B-block is multiplied by LPF and HPF coefficients and successively updated to compute a coefficient of HL and HH subband (eqn. 7, eqn.8).

3.2.3 Mux's block

The Mux's block, as shown in Fig. 8, consists of four multiplexers. The output of the three T-blocks are fed as the inputs of these multiplexers. In Fig. 8, LL_{T1}, LH_{T1}, HL_{T1} and HH_{T1} are the outputs of T1 block. The outputs of T2 blocks are LL_{T2}, LH_{T2}, HL_{T2} and HH_{T2}. Similarly, LL_{T3}, LH_{T3}, HL_{T3} and HH_{T3} are the outputs of T3 block. Out of the three coefficients of LL subband, i.e. LL_{T1}, LL_{T2} and LL_{T3}, only one coefficient is selected at a time from the multiplexer and it is stored as a coefficient of LL subband in the SD-card. Likewise one coefficient at a time of LH, HL and HH subbands are selected by the multiplexer to be saved in the SD-card.

As discussed in previous subsection that VFA is shifted by two lines which leads to overlapping of image lines in multiple VFA's, leading to multiple reading of image lines. Thereby increasing the number of computation cycles of the proposed architecture. In order to reduce the number of computation cycles, the pixels can be read in parallel. The parallel processing would result in faster computations. However, the hardware resources will increase by a factor of S (number of parallel processing units). Thus there exists a trade-off between hardware resources and speed. Furthermore, as seen from Fig. 7, the B-block uses multipliers, which increases the CPD. The use of multipliers can be avoided using the concept of shift and add if the filters used have integer multiplications and divisions. The multiplierless implementation of the proposed architecture with 5/3 filter bank is presented in the next subsection.

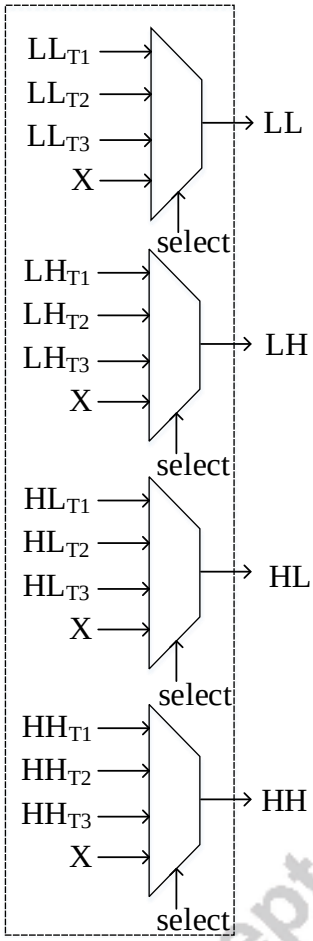

Fig. 8 Schematic view of Mux's block.

3.3 Multiplierless architecture

The proposed architecture can also be implemented without the use of multipliers. The benefit of multiplierless approach is that it reduces the critical path delay (CPD) from T_m to T_a, $T_a < T_m$ (where T_a and T_m are the delay of an adder and a multiplier respectively). Since coefficients of 5/3 filter bank, given in Table 1, have integer multiplication and division, they can be obtained by shift and add operations only. Using the concept of shift and add, the computation block discussed earlier can be simplified as shown in Fig. 9, for low pass 5/3 filter. Similar implementation can be done for 5/3 HPF coefficients.

Table 1 5/3 Filter Coefficients

LPF coefficients	Value	HPF coefficients	Value
l_{-2}	$-\frac{1}{8}$	h_{-2}	0
l_{-1}	$\frac{2}{8}$	h_{-1}	0
l_0	$\frac{6}{8}$	h_0	$-\frac{1}{2}$
l_1	$\frac{2}{8}$	h_1	1
l_2	$-\frac{1}{8}$	h_2	$-\frac{1}{2}$

