Multidimensional Systems and Signal Processing (2021) 32:821–844
https://doi.org/10.1007/s11045-020-00759-4

Efficient FPGA architecture of optimized Haar wavelet transform for image and video processing applications

Sayantam Sarkar[1] · Satish S. Bhairannawar[2]

Received: 21 April 2020 / Revised: 17 December 2020 / Accepted: 25 December 2020 /
Published online: 29 January 2021

Abstract
Discrete Wavelet Transform (DWT) is widely used in digital image and video processing due to its various advantages over other similar transform techniques. In this paper, efficient hardware architecture of *Optimized Haar Wavelet Transform* is proposed which is modeled using *Optimized Kogge–Stone Adder/Subtractor, Optimized Controller, Buffer, Shifter* and *D_FF* blocks. The existing Kogge–Stone Adder architecture is optimized by using *Modified Carry Correction* block which uses parallel architecture to reduce the computational delay. Similarly, the *Controller* block is optimized by using *Clock Dividers* and *Reset Counter* inter-dependently. To preserve the accuracy of the processed data, suitable size of intermediate bits in fractional format with the help of *Q-notation* is considered. The comparison results show that the proposed architecture performs better than existing ones concerning both hardware utilization and data accuracy.

Keywords FPGA Implementations · Image/Video Processing · Modified Carry Correction · Optimized Controller · Optimized Haar Wavelet Transform · Optimized Kogge–Stone Adder/Subtractor

1 Introduction

Digital Image Processing (DIP) is used in almost every fields known by today's modern human society such as medical, astronomy, entertainment and computer vision (Gonzalez and Woods 2008; Jayaraman et al. 2009) etc. Video Processing is the extension of the digital image processing (Marques 2012) where a sequence of still images are changing at very fast rate with proper sequences. This makes illusion to the viewer that the objects present in the

✉ Sayantam Sarkar
sayantam.61@gmail.com

Satish S. Bhairannawar
satishbhairannawar@gmail.com

1 Department of Electronics and Communication Engineering, Vijaya Vittala Institute of Technology, Bangalore, Karnataka 560077, India

2 Department of Electronics and Communication Engineering, Shri Dharmasthala Manjunatheshwara College of Engineering and Technology, Dharwad, Karnataka 580002, India

frame are moving. In the case of the video, each still image is known as frame and the rate at which the frame changes are calculated in *frames per second (fps)* unit. As a result, image processing techniques (Jayaraman et al. 2009) can also be used in video processing. For good quality image, the number of the pixels present in the corresponding image must be high and similarly for video both number of pixels in the frame and frame rate must be high.

In the field of the image and video processing, wavelet transforms are widely used in various applications due to its various advantages over other similar kind of transformation techniques (Vaidyanathan 1993; Bhairannawar et al. 2018b). *Image compression, image denoising, image fusion and image recognition* are the most used applications of wavelet transforms. (i). *Image Compression:* Any wavelet transform can be able to separate the redundant components present in the image. As a result the size of the generated bands are reduced. Here the bands generated by low pass filter consists of the compressed version of the input image directly. As a result, the wavelet transform can be used for image compression purpose (Nashat and Hussain 2016; Rajasekhar et al. 2014). (ii). *Image Denoising:* The filters used in any wavelet transforms are designed using *Perfect Reconstruction (PR)* condition (Mallat 2008) which introduces de-noising capabilities within the mathematical model of wavelet transform itself. So, the wavelet transform can be used for image de-noising (Aravind and Suresh 2015; Gupta et al. 2013) also. (iii). *Image Fusion:* Merging of two images into a single composite image is known as image fusion. In most of the existing fusion techniques, wavelet transforms are used with different types of fusion equations to retain the property of both images (Aishwarya et al. 2016; Zhang and Zhang 2015). (iv). *Image Recognition:* This is used to identify some predefined objects present in input image. Wavelet transform techniques are used in most of the existing recognition algorithms (Elakkiya and Audithan 2014; Alsubari et al. 2017).

To implement real time high speed discrete wavelet transform, it is essential to use hardware level implementation techniques. But most of the existing wavelet transform techniques are implemented on either software or embedded system based hardware techniques which are not suitable for real time high speed applications mainly due to the lower processing speed. Moreover some ASIC based hardware architectures were presented for the same which uses complex architectural model and is not suitable for real time high speed applications due to the architectural complexcity.

Contributions *The novel concepts of this paper are listed as follows:*

(i). Optimized the Kogge–Stone Adder architecture using Modified Carry Correction block.

(ii). The novel Clock Dividers block is used to generate different frequencies required to synchronize the filtered data at output side in optimized way.

(iii). The novel Reset Controller block is used to discard the overlapped data at output side in efficient manner to generate proper output coefficients.

2 Related works

The existing techniques of *Adder* and *Haar Wavelet Transform* along with its advantages and disadvantages are discussed briefly in this section.

2.1 Adders

Addition is a commonly used mathematical operation in most of the digital implementations. *Ripple Carry Adder* (Koyada et al. 2017) is the basic architecture normally used to implement addition operation in digital logic which works fine for small number of bits, but to implement adders with large number of bits, delay will be major issue which is proportional to the number of bits. To overcome from this delay problem, various types of adder architectures are introduced which are generally categoried as *Fast Adders* (Smith et al. 2004). The *Carry Select Adder* (Tyagi 1993) is the modified version of the *Ripple Carry Adder* where the carry propagation time is reduced. This architecture generates separate *Sum* and *Carry* for each bit for all possible combinations of *Cin* (i.e., 0 and 1 respectively) using separate *Ripple Carry Adder* blocks. The corrected *Sum* and *Carry* values are then selected by *Multiplexer* with the help of the *Carry* signal generated at respective previous stages. The main disadvantage of this architecture is the requirement of large area and power. To reduce the carry propogation without increasing the area requirement upto a great extent, *Carry LookAhead Adder* (Lee and Oklobdzija 1990) is introduced in which the intermediate stages are calculated separately using *Carry Propogation* and *Carry Generation* logics regardless of the previous level of Carry. The Sum is then calculated using the *Carry Propogation* and *Carry Generation* values with the help of the corresponding previous level Carry. Some extra hardware components are needed to implement *Carry Propogation* and *Carry Generation* equations which increases the area requirements upto some small extent. Instead of immediate transfer of the previous stage carry to the next stage, the *Carry Save Adder* (Vamsi et al. 2018) saves the Carry and then add it with the next Sum generated by the architecture using simple *Full Adder* (Koyada et al. 2017) circuit. This architecture is similar to Ripple Carry Adder where the stored Sum and Carry of each bits are added seperately. For smaller number of bits (i.e., 4 or less), the amount of delay generated by the Ripple Carry Adder is almost negligible. As a result, in the case of *Carry Skip Adder* (Arora and Niranjan 2017), the total number of bits are added by considering a finite number of Ripple Carry Adder with smaller bit size. The error introduced in the intermediate block of carry is corrected by AND gate. To get proper *tradeoff* between hardware parameters, it is crucial to select proper size of intermediate Ripple Carry Adder. The *Carry LookAhead Adder* is arranged in *parallel-prefix form* to get adder architecture with good tradeoff between different parameters, known as *Kogge–Stone Adder* (Xiang et al. 2018; Kogge and Stone 1973). Among all the existing adder architectures, *Kogge–Stone Adder* shows better performance in speed with moderate area utilizations (Koyada et al. 2017) making it suitable to be used in high speed and area efficient architectures. Soares et al. (2019) presented approximation based adder architecture which is then used to design efficient multiplier architecture. In this case, the overall carry propogation signals are divided into a finite number of blocks depending upon the generated values which are then segregated and approximated. This reduces overall area and power consumption with some decrement in output accuracy. This architecture is implemented using 45-nm standard cell based ASIC technique. Mohammadi et al. (2010) presented power and area efficient fault tolerant adder architecture. To achieve this, Berger Code checker with multivalued logic in current mode is used which can detect faults more effectively. This architecture is implemented using 90-nm ASIC technology. The large delay time to detect and correct error is the main drawback of this architecture.