4 Results and discussion

In this section, implementation of the proposed architectures and hardware resources are presented. For comparison purpose, one of the best existing state-of-the-art low memory 2D-DWT architecture proposed in [47] and our architectures are modeled in HDL and the post implementation results are compared on a common platform xczu6cg-ffvb 1156 board (Zynq UltraScale+) FPGA. Input pixel width used is 8 bits and datapath width is 16 bits. The designs are synthesized for gray-scale standard test images of dimension ranging from 256×256 to 2048×2048. The image of dimension 256×256 is '*Cameraman*', of dimension 512×512 is '*Lena*', of dimension 1024×1024 is '*Man*'and of dimension 2048×2048 is '*Bike*'. The power utilization is measured at the same operating frequency for both the architectures by implementing the architectures in Vivado. The proposed architectures are also compared with other state-of-the-art low memory DWT architectures in terms of various hardware parameters.

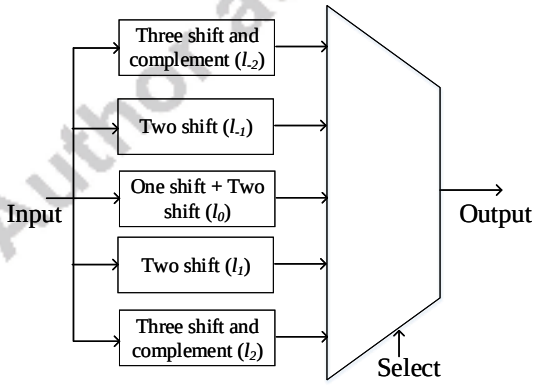

Fig. 9 Multiplierless block for 5/3 LPF.

Table 2 Hardware utilization and power consumption for different image sizes

Parameters	Proposed multiplierless			Hu. et. al [47] (S=4)			
	$S=1$	$S=2$	$S=4$	256×256	512×512	1024×1024	2048×2048
LUT	502	1032	1935	1665	2057	2907	4539
LUT-RAM	-	-	-	392	784	1632	3264
FF	318	636	1272	1776	2032	2548	3572
DSP	-	-	-	24	24	24	24
Total power	0.691W	0.799W	0.875W	0.925W	0.942W	0.957W	0.972W
Logic delay	0.67 ns	0.67 ns	0.67 ns	3.211 ns	3.211ns	3.211 ns	3.211 ns

Table 3 Comparison between existing and proposed architectures for one level 2D-DWT.

Archit-ecure	**Number of slice registers**	**Number of slice LUTs**	**CPD**
Rafi et. al. [68]	511	433	3.040 ns
Proposed	450	410	0.67 ns

4.1 Hardware Estimation

As already discussed in Fig. 5, the proposed architecture to compute 2D-DWT using 5/3 filter consists of 3 T-blocks, a Mux Block and a control block. The Mux block consists of 4 multiplexers, one each to select LL, LH, HL and

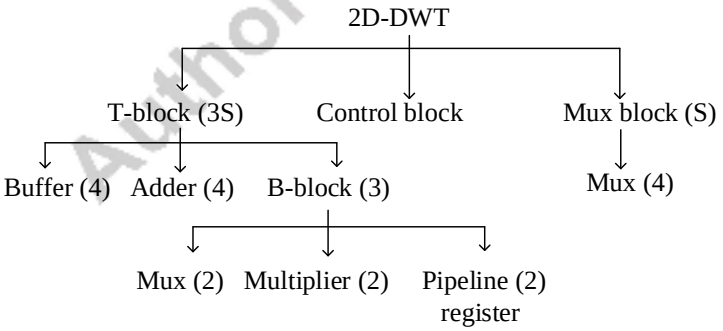

Fig. 10 Hardware resource tree.

Table 4 Comparison between existing and proposed architectures for one level 2D-DWT. S represents the number of parallel processing units.