2.2 Haar wavelet transform

Talukder and Harada (2007) presented image compression based on wavelet transform which is used to check the quality of the compressed image with respect to different thresholding techniques. The basic Haar wavelet transform is used to perform this compression and the entire algorithm is implemented using software based simulation technique. The result shows that the soft thresholding technique is able to generate better quality image in terms of PSNR than hard thresholding technique. Nedunuri et al. (2006) presented a hardware based discrete wavelet transform architecture. To reduce the memory requirements in the architecture, novel diagonal scan method is used to read the input image and also to design efficient filter banks, recursive pyramid hierarchical approach is considered. The three level DWT is implemented in this paper where VHDL is used to code the architecture and synthesized it for Virtex-2 FPGA board. Hasan et al. (2013) presented multilevel decomposition of image through discrete wavelet transform for image compression. The decomposition is performed through hardware architectures which is derived from fast Haar wavelet transform. This technique reduces the hardware utilizations required to decompose the input image. This architecture is coded using VHDL language and implemented on Quartus-II FPGA. Mamatha et al. (2015) presented image fusion using wavelet decomposition method for satellite image. The fusion technique is implemented using co-simulation based techniques where the in built blocks present in the System Generator tool are used which increases the overall hardware utilizations. Vijendra and Kulkarni (2016) used Haar wavelet transform to filter ECG signal. The entire filtering process is done through co-simulation method where the architecture is designed using VHDL language and the input ECG signal was fed through MATLAB present in System Generator tool. This architecture is inefficient in terms of hardware utilizations due to the use of built-in blocks without proper optimizations. Gafsi et al. (2016) presented a hardware based architecture to implement watermarking technique in which the Haar wavelet is used as main component and to reduce the design complexity, modified lifting scheme is proposed. The complete architecture is built using the in built functions that are available on System Generator tool which generates inefficient hardware architecture. Harender and Sharma (2017) presented de-noising of ECG signal based on Haar wavelet transform and universal thresholding techniques. The entire architecture is designed through the built-in functions present in the System Generator tool and synthesized it using XST tool. Khan et al. (2019) presented image compression based on Haar transform, DCT and Run Length Encoding techniques separately for JPEG image. All these techniques were simulated separately on MATLAB tool and then the size of the compressed images were compared. The Haar wavelet transform showed good compression ratio in terms of image size and the PSNR than existing techniques. Bhardwaj and Khunteta (2017) presented a research paper on video watermarking. The watermarking architecture was designed using Haar DWT and DCT algorithm where the conventional algorithms were modified to process videos frame by frame which is simulated using MATLAB software. Chakraborty and Banerjee (2020) presented a tunable VLSI architecture of DWT and to achieve area and memory efficient architecture, Distributed Arithmetic technique was used along with some degree of parallelism. The authors tested this architecture for Daubechies 9/7 and LeGall 5/3 filters. The entire architecture was implemented on Xilinx FPGA board using high level synthesis method. Talukder et al. (2020) presented efficient integer wavelet transform architecture which is used for QRS detection of ECG signal effectively where the wavelet transform is mainly used to de-noise the ECG signal. The haar wavelet, zero-crossing detector, threshold and decision blocks are used to implement the entire architecture which is coded using verilog language and implemented on Digilent Nexys-4 FPGA.

3 Mathematical background

In this section, the mathematical models of existing *Kogge–Stone Adder* and *Haar Wavelet Transform* are discussed.

3.1 Kogge–Stone adder

The *Kogge–Stone Adder* (Kogge and Stone 1973) is the modified version of *Carry Look-Ahead Adder* (Wang et al. 1993). The modification is done to reduce the delay problem in generating *carry* signal for large size adder architecture. This adder is able to produce the output faster than other existing adders with small area overheads (Koyada et al. 2017). The operation of this adder is divided into three parts as *Pre-processing, Carry LookAhead Network* and *Post-Processing* (Kogge and Stone 1973) respectively.

1. *Pre-processing:* In this stage, the *propogate (p)* and *genearte (g)* signals are computed separately for each *'A'* and *'B'* signals respectively. The logical equations of this block can be written as

$$p_i = A_i \oplus B_i \tag{1}$$
$$g_i = A_i \& B_i \tag{2}$$

   where i ← Length of the adder.
2. *Carry Lookahead Network:* The *carry* of the corresponding bits are computed separately in this stage which increases the maximum operating speed of this adder. This stage uses the *propogate (p)* and *generate (g)* signals to determine the corresponding *carry* signal. The logical equation of this stage is given as

$$p_{i:j} = p_{i:k+1} \& p_{k:j} \tag{3}$$
$$g_{i:j} = g_{i:k} | (p_{i:k+1} \& g_{k:j}) \tag{4}$$

   where {j, k} ← Intermediate integer values used to mix signals.
3. *Post-processing:* The computation of the final sum of the corresponding bits are calculated in this stage. The logical equation for this stage is

$$S_i = p_i \oplus c_{i-1} \tag{5}$$

   where c_{i-1} ← Generated carry from previous adder block.

3.2 Haar wavelet transform

For many cases of image analysis, it is necessary to convert the input image into frequency domain to overcome various issues that occurs in time domain analysis (Prasad and Iyengar 1997). Normally various types of Fourier Transforms such as DFT, FFT and STFT etc., are used (Gonzalez and Woods 2008) where complex sinusoidal input data is considered. In most of the real time scenarios, the input data is infinite where the information is spread over the whole time axis of the signal making it difficult to model through regular Fourier Transforms. To overcome from this type of problems, windowing methods are used (Sateesh Kumar et al. 2015). The windowed version of Fourier Transforms is known as windowed Fourier Transforms (Gonzalez and Woods 2008) which is given in Eq. (6) as

$$X(\tau, \omega) = \int_{-\infty}^{\infty} \omega(t - \tau) \cdot x(t) \cdot e^{-j\omega t} dt \tag{6}$$

where $\omega(\cdot) \leftarrow$ Appropriate window Size.

The $X(\tau, \omega)$ is the Fourier Transforms of x(t) where the window $\omega(\cdot)$ is shifted by an amount 'τ' which is modulated version of the window and named as *Short-Time Fourier Transform (STFT)*. Due to the use of single window, the resolution of the analysis is always same for all locations in the time-frequency plane. By varying the window size, the resolution in both time and frequency domain can be changed which is achieved by wavelet transform. In the case of Wavelet Transform, the basis function is obtained from a *Wavelet* and *Dilation/Contraction* operations (Prasad and Iyengar 1997) as

$$h_{a,b}(t) = \frac{1}{\sqrt{a}} \cdot h\left(\frac{t-a}{b}\right) \tag{7}$$

where a $\leftarrow$ Haar basis function.

For large 'a' values, the basis function becomes low frequency function and for small 'a' values, it becomes high frequency function. So, the equation of basic Wavelet Transform (Gonzalez and Woods 2008) is

$$X(a,b) = \frac{1}{\sqrt{a}} \int_{-\infty}^{\infty} x(t) \cdot h^{*}\left(\frac{t-a}{b}\right) dt \tag{8}$$

Different tradeoffs are used in Eqs. (7) and (8) to generate different time-frequency resolutions. To find the equation of Discrete Wavelet Transform, it is needed to discretize the resolution and Dilation/Contraction parameters of Eq. (7) which corresponds to $a = {a_0}^m$ and $b = n{a_0}^m b_0$

$$h_{m,n}(t) = {a_0}^{-\frac{m}{2}} \cdot h({a_0}^m t - nb_0) \tag{9}$$

where $\{m, n\} \in z, a_0 > 1, b_0 \neq 0$.

Now the equation for Discrete Wavelet Transform is

$$X(m,n) = {a_0}^{-\frac{m}{2}} \int_{-\infty}^{\infty} x(t) \cdot h\left({a_0}^{-\frac{m}{2}} t - nb_0\right) dt \tag{10}$$

In the case of wavelets, it is possible to design the wavelet function $h(t)$ such that the set of translated and scaled versions of $h(t)$ forms an *Orthogonal* basics function with the input signal (Prasad and Iyengar 1997). Using this *Orthogonality*, the *Haar basis function* (Gonzalez and Woods 2008; Prasad and Iyengar 1997) can be defined as

$$h(t) = \begin{cases} 1, & 0 < t \leq \frac{1}{2} \\ -1, & \frac{1}{2} \leq t < 1 \\ 0, & \text{Otherwise} \end{cases} \tag{11}$$

The non-separable (Bamerni et al. 2019) version of the two dimensional Haar transform of a M$\times$N matrix in discrete format is given as

$$B = \omega_M \cdot A \cdot \omega_N^T \tag{12}$$

where A $\leftarrow$ Input matrix of M$\times$N size, $\{\omega_M, \omega_N\} \leftarrow$ Haar basis function of corresponding axis in XY coordinate.

But for simplicity in mathematical modelling, 4×4 sized matrix value of signal 'A' is considered (Bhairannawar et al. 2016) as

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \tag{13}$$

Any *Wavelet Transform* uses two different filter banks namely high-pass and low-pass filter which generates low and high frequency coefficients present of the respective input image. For such partitioning, let us consider

$$\omega_4 = \begin{bmatrix} H \\ G \end{bmatrix} \tag{14}$$

where H ← Low-pass filter coefficient matrix for Haar Wavelet, G ← High-pass filter coefficient matrix for Haar Wavelet.

For Haar Wavelet transform, the value of ω_4 (Prasad and Iyengar 1997) becomes

$$\omega_4 = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ -\frac{1}{2} & \frac{1}{2} & 0 & 0 \\ 0 & 0 & -\frac{1}{2} & \frac{1}{2} \end{bmatrix} \tag{15}$$

By using the relation of Eqs. (14) and (15), the value of 'H' and 'G' becomes

$$H = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} \end{bmatrix} \tag{16}$$

$$G = \begin{bmatrix} -\frac{1}{2} & \frac{1}{2} & 0 & 0 \\ 0 & 0 & -\frac{1}{2} & \frac{1}{2} \end{bmatrix} \tag{17}$$

Then Eq. (12) becomes

$$B = \begin{bmatrix} H \\ G \end{bmatrix} A \begin{bmatrix} H^T & G^T \end{bmatrix} \tag{18}$$

$$B = \begin{bmatrix} (HAH^T) & (HAG^T) \\ (GAH^T) & (GAG^T) \end{bmatrix} \tag{19}$$

$$B = \begin{bmatrix} y_{LL} & y_{HL} \\ y_{LH} & y_{HH} \end{bmatrix} \tag{20}$$

where $y_{LL} = HAH^T$ ← Approximation (LL-Band), $y_{LH} = HAG^T$ ← Vertical Difference (LH-Band), $y_{HL} = GAH^T$ ← Horizontal Difference (HL-Band), $y_{HH} = GAG^T$ ← Diagonal Difference (HH-Band).

By substituting the corresponding values into Eq. (20), the equations for all four bands namely *LL, LH, HL and HH* as

$$y_{LL} = \frac{1}{4} \times \begin{bmatrix} (a_{11} + a_{12} + a_{21} + a_{22}) & (a_{13} + a_{14} + a_{23} + a_{24}) \\ (a_{31} + a_{32} + a_{41} + a_{42}) & (a_{33} + a_{34} + a_{43} + a_{44}) \end{bmatrix} \tag{21}$$

$$y_{HL} = \frac{1}{2} \times \begin{bmatrix} (a_{12} + a_{22} - a_{11} - a_{21}) & (a_{14} + a_{24} - a_{13} - a_{23}) \\ (a_{32} + a_{42} - a_{31} - a_{41}) & (a_{34} + a_{44} - a_{33} - a_{43}) \end{bmatrix} \tag{22}$$

$$y_{LH} = \frac{1}{2} \times \begin{bmatrix} (a_{21} + a_{22} - a_{12} - a_{11}) & (a_{23} + a_{24} - a_{13} - a_{14}) \\ (a_{31} + a_{32} - a_{42} - a_{41}) & (a_{43} + a_{44} - a_{33} - a_{34}) \end{bmatrix} \tag{23}$$

$$y_{HH} = \frac{1}{2} \times \begin{bmatrix} (a_{11} + a_{22} - a_{12} - a_{21}) & (a_{13} + a_{24} - a_{23} - a_{14}) \\ (a_{31} + a_{42} - a_{32} - a_{41}) & (a_{33} + a_{44} - a_{43} - a_{34}) \end{bmatrix} \tag{24}$$

By observing Eqs. (21)–(24), we can decrease the size of input matrix from 4×4 to 2×2. So, now consider the input matrix $\begin{bmatrix} a & b \\ c & d \end{bmatrix}$ then the above equations can be modified to Eqs.

(25)–(28) as

$$y_{LL} = \frac{1}{4} \times [(a+b)+(c+d)] \tag{25}$$

$$y_{HL} = \frac{1}{2} \times [(b+d)-(a+c)] \tag{26}$$

$$y_{LH} = \frac{1}{2} \times [(c+d)-(b+a)] \tag{27}$$

$$y_{HH} = \frac{1}{2} \times [(a+d)-(b+c)] \tag{28}$$

4 Proposed architecture

The equations of nonseperable Haar Wavelet Transform are given in Eqs. (25)–(28) which consists of constant division factor of ‘4’ and ‘2’ respectively. The constant division factor can be replaced by shifters in binary arithmetic (Bhairannawar et al. 2016, 2018b) to get optimum hardware architecture. As a result, these equations can be rewritten as

$$y_{LL} = RS_2 \times [(a+b)+(c+d)] \tag{29}$$

$$y_{HL} = RS_1 \times [(b+d)-(a+c)] \tag{30}$$

$$y_{LH} = RS_1 \times [(c+d)-(b+a)] \tag{31}$$

$$y_{HH} = RS_1 \times [(a+d)-(b+c)] \tag{32}$$

where $RS_2 \leftarrow$ Right shift logical by position of ‘2’, $RS_1 \leftarrow$ Right shift logical by position of ‘1’.

The proposed hardware architecture of *Optimized Haar Wavelet Transform* is shown in Fig. 1 which consists of *Pre-processing, Reset Controller, Data Format Conversion, Optimized Controller, Moving Window Architecture, Optimized Kogge–Stone Adder/Subtractor, Buffer, Shifter* and *D_FF* blocks respectively. First the input video is converted into a number of finite frames of standard size (256 × 256) by the *Pre-processing* block through *MATLAB* (Gonzalez et al. 2009) and *Simulink/System Generator* (Karris 2006) tool. The pixel values of those frames are then converted into corresponding user-defined format by the *Data Format Conversion* block to increase data accuracy which is then used to generate 2 × 2 overlapped sub-matrix through *Moving Window Architecture* (Sateesh Kumar et al. 2015; Bhairannawar et al. 2016) block. These sub-matrix pixel values are then processed by *Optimized Kogge–Stone Adder/Subtractor* blocks to generate all four sub-bands (i.e., LL, LH, HL and HH) respectively. Among these bands, *HL, LH and HH Bands* produce some negative coefficients which are removed by *Buffer* (Bhairannawar et al. 2018b; Sateesh Kumar et al. 2015) block. Now the intermediate signals are shifted using seperate *Shifter* (Bhairannawar et al. 2018b; Sateesh Kumar et al. 2015; Palnitkar 2003) blocks to perform the corresponding division factors. But in the case of *non-separable Haar Wavelet Transform*, it must be in non-overlapped format. As a result, *Optimized Controller* and *D_FF* blocks are used in interdependent manner for discarding the intermediate values generated by these overlapped matrix pixels which are also used to implement *Downsample by 2* of the intermediate values by the *D_FF (D-Flipflop)* block. The extra output signal *clk_out* and *rst_out* are used for proper synchronization purpose. The entire architecture is synchronized to the specific video parameters through *Reset Controller* block which is used to reset the entire architecture to