Architecure	**Number of adders**	**Number of multipliers**	**Memory**	**Computation Cycles**	**Critical path delay (CPD)**
Zhang 2012 et. al. [56]	16	10	$4N + 37$	$N^2/2$	T_m
Mohanty et. al. [44]	16S	9S	$4N + 20S$	$N^2/2S$	$T_m + 2T_a$
Hu et. al. [47]	16S	10S	$3N + 24S$	$N^2/2S$	$T_m + T_a$
Goran et. al. [69]	22	17	$4N$	N^2	$T_m + T_a$
Rafi et. al. [68]	16	8	$4N$+25	$N^2/2 + N$+7	$2T_a$
Zhang 2016 et. al. [70]	32	20	$13N$	$N^2/2$	T_m
Darji et. al. [71]	12	6	$6.5N$	$N^2/2 + N$	$2T_a + T_m$
Proposed	$12S$	$18S$	$30S$	$5N^2/4S$	T_m
Proposed (Multiplierless)	$30S$	0	$30S$	$5N^2/4S$	T_a

HH subband coefficients for storage in SD-card from the corresponding outputs of the T-blocks. If parallel processing is incorporated in this architecture, the number of T-blocks would then be $3S$ and number of Mux blocks would be S (where S is the number of parallel processing units). Furthermore, each T-block consists of 3 B-blocks, 4 adders and 4 buffers (refer Fig. 6) and each B-block has 2 Mux, 2 multiplier and 2 pipeline register (refer Fig. 7). The hierarchical tree for estimating hardware of the proposed architecture is shown in Fig. 10. The number of adders and multipliers are $12S$ and $18S$ respectively and total number of registers in the data path are $30S$ ($12S$ buffers in $3S$ T-blocks and $18S$ pipeline registers from $9S$ B-blocks). The hardware resources in terms of number of adders and number of multipliers and number of registers (memory) of the proposed architectures are listed and compared with other state-of-the-art low memory DWT architectures in first three columns of Table 4.

4.2 Performance Comparison

Table 2 compares different parameters of the proposed multiplierless architecture with one of the best existing low-memory state-of-the-art DWT architecture [47] by actually implementing both the architectures on a common platform xczu6cg-ffvb 11156 board (Zynq UltraScale+) FPGA. The results shown are for gray-scale images of dimension ranging from 256×256 up-to 2048×2048. Furthermore, the proposed multiplierless architecture is implemented for S =1, S =2 and S =4, and the architecture of [47] is implemented for S =4. It is worth mentioning here that the parameters (mentioned in Table 2) of proposed multiplierless architecture is independent of image size. Whereas, the parameters of Hu. et. al [47] increases as the image dimension increases. As we target high-resolution (HR) images, the results of image size 2048×2048 and for S =4 are compared.

From Table 2, it is clear that the proposed multiplierless architecture (S =4) requires 57.37% less look up tables (LUT) and 64.39% less flip-flops (FF) than corresponding values of Hu. et. al [47] for 2048×2048 image dimension. The proposed architecture needs no LUTRAM and DSP, whereas the architecture of [47] requires 3264 LUTRAM and 24 DSP's. The proposed architecture is also superior to the architecture in [47] in terms of logic delay. The logic delay of the proposed architecture is 79.13% less than that of [47]. Also the proposed architecture needs less power than the architecture of [47]. It should be noted that the power requirement of proposed architecture is independent of image dimension, whereas it increases with image dimension for the architecture of [47]. Thus the multiplierless architecture has an edge over the architecture of [47].

The Table 3 compares the number of slice registers, slice LUTs, and CPD of proposed multiplierless architecture with that of [68]. It can be inferred from this table that the proposed architecture requires less number of slice registers and slice LUTs compared to the architecture of [68]. Moreover, the CPD of proposed architecture is less than that of [68].

From Table 4, it can be observed that the memory requirement of the proposed architectures are the lowest compared to the other considered architectures. The best part of the proposed architectures is that their memory requirement is independent of the image dimension. So our proposed architectures can be used for computing DWT coefficients of even HR images on low-cost handheld multimedia devices/VSN/IoMT. On the other hand, the memory requirement of the other architectures are dependent on the image size and may not be suitable for computing transform coefficients of HR images on low-cost memory-limited handheld multimedia devices/VSN/IoMT.

It can be observed from Table 4 that the proposed architecture with multipliers needs less number of adders than the other considered architectures and some more multipliers than the other architectures. However, the proposed multiplierless architecture requires no multiplier but it needs more adders than other architectures. From implementation point of view, implementing an adder is much simpler than a multiplier.