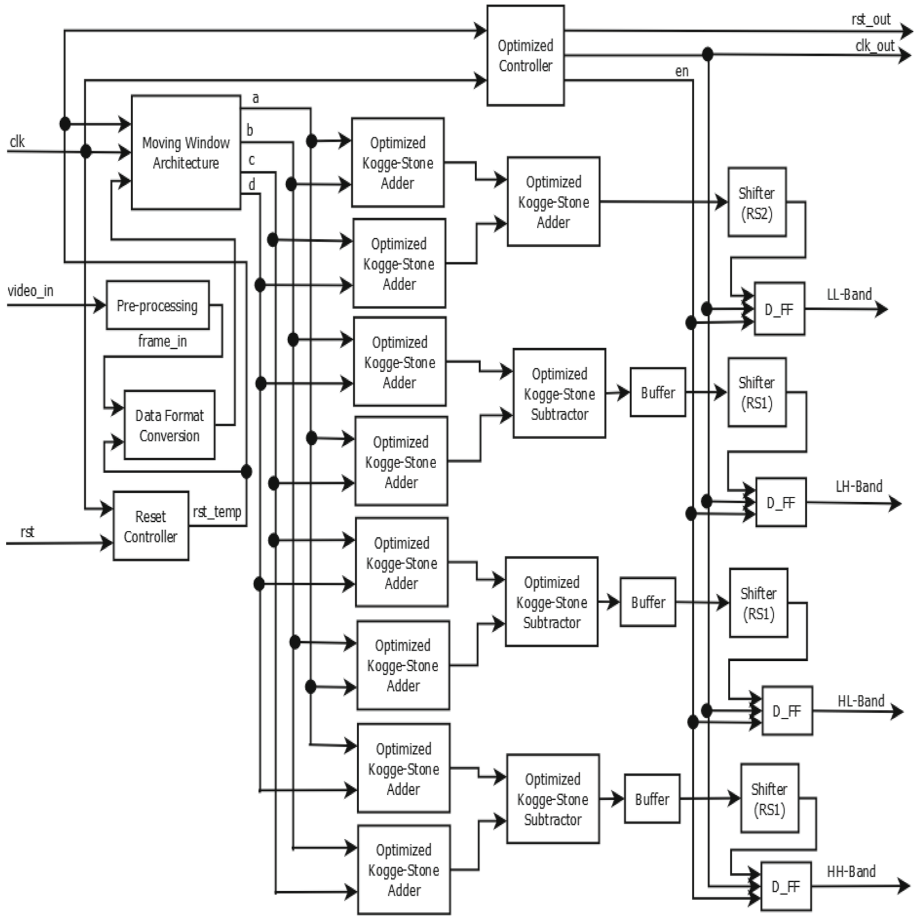


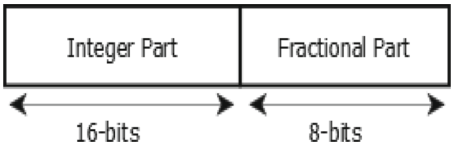
Fig. 1 Proposed architecture of optimized Haar wavelet transform

its initial condition and makes it ready to process next video frame in the similar manner for video applications.

4.1 Data format conversion

From the equations of *Haar Wavelet Transform* given in Eqs. (26)–(29), it is clear that the fractional numbers do appear during the computations. To address this problem, normally IEEE 754 format (IEEE 754 format for floating number 2020) is used which increases the hardware requirement to a great extent. As a result, *Q-format* (Singh and Srinivasan 2003) is considered to implement the fractional part to provide good tradeoff between hardware utilizations and data accuracy. The *Data Format Conversion* block is used to convert the pixel value into this particular user-defined format as shown in Fig. 2 where 16-bits from MSB side are considered as *Integer Part* and the 8-bits at LSB side are considered as the *Fractional Part* of a floating point number. The *Concatenation* operation present in binary arithmetic (Palnitkar 2003) is used to convert the input pixel values into this user-defined format.

Fig. 2 Bit arrangement of proposed user defined data format



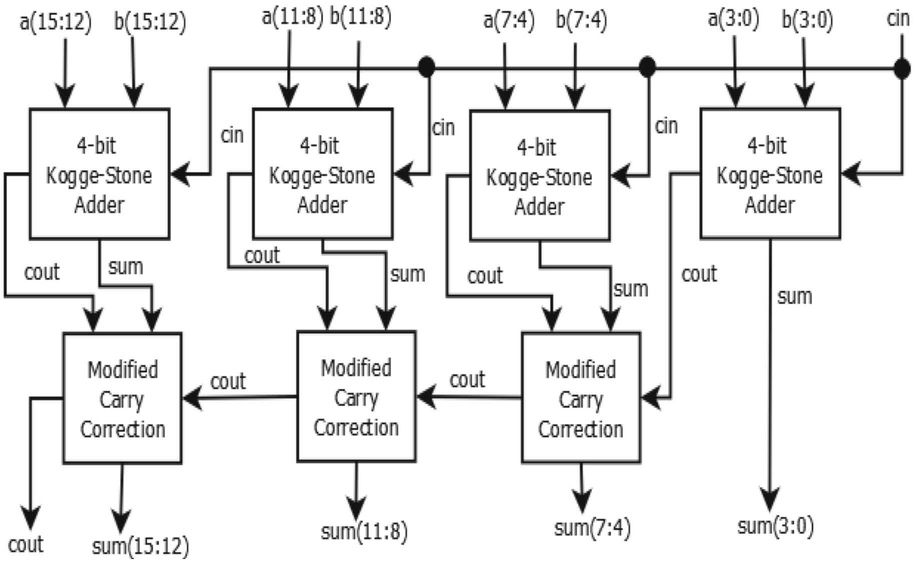


Fig. 3 Proposed optimized Kogge–Stone adder architecture (16-bit)

4.2 Optimized Kogge–Stone adder

The general *Kogge–Stone Adder* (Kogge and Stone 1973) architecture shows increment in delays when larger bits are used for addition which is not acceptable for many high speed real time applications. In such cases, the total number of bits are divided into a finite number of blocks where each block consists of 4-bits of data which are then added using seperate 4-bit Kogge–Stone adder block and the carry continuity problem is eliminated by adding some extra logic at output stage (Tapasvi et al. 2015). Such an implementation is shown in Fig. 3 where *Modified Carry Correction* block is used with *existing Kogge–Stone Adder* architecture (Kogge and Stone 1973) to optimize it's performance.