It can also be inferred from Table 4 that the CPD of the proposed architecture is same as that of [56] and [70] but less than other existing architectures. Furthermore, the CPD of the proposed multiplierless architecture is least among all the architectures mentioned in Table 4. The reduction in the CPD to T_a also bolsters the balancing of effective computation time as the operating frequency can be scaled up by several factors and can be chosen depending upon the application. If the CPD is less then the operating frequency can be increased.

Most of the architectures receive input pixels in parallel fashion and therefore require less computational cycles. Our design's single unit, reads pixels in serial and due to overlapping in VFA the computational complexity is high. Using multiple modules and reading pixels in parallel, computational time of our design can also be reduced. In contrast, due to low hardware requirement of each individual module, the proposed architecture provides more flexibility which makes it suitable for different applications. The time and area-on-chip can be traded-off as per the application and the independence of memory of the image size provides a big plus in scalability.

5 Conclusion

High memory consumption is one of the main constraint in implementing 2D-DWT of images on low-cost visual sensors used in IoMT. In this work, we have focused on reducing the memory requirement in computation of 2D-DWT coefficients and proposed architectures for the same. The proposed architectures do not require any temporal memory for their implementation. Furthermore, memory requirement of the proposed architectures are independent of image size. This makes it easy for transforming even high-resolution images by DWT using these architectures. The multiplierless version of the architecture reduces the CPD to T_a, making it the smallest among the state-of-the-art low memory 2D-DWT architectures. The efficiency of the proposed architectures are evaluated and compared with other existing state-of-the-art low memory DWT architectures. The results clearly show that the proposed architectures are at par with other state-of-the-art low memory DWT architectures in terms of number of arithmetic resources needed and superior to them in terms of memory requirement and CPD. Furthermore, the proposed multiplierless architecture needs less on-chip power and has lower logic delay than one of the best existing state-of-the-art DWT architecture. These features make the proposed architecture suitable for transforming HR images on low-cost portable devices and VSN. An interesting future research direction is to use artificial intelligence to decide the number of parallel processing units to be used according to the application requirement.

Conflict of interest

The authors declare that they have no conflict of interest.

References

1. S. Rani, S.H. Ahmed, R. Talwar, J. Malhotra, H. Song, IEEE Internet of Things Journal **4**(3), 832 (2017). DOI 10.1109/JIOT.2017.2671460
2. P. Kaur, R. Kumar, M. Kumar, Multimedia Tools and Applications **78**(14), 19905 (2019)
3. B.D. Martino, M. Rak, M. Ficco, A. Esposito, S. Maisto, S. Nacchia, Internet of Things **1-2**, 99 (2018). DOI https://doi.org/10.1016/j.iot.2018.08.008
4. S. AlZu'bi, B. Hawashin, M. Mujahed, Y. Jararweh, B.B. Gupta, Multimedia Tools and Applications **78**(20), 29581 (2019)
5. J. Lin, W. Yu, N. Zhang, X. Yang, H. Zhang, W. Zhao, IEEE Internet of Things Journal **4**(5), 1125 (2017). DOI 10.1109/JIOT.2017.2683200
6. B. Tavli, K. Bicakci, R. Zilan, J.M. Barcelo-Ordinas, Multimedia Tools and Applications **60**(3), 689 (2012)
7. F. Javed, M.K. Afzal, M. Sharif, B. Kim, IEEE Communications Surveys Tutorials **20**(3), 2062 (2018). DOI 10.1109/COMST.2018.2817685
8. A.K. Abdulrahman, S. Ozturk, Multimedia Tools and Applications **78**(12), 17027 (2019)
9. C. Chrysafis, A. Ortega, IEEE Transactions on Image Processing **9**(3), 378 (2000). DOI 10.1109/83.826776
10. A. Madanayake, R.J. Cintra, V. Dimitrov, F. Bayer, K.A. Wahid, S. Kulasekera, A. Edirisuriya, U. Potluri, S. Madishetty, N. Rajapaksha, IEEE Circuits and Systems Magazine **15**(1), 25 (2015). DOI 10.1109/MCAS.2014.2385553
11. T. Çelik, H. Özkaramanlı, H. Demirel, Computer Vision and Image Understanding **111**(2), 229 (2008). DOI https://doi.org/10.1016/j.cviu.2007.12.001
12. S. Rein, M. Reisslein, Ad Hoc Networks **9**(4), 482 (2011). DOI https://doi.org/10.1016/j.adhoc.2010.08.004
13. S. Rein, M. Reisslein, IEEE Communications Surveys Tutorials **13**(2), 291 (2011). DOI 10.1109/SURV.2011.100110.00059
14. B.K. Mohanty, P.K. Meher, IEEE Transactions on Circuits and Systems for Video Technology **23**(2), 353 (2013). DOI 10.1109/TCSVT.2012.2203745
15. Y. Hu, C.C. Jong, IEEE Transactions on Signal Processing **61**(20), 4975 (2013). DOI 10.1109/TSP.2013.2274640
16. M. Vishwanath, IEEE Transactions on Signal Processing **42**(3), 673 (1994). DOI 10.1109/78.277863
17. J. Oliver, M.P. Malumbres, IEEE Transactions on Circuits and Systems for Video Technology **18**(2), 237 (2008). DOI 10.1109/TCSVT.2007.913962
18. L. Ye, J. Guo, B. Nutter, S. Mitra, Optical Engineering **50**(2), 027005 (2011). DOI 10.1117/1.3541802
19. L. Ye, J. Guo, B. Nutter, S. Mitra, in *Proceedings of Data Compression Conference, 2007. DCC '07* (2007), pp. 213–222. DOI 10.1109/DCC.2007.55
20. Y. Chung-Hsien, W. Jia-Ching, W. Jhing-Fa, C.W. Chang, IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences **90**(5), 1062 (2007). DOI 10.1093/ietfec/e90-a.5.1062
21. Y. Bao, C.C.J. Kuo, IEEE Transactions on Circuits and Systems for Video Technology **11**(5), 642 (2001). DOI 10.1109/76.920193
22. C.K. Hu, W.M. Yan, K.L. Chung, Signal processing **84**(9), 1689 (2004). DOI 10.1016/j.sigpro.2004.05.014
23. L.W. Chew, W.C. Chia, L.m. Ang, K.P. Seng, EURASIP journal on embedded systems **2009**, 1 (2009). DOI 10.1155/2009/479281
24. L.W. Chew, W.C. Chia, L.M. Ang, K.P. Seng, International Journal of Sensor Networks **11**(1), 33 (2012). DOI 10.1504/IJSNET.2012.045033
25. W.C. Chia, L.W. Chew, L.M. Ang, K.P. Seng, International Journal of Sensor Networks **11**(1), 22 (2012). DOI https://doi.org/10.1504/IJSNET.2012.045037
26. M.Y. Chiu, K.B. Lee, C.W. Jen, in *Proceedings of 2003 IEEE Workshop on Signal Processing Systems (IEEE Cat. No.03TH8682)* (2003), pp. 177–182. DOI 10.1109/SIPS.2003.1235665
27. L. Ye, Z. Hou, IEEE Transactions on Circuits and Systems for Video Technology **25**(11), 1773 (2015). DOI 10.1109/TCSVT.2015.2400776