To reduce the *carry propagation delay*, the *Modified Carry Correction* block is implemented using basic logic gates through parallel architecture. The equations used to design the *Modified Carry Correction* block are given in Eqs. (33)–(37) (Tapasvi et al. 2015).

$$S\_C_{4n+4} = S_{4n+4} \oplus C_{4n+3}; \tag{33}$$

$$S\_C_{4n+5} = S_{4n+5} \oplus \{S_{4n+4} \;\&\; C_{4n+3}\}; \tag{34}$$

$$S\_C_{4n+6} = S_{4n+6} \oplus \{S_{4n+5} \;\&\; S_{4n+4} \;\&\; C_{4n+3}\}; \tag{35}$$

$$S\_C_{4n+7} = S_{4n+7} \oplus \{S_{4n+6} \;\&\; S_{4n+5} \;\&\; S_{4n+4} \;\&\; C_{4n+3}\}; \tag{36}$$

$$C\_C_{4n+7} = C_{4n+7} \oplus \{S_{4n+7} \;\&\; S_{4n+6} \;\&\; S_{4n+5} \;\&\; S_{4n+4} \;\&\; C_{4n+3}\}; \tag{37}$$

where $\{S\_C_{4n+4}, S\_C_{4n+5}, S\_C_{4n+6}, S\_C_{4n+7}\} \leftarrow$ Corrected sum of corresponding bits, $\{S_{4n+4}, S_{4n+5}, S_{4n+6}, S_{4n+7}\} \leftarrow$ Intermediate sums of the corresponding stage, $\{C_{4n+3},$

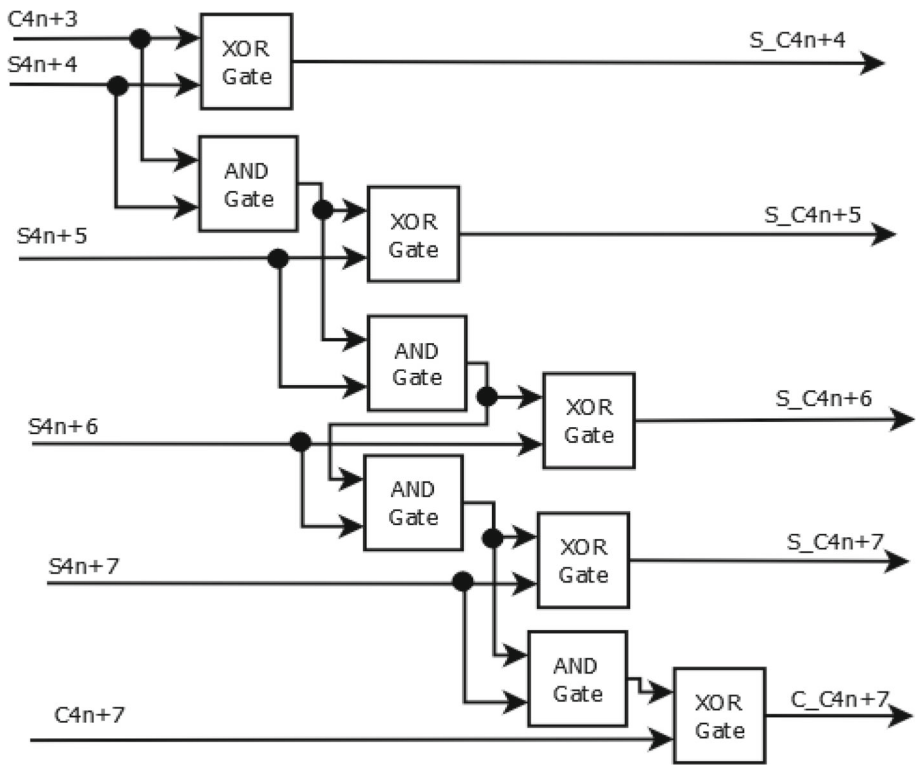


Fig. 4 Proposed hardware architecture of modified carry correction

$C_{4n+7}\}$ ← Intermediate carry of the corresponding stage, C_C_{4n+7} ← Corrected carry of the corresponding stage, n ← Number of stages ($i.e.$, $0, 1, 2, \ldots, (\frac{m}{4} - 1)$), m ← Number of bits are used for additions.

The diagram of *Modified Carry Correction* block is given in Fig. 4 where parallel architecture is considered to reduce the carry propagation delay as much as possible without increasing hardware utilizations to much extent.

4.3 Optimized Kogge–Stone subtractor

In binary arithmetic, negative numbers are modelled by *2's complement format* (Roth 1992). As a result, any subtraction can be modelled by additions only in the case of the binary arithmetic and can be written as

$$y = a - b; \tag{38}$$

$$y = a + (-b); \tag{39}$$

$$y = a + \{(\sim b) + 1\}; \tag{40}$$

where $\{a,\ b\}$ ← Input Signals, y ← Output Signal.

The general architecture of binary subtractor (Roth 1992) is used to implement the *Optimized Kogge–Stone Subtractor* architecture where the normal adder block is replaced by the

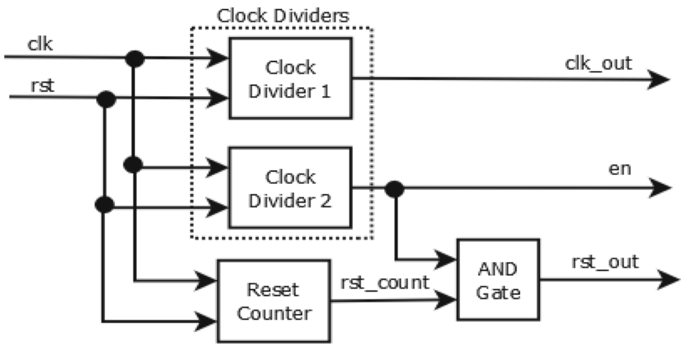


Fig. 5 Proposed optimized controller architecture

Optimized Kogge–Stone Adder block which increases the operating speed and reduces the hardware requirements moderately.

4.4 Optimized controller

The block diagram of the *Optimized Controller* is shown in Fig. 5 which consists of novel *Clock Dividers* and novel *Reset Counter* blocks respectively. This block is used to control the operation of the entire architecture by controlling it's intermediate datapath with the help of *D_FF* block in proper synchronized manner. The use of simple architectural model designed by basic logical elements simplifies the entire architecture with respect to existing (Bhairannawar et al. 2016) where complex *Finite State Machine (FSM)* model is used.

4.4.1 Clock dividers

The *Clock Dividers* block is used to generate different control signals through seperate clock division like logics which mainly consists of two blocks namely *Clock Divider 1* and *Clock Divider 2* respectively.

1. *Clock Divider 1:* The *Clock Divider 1* is basically a simple clock divider circuit with a division factor of ‘2’ which is used to perform *Downsample by 2* operation present in the *Wavelet Algorithm* (Mallat 2008) with the help of *D_FF* blocks. The *D-Flipflop* and *NOT Gate* arrangement (Roth 1992) is used to implement the *Clock Divider 1* block on FPGA.
2. *Clock Divider 2:* The *Moving Window Architecture* (Sateesh Kumar et al. 2015; Bhairannawar et al. 2016) block is used to generate 2 × 2 sub-matrices from the serial data. But to calculate proper sub-band coefficients, the input sub-matrices must be in non-overlapped fashion (Mallat 2008). To generate non-overlapped image sub-matrix, novel *Clock Divider 2* and *D_FF* blocks are used synchronously in which the *Clock Divider 2* block is used to generate control signal (*‘en’*) which is further used by *D_FF* block to eliminate the overlapped data present at the output side. The algorithm is used to built this block is shown in *Algorithm* 1.

Algorithm 1 : Clock Divider 2

```
    Inputs: rst,clk;
    Variable: count_temp ;
    Constant: row_size ;
    Output: en;
  if $(rst = 0)$ then
      $en = 0$;
      $countl\_temp = 0$;
  else if $\{(clk)_{rising\_edge}\}$ then
      if $\{count\_temp < (row\_size - 1)\}$ then
          $count\_temp = count\_temp + 1$;
          $en = 1$;
      else if $[\{(row\_size - 1)\} \geq count\_temp < \{(2 * row\_size) - 1\}]$ then
          $count\_temp = count\_temp + 1$;
          $en = 1$;
      else
          $count\_temp = 0$;
          $en = 0$;
      end if
  end if
```

4.4.2 Reset counter

To perform all the calculations and to generate proper output bands, the entire architecture needs some time and also it is essential to synchronize the output frequency with display devices. This is because the output frequency of any wavelet transform is equal to some factor of the input frequency. For this frequency synchronization, the novel *Reset Counter* and *AND Gate* blocks are used. The algorithm is used to design the *Reset Counter* is shown in *Algorithm* 2.