28. R.K. Bhattar, K. Ramakrishnan, K. Dasgupta, Signal Processing: Image Communication **17**(6), 441 (2002). DOI https://doi.org/10.1016/S0923-5965(02)00019-X
29. P. Cosman, K. Zeger, IEEE Signal Processing Letters **5**(9), 221 (1998). DOI 10.1109/97.712104
30. H. Liao, M.K. Mandal, B.F. Cockburn, IEEE Transactions on Signal Processing **52**(5), 1315 (2004). DOI 10.1109/TSP.2004.826175
31. C.T. Huang, P.C. Tseng, L.G. Chen, IEEE Transactions on Circuits and Systems for Video Technology **15**(7), 910 (2005). DOI 10.1109/TCSVT.2005.848307
32. B.F. Wu, C.F. Lin, IEEE Transactions on Circuits and Systems for Video Technology **15**(12), 1615 (2005). DOI 10.1109/TCSVT.2005.858610
33. B.K. Mohanty, P.K. Meher, IEEE Transactions on Signal Processing **59**(5), 2072 (2011). DOI 10.1109/TSP.2011.2109953
34. Y.H. Seo, D.W. Kim, IEEE Journal of Solid-State Circuits **42**(2), 431 (2007). DOI 10.1109/JSSC.2006.889368
35. X. Tian, L. Wu, Y.H. Tan, J.W. Tian, IEEE Transactions on Computers **60**(8), 1207 (2011). DOI 10.1109/TC.2010.178
36. X. Lan, N. Zheng, Y. Liu, IEEE Transactions on Consumer Electronics **51**(2), 379 (2005). DOI 10.1109/TCE.2005.1467975
37. W.H. Chang, Y.S. Lee, W.S. Peng, C.Y. Lee, in *Proceedings of the 2001 IEEE International Symposium on Circuits and Systems ISCAS 2001 (Cat. No.01CH37196)*, vol. 4 (2001), vol. 4, pp. 330–333. DOI 10.1109/ISCAS.2001.922239
38. G. Dillen, B. Georis, J.D. Legat, O. Cantineau, IEEE Transactions on Circuits and Systems for Video Technology **13**(9), 944 (2003). DOI 10.1109/TCSVT.2003.816518
39. K. Mei, N. Zheng, C. Huang, Y. Liu, Q. Zeng, IEEE Transactions on Circuits and Systems for Video Technology **17**(8), 1065 (2007). DOI 10.1109/TCSVT.2007.903555
40. C.Y. Xiong, J.W. Tian, J. Liu, IEEE Transactions on Circuits and Systems for Video Technology **16**(2), 309 (2006). DOI 10.1109/TCSVT.2005.860121
41. C. Xiong, J. Tian, J. Liu, IEEE Transactions on Image Processing **16**(3), 607 (2007). DOI 10.1109/TIP.2007.891069
42. W. Zhang, Z. Jiang, Z. Gao, Y. Liu, IEEE Transactions on Circuits and Systems II: Express Briefs **59**(3), 158 (2012). DOI 10.1109/TCSII.2012.2184369
43. Y.K. Lai, L.F. Chen, Y.C. Shih, IEEE Transactions on Consumer Electronics **55**(2), 400 (2009). DOI 10.1109/TCE.2009.5174400
44. B.K. Mohanty, A. Mahajan, P.K. Meher, IEEE Transactions on Circuits and Systems II: Express Briefs **59**(7), 434 (2012). DOI 10.1109/TCSII.2012.2200169
45. C.T. Huang, P.C. Tseng, L.G. Chen, IEEE Transactions on Signal Processing **52**(4), 1080 (2004). DOI 10.1109/TSP.2004.823509
46. G. Shi, W. Liu, L. Zhang, F. Li, IEEE Transactions on Circuits and Systems II: Express Briefs **56**(4), 290 (2009). DOI 10.1109/TCSII.2009.2015393
47. Y. Hu, C.C. Jong, IEEE Transactions on Circuits and Systems II: Express Briefs **60**(8), 502 (2013). DOI 10.1109/TCSII.2013.2268335
48. C.H. Hsia, J.M. Guo, J.S. Chiang, IEEE Transactions on Circuits and Systems for Video Technology **19**(8), 1202 (2009). DOI 10.1109/TCSVT.2009.2020259
49. T. Acharya, C. Chakrabarti, Journal of VLSI signal processing systems for signal, image and video technology **42**(3), 321 (2006). DOI 10.1007/s11266-006-4191-3
50. J. Song, I.C. Park, IEEE Transactions on Circuits and Systems II: Express Briefs **56**(12), 916 (2009). DOI 10.1109/TCSII.2009.2035257
51. C. Cheng, K.K. Parhi, IEEE Transactions on Signal Processing **56**(1), 393 (2008). DOI 10.1109/TSP.2007.900754
52. K. Andra, C. Chakrabarti, T. Acharya, IEEE Transactions on Signal Processing **50**(4), 966 (2002). DOI 10.1109/78.992147
53. B.F. Wu, C.F. Lin, IEEE Transactions on Circuits and Systems II: Express Briefs **53**(4), 304 (2006). DOI 10.1109/TCSII.2005.862042
54. W. Jiang, A. Ortega, IEEE Transactions on Circuits and Systems for Video Technology **11**(5), 651 (2001). DOI 10.1109/76.920194
55. C.T. Huang, P.C. Tseng, L.G. Chen, IEEE Transactions on Signal Processing **53**(4), 1575 (2005). DOI 10.1109/TSP.2005.843704
56. C. Zhang, C. Wang, M.O. Ahmad, IEEE Transactions on Circuits and Systems I: Regular Papers **59**(8), 1775 (2012). DOI 10.1109/TCSI.2011.2180432