Algorithm 2 : Reset Counter

```
    Inputs: rst,clk;
    Variable: count_temp ;
    Constant: row_size ;
    Output: rst_count;
  if $(rst = 0)$ then
      $count\_temp = 0$;
      $rst\_count = 0$;
  else if $\{(clk)_{rising\_edge}\}$ then
      if $(count\_temp \leq row\_size)$ then
          $count\_temp = count\_temp + 1$;
          $rst\_count = 0$;
      else
          $count\_temp = count\_temp$;
          $rst\_count = 1$;
      end if
  end if
```

Table 1 Hardware utilizations of different size of optimized Kogge–Stone Adder

Parameters	Modified Carry Correction	Optimized Kogge–Stone Adder				
		8-Bits	12-Bits	16-Bits	24-Bits	32-Bits
FPGA	Spartan-6 (xc6slx45-3csg324)					
Slice LUTs	2	6	12	16	22	29
Occupied slices	4	13	22	30	35	38
LUT-FF Pairs	2	6	12	16	22	29
Delay[a] (ns)	5.602	7.695	9.997	11.215	15.357	18.124

[a]For maximum combinational path

Table 2 Hardware utilizations of different size of Optimized Kogge–Stone Subtractor

Parameters	Optimized Kogge–Stone Subtractor					
	4-Bits	8-Bits	12-Bits	16-Bits	24-Bits	32-Bits
FPGA	Spartan-6 (xc6slx45-3csg324)					
Slice LUTs	2	6	13	18	24	31
Occupied Slices	6	18	26	42	60	46
LUT-FF Pairs	2	6	13	18	24	31
Delay[a] (ns)	6.469	9.032	11.284	13.619	17.093	21.769

[a]For maximum combinational path

5 FPGA implementation

In this section, the implemented results of the proposed architecture with it's sub-blocks are discussed. For hardware implementation, *Digilent Spartan-6 (xc6slx45-3csg324)* FPGA (Datasheet of Digilent ATLYS FPGA board 2020) is used and standard *VHDL laguage* (Roth 2006) is used to simulate the architecture with the help of *Xilinx 14.7 Design Suite* (Xilinx ISE In-Depth Tutorial 2020) tool through *balanced synthesis* option. The image and video frames are then fed to the proposed architecture using *Simulink/System Generator* (Karris 2006) tool with the help of *MATLAB* (Gonzalez et al. 2009) in real time using standard interfacing procedure.

5.1 Optimized Kogge–Stone adder

The hardware utilizations of *Optimized Kogge–Stone Adder* with different input bit sizes and *Modified Carry Correction* blocks are given in Table 1. From Table 1, it can be seen that the delay and hardware requirements have not increased rapidly with the size of input bits which further helps to achieve faster adder for higher bit sizes.

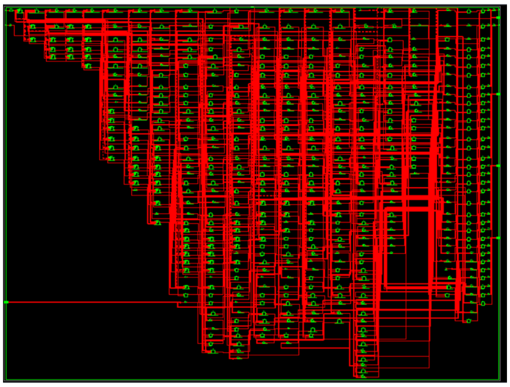
5.2 Optimized Kogge–Stone subtractor

The hardware utilization of Optimized Kogge–Stone Subtractor with different input bit sizes are given in Table 2 and from the table it can be seen that the delay and hardware utilizations have not increased rapidly with the size of the bits which further helps to achieve faster subtractor for higher bit sizes.

Table 3 Hardware utilizations of Optimized Haar DWT with its individual components

Parameters	Moving Window Architecture	Optimized Controller	Optimized Haar Wavelet Transform
FPGA	Spartan-6 (xc6slx45-3csg324)		
Slice Registers	40	21	96
Slice LUTs	76	46	217
Memory	64	0	64
Occupied slice	23	17	85
LUT-FF pairs	92	46	235
Minimum period (ns)	2.127	3.868	3.868
Maximum frequency (MHz)	470.256	266.654	258.528
Latency[a] (μs)	–	–	1.992
Processing time[a] (ms)	–	–	0.256

[a]For each frame


(a) Technology Schematic

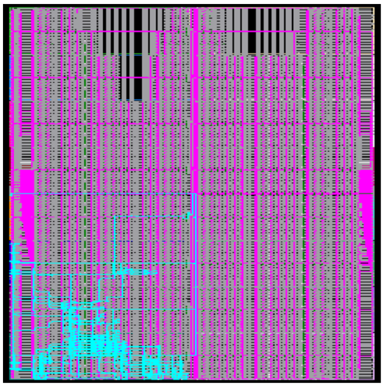

(b) FPGA Schematic

Fig. 6 Generated schematic diagrams of the proposed optimized Haar DWT after synthesis

5.3 Optimized Haar wavelet transform

The hardware utilization of two-dimensional non-separable Optimized Haar Wavelet transform is given in Table 3 with it's corresponding internal components in which *24-bit Optimized Kogge–Stone Adder and Subtractor* are used to get optimum result with optimum hardware utilizations.

The generated *Technology Schematic* and *FPGA Schematic* of Optimized Haar Wavelet is shown in Fig. 6a, b respectively where the *Technology Schematic* represent that how the designed architecture is mapped inside FPGA through its building blocks (*LUTs, Slice Registers* and *LUT-FF Pairs* etc.) and the *FPGA Schematic* shows that how effectively the design is *placed and routed* inside FPGA with all design constraints. Both diagrams are indirectly related to the area and resource utilizations at hardware level when implemented at chip level using various backend VLSI techniques (Weste and Eshraghian 1994).

The four sub-bands (LL, LH, HL and HH respectively) generated by the proposed architecture for both standard videos (Standard Video Database 2020) and standard images (Gonzalez

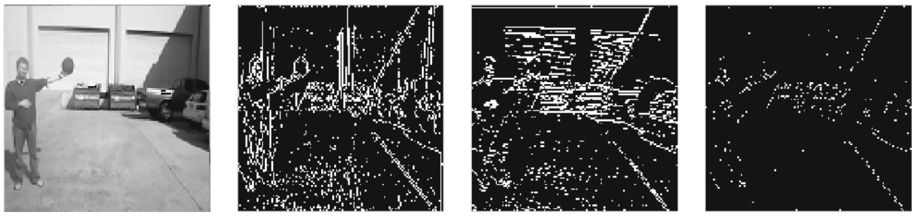

(a) LL Band **(b)** LH Band **(c)** HL Band **(d)** HH Band

Fig. 7 Generated sub-bands by optimized Haar DWT architrcture for 36th frame of a standard video

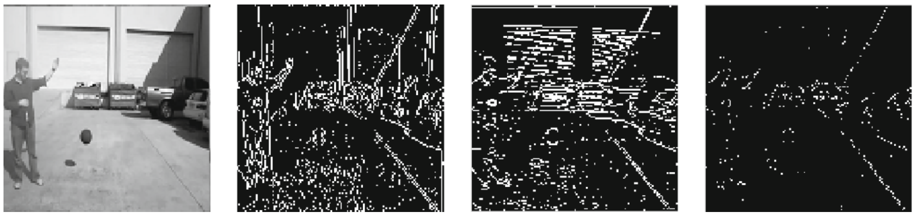

(a) LL Band **(b)** LH Band **(c)** HL Band **(d)** HH Band

Fig. 8 Generated sub-bands by optimized Haar DWT architrcture for 75th frame of a standard video

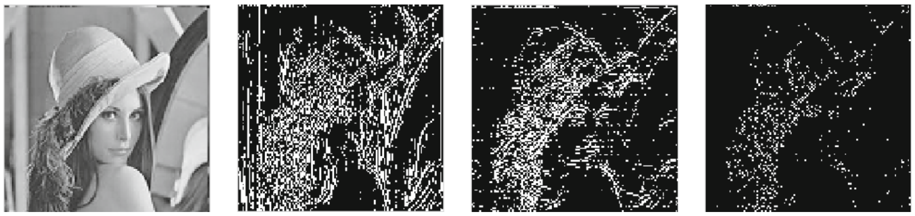

(a) LL Band **(b)** LH Band **(c)** HL Band **(d)** HH Band

Fig. 9 Generated sub-bands by optimized Haar DWT architrcture for Lena image

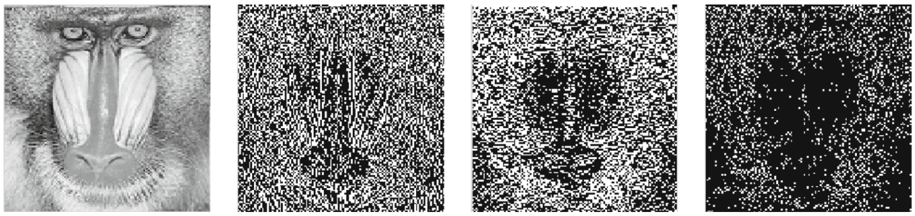

(a) LL Band **(b)** LH Band **(c)** HL Band **(d)** HH Band

Fig. 10 Generated sub-bands by optimized Haar DWT architrcture for Mandrill Image

and Woods 2008; Jayaraman et al. 2009) are given in Figs. 7, 8, 9 and 10 for different video frames and images respectively. In all the cases, it can be seen that a small portion of edges are distorted and is mainly due to the use of overlapped image sub-matrix for internal processing. The processed video frames of *Sample 1* video is shown in Figs. 7 and 8 where the sub-bands for 36th frame are shown in Fig. 7 and 75th frame in Fig. 8 respectively.

In the similar manner, the sub-bands of standard images (Lena and Mandrill) are shown in Figs. 9 and 10 respectively.

6 Performance analysis

The performance of the proposed *Optimized Haar Wavelet* transform architecture is compared with various existing architectures in various different aspects to check the overall performance of the proposed architecture.

6.1 Performance parameters

The performance parameters are used in this case is mainly divided into two catagories as *Data Accuracy* and *Hardware Parameters*.

6.1.1 Data accuracy

The data accuracy of any image and video processing system is mainly depends upon the accuracy of all the pixels present in that image or video frame. Normally frame or image comparisons are used to check the data accuracy of the the system which indicates the quality of the processed image with respect to input image or frame in terms of some mathematical parameter having huge impact on calculating the accuracy of the image and video processing systems (Bhairannawar et al. 2016, 2018b). In this paper, *Euclidean Distance* is used to compare the data accuracy of the frame or image. The equation is used to calculate *Euclidean Distance* between two same sized image/frame (Basics of Euclidean Distance for Image Processing 2020) is shown in Eq. (41) as

$$Eucledian\ Distance = \sqrt{\sum_{i=0}^{(M-1)} \sum_{j=0}^{(N-1)} (I_{i,j} - O_{i,j})^2} \tag{41}$$

where I ← Input image/frame pixel values, O ← Output image/frame pixel values, $\{M, N\}$ ← Image/frame size in both direction (XY) respectively, $\{i, j\}$ ← Integer values.

6.1.2 Hardware parameters

Hardware parameters of any architecture is directly depends upon the design efficiency of that architecture such as the total area required by the architecture on silicon wafer, maximum operating frequency and power consumption etc. As a result, the main hardware parameters like *Slice LUTs, Occupied Slice, LUT-FF Pairs, Path Delay, Slice Registers, Memory and Maximum Frequency* (Wolf 2004) etc., are considered for comparison purpose.

6.2 Performance comparison

In this section, the performance of the proposed architecture is compared with existing to valuate the effectiveness of the proposed architecture.

6.2.1 Data accuracy

The *Haar Wavelet Transform* of a $(M \times N)$ frame or image produces four sub-bands of size $(\frac{M}{2} \times \frac{N}{2})$. Among those sub-bands, only LL-Band consists of major frame data which can be visualized as the compressed version of the input frame. To compare the data accuracy of the

Table 4 Euclidean distance comparisons of the LL band of proposed Haar DWT with actual LL band for different standard video frames

Video	Frame Number	LL Band of Software Implementation Gonzalez et al. (2009)	LL Band of Hardware Implementation	| Difference |
Sample 1	23	1.6421e+04	1.6405e+04	0.0006e+04
	36	1.6426e+04	1.6415e+04	0.0011e+04
	41	1.6423e+04	1.6413e+04	0.0010e+04
	75	1.6421e+04	1.6400e+04	0.0021e+04
	102	1.6421e+04	1.6410e+04	0.0011e+04
	194	1.6407e+04	1.6383e+04	0.0024e+04
Sample 2	5	1.6466e+04	1.6461e+04	0.0006e+04
	17	1.6444e+04	1.6440e+04	0.0005e+04
	39	1.6384e+04	1.6378e+04	0.0006e+04
	64	1.6452e+04	1.6448e+04	0.0003e+04
	95	1.6368e+04	1.6360e+04	0.0008e+04
	107	1.6467e+04	1.6457e+04	0.0010e+04

frame or image generated by the proposed architecture with respect to existing, the input frame or image is first resized into the same size of the generated LL band ($\frac{M}{2} \times \frac{N}{2}$) and then the corresponding *Euclidean Distances* are calculated for the output of the proposed architecture and the output generated by existing software (MATLAB) implementation (Gonzalez et al. 2009) seperately with the help of the resized frame or image. The *Euclidean Distance* values for different standard frames and images are tabulated in Tables 4 and 5 respectively where $\mid Difference \mid$ (Bhairannawar et al. 2018a, b) is calculated using Eq. (42) as

$$\mid Difference \mid = \mid A - B \mid \tag{42}$$

where A $\leftarrow$ LL-Band generated by the hardware architecture, B $\leftarrow$ LL-Band generated by the existing software algorithm.

From Tables 4 and 5, it can be seen that the *Euclidean Distance* values of the hardware generated LL-Band is very near to the existing software generated LL-Band (Gonzalez et al. 2009), resulting in very less $\mid Difference \mid$ value and almost nigligible.This proves that the proposed architecture is able to produce results which are very much nearer to the actual results. The small differences are due to the *truncation error* (Roth 2006; Weste and Eshraghian 1994) which normally occures in any fixed point binary arithmetic.

6.2.2 Hardware comparisons

The hardware parameters which are explained in *Sect.* 5.3, are used to compare the proposed *Optimized Kogge–Stone Adder* and *Optimized Haar Wavelet Transform* with corresponding existing techniques in terms of hardware utilizations.