57. Y. Zhang, H. Cao, H. Jiang, B. Li, Journal of Visual Communication and Image Representation **38**, 297 (2016). DOI https://doi.org/10.1016/j.jvcir.2016.03.014
58. C.H. Hsia, J.S. Chiang, J.M. Guo, IEEE Transactions on Circuits and Systems for Video Technology **23**(4), 671 (2013). DOI 10.1109/TCSVT.2012.2211953
59. A. Darji, A. R., S.N. Merchant, A. Chandorkar, IET Computers Digital Techniques **9**(2), 113 (2015). DOI 10.1049/iet-cdt.2013.0167
60. S.E. Iftikhar Gardezi, F. Aziz, S. Javed, C.J. Younis, M. Alam, Y. Massoud, in *Proceedings of the 2019 16th International Bhurban Conference on Applied Sciences and Technology (IBCAST)* (2019), pp. 548–552
61. G.K. Kumar, N. Balaji, K.S. Reddy, V. Thanuja, International Journal of Intelligent Engineering & Systems **12**(3), 148 (2019). DOI 10.22266/ijies2019.0630.16
62. W. Zhang, C. Wu, P. Zhang, Y. Liu, Applied Sciences **9**(21), 1 (2019)
63. S. Divakara, S. Patilkulkarni, C.P. Raj, International Journal of Image and Graphics **20**(03), 2050017 (2020)
64. P. Kannan, I. Asma, A. Benasir, R. Deepikha, R. Divya, International Research Journal of Engineering and Technology (2019)
65. M. Tausif, A. Jain, E. Khan, M. Hasan, in *Proceedings of the 17th IEEE Conference on Sensors (IEEE Sensors 2018)* (2018)
66. M. Tausif, E. Khan, M. Hasan, M. Reisslein, in *Proceedings of 4th IEEE Uttar Pradesh Section International Conference on Electrical, Computer and Electronics (UPCON)* (2017), pp. 593–597. DOI 10.1109/UPCON.2017.8251116
67. S. Rein, S. Lehmann, C. Gühmann, in *Proceedings of 4th International ICST Mobile Multimedia Communications Conference, MobiMedia* (2008). DOI 10.4108/ICST.MOBIMEDIA2008.4026
68. M. Rafi, N. Din, International Journal of Advances in Telecommunications, Electrotechnics, Signals and Systems **6**(1), 24 (2017). DOI 10.11601/ijates.v6i1.202
69. G. Savić, V. Rajović, Journal of Circuits, Systems and Computers **28**(7), 1950118 (2018). DOI 10.1142/S0218126619501184
70. Y. Zhang, H. Cao, H. Jiang, B. Li, Journal of Visual Communication and Image Representation **38**, 297 (2016). DOI https://doi.org/10.1016/j.jvcir.2016.03.014
71. A.D. Darji, S.S. Kushwah, S.N. Merchant, A.N. Chandorkar, EURASIP Journal on Image and Video Processing **2014**(1), 1 (2014)