1. *Optimized Kogge–Stone Adder:* The comparison of proposed 8-bit Kogge–Stone adder with existing is given in Table 6. Similarly, the comparison of the proposed 16-bit and 32-bit *Kogge–Stone Adder* are given in Tables 7 and 8 respectively. From the tables, it can be observed that the proposed architecture requires less hardware resources and less execution time in the form of delay than existing which makes this architecture more

Table 5 Eucledian distance comparisons of the LL band of proposed Haar DWT with actual LL band for different standard images

Image	LL Band of Software Implementation Gonzalez et al. (2009)	LL Band of Hardware Implementation	| Difference |
Lena	1.6437e+04	1.6431e+04	0.0006e+04
Cameraman	1.6465e+04	1.6460e+04	0.0005e+04
Boat	1.6468e+04	1.6452e+04	0.0016e+04
Boy	1.6512e+04	1.6510e+04	0.0002e+04
Barbara	1.6431e+04	1.6424e+04	0.0007e+04
House	1.6597e+04	1.6591e+04	0.0006e+04
Mandrill	1.6449e+04	1.6437e+04	0.0012e+04
Pepper	1.6436e+04	1.6420e+04	0.0016e+04
Plane	1.6496e+04	1.6491e+04	0.0005e+04
Rice	1.6474e+04	1.6467e+04	0.0007e+04
Bird	1.6455e+04	1.6451e+04	0.0004e+04

Table 6 Hardware comparisons of the existing and proposed 8-bit Kogge–Stone Adder

Parameters	Shilpa et al. (2018)	San and Yakunin (2018)	Proposed Architecture
FPGA	Spartan-6	Cyclone-II	Spartan-6
Slice LUTs	10	–	6
Delay (ns)	8.397	12.2	7.695

Table 7 Hardware comparisons of the existing and proposed 16-bit Kogge–Stone Adder

Parameters	San and Yakunin (2018)	Siddmal (2017)	Shrivastava et al. (2017)	Penchalaiah and Siva Kumar (2018)	Proposed Architecture
FPGA	Cyclone-II	Spartan-3	Spartan-6	Spartan-6	Spartan-6
Slice LUTs	–	18	43	–	16
Delay (ns)	16.1	13.002	11.935	14.089	11.215

efficient than existing. The main reason behind this optimizations is the use of *Modified Carry Correction* architecture which helps to reduce the *worst case path delay* (Smith et al. 2004; Weste and Eshraghian 1994) without increasing area utilizations.

2. *Optimized Haar Wavelet Transform:* The comparison of the hardware utilizations of the proposed Haar DWT with existing is given in Table 9. From the Table 9, it is clear that the proposed Haar Wavelet Transform architecture requires less hardware resources and can operate at higher speed than existing which makes it more superior than existing. The main reason behind this superiority is the optimization of each sub-blocks present in the architecture through various optimization techniques such as the constant division factors are replaced by corresponding shifters, the existing Kogge–Stone Adder/Subtractor architecture is optimized by Modified Carry Correction block and the Controller architecture is optimized through the uses of basic gates.

Table 8 Hardware comparisons of the existing and proposed 32-bit Kogge–Stone Adder

Parameters	San and Yakunin (2018)	Siddmal (2017)	Kaur and Toor (2017)	Anitha and Sathish (2018)	Proposed Architecture
FPGA	Cyclone-II	Spartan-3	Spartan-3E	–	Spartan-6
Slice LUTs	–	46	46	332	36
4-i/p LUTs	–	–	81	–	69
Delay (ns)	18.7	19.293	21.913	24.56	18.124

Table 9 Hardware comparisons of the existing and proposed Haar DWT

Parameters	Hasan et al. (2013)	Hajjaji et al. (2019)	Mhamunkar and Gayal (2015)	Bamerni et al. (2018)	Proposed Architecture
FPGA	DE2	Virtex-5	Spartan-3	Spartan-3A	Spartan-6
Slice Registers	–	1536	1880	99	96
4-i/p LUTs	–	2092	2971	166	128
LUT-FF pairs	–	–	2118	–	235
Total logics	700	–	–	–	151
Latency (ns)	–	20	–	–	1.992
Maximum frequency (MHz)	50	224	–	–	258.528

Limitation: The entire architecture is designed to operate on specific video parameters such as frame size and frame rate etc., which are briefly discussed in the paper. To process videos with different frame size and frame rate, the proposed architecture needs to be modified according to required parameter values.

7 Conclusion

In this paper, efficient FPGA architecture to implement Haar wavelet transform for image and video processing is proposed which is optimized by using *Moving Window Architecture, Modified Kogge–Stone Adder/Subtractor, Optimized Controller, Buffer and Shifter* respectively. The *Modified Carry Correction* block is used to build the Optimized Kogge–Stone adder/Subtractor block through its parallel architecture. The *Optimized Controller* is modelled using *Counter* and *Clock Dividers* to reduce the architectural complexity and hardware requirements. Similarly, for same reason the dividers are replaced by the shifters. The *Reset Controller* block makes this architecture suitable for video processing, by resetting it after each frame. To generate nearly accurate result, enough bit sizes are considered at intermediate level with *Q-notations*. In future, high resolution camera is interfaced with FPGA from which real time high speed video is captured and processed directly using this architecture with some modifications.

Appendices

The user-defined symbols are used in all the algorithms and equations present in this paper are as follows

$\sim \leftarrow$ Logical NOT Operation.
$| \leftarrow$ Logical OR Operation.
$\bigoplus \leftarrow$ Logical XOR Operation.
$\& \leftarrow$ Logical AND Operation.
$(,)/\{,\}/[,] \leftarrow$ Logical Concatenation Operation.
$(x\ downto\ y) \leftarrow$ Data of Length (x-y+1).
$(N) \leftarrow$ Bit Present in Nth Position of a Binary Digit.

References

Aishwarya, N., Abirami, S. & Amutha, R. (2016). Multi-focus image fusion using discrete wavelet transform and sparse representation. In *IEEE international conference on wireless communications, signal processing and networking (WiSPNET)* (pp. 2377–2382). https://doi.org/10.1109/WiSPNET.2016.7566567.

Alsubari, A., Satange, D. N., Ramteke, R. J. (2017). Facial expression recognition using wavelet transform and local binary pattern. In *2nd IEEE international conference for convergence in technology (I2CT)* (pp. 338–342). https://doi.org/10.1109/I2CT.2017.8226147.

Anitha, M., & Sathish, K. (2018). Implementation of high speed 32 bit Kogge–Stone adder. *Journal of Signal Processing and Wireless Networks*, *3*(1), 37–40.

Aravind B. N. & Suresh, K. V. (2015). An improved image denoising using wavelet transform. In *IEEE international conference on trends in automation, communications and computing technology (I-TACT-15)* (pp. 1–5). https://doi.org/10.1109/ITACT.2015.7492679.

Arora, A., & Niranjan, V. (2017). A new 16-bit high speed and variable stage carry skip adder. In *3rd IEEE international conference on computational intelligence and communication technology (CICT)* (pp. 1–4). https://doi.org/10.1109/CIACT.2017.7977359.

Bamerni, S. A., & Al-Sulaifanie, A. K. (2018). An efficient non-separable architecture for Haar wavelet transform with lifting structure. *International Journal of Computers*, *12*, 43–53.

Bamerni, S. A., & Al-Sulaifanie, A. K. (2019). An efficient non-separable architecture for Haar wavelet transform with lifting structure. *Journals of Microprocessors and Microsystems*, *71*, 1–7. https://doi.org/10.1016/j.micpro.2019.102881.

Basics of Euclidean distance for image processing. [Online] Available https://en.wikipedia.org/wiki/Euclidean_distance. Accessed 26 March 2020.

Bhairannawar, S. S., Sarkar, S., & Raja, K. B. (2018a). FPGA implementation of optimized Karhunen–Loeve transform for image processing applications. *Journal of Real time Image Processing*. https://doi.org/10.1007/s11554-018-0776-x.

Bhairannawar, S. S., Sayantam, S., Raja, K. B., & Venugopal, K. R. (2018b). Implementation of fingerprint based biometric system using optimized 5/3 DWT architecture and modified CORDIC based FFT. *Journal of Circuits, Systems and Signal Processing,37*(1), 342–366. https://doi.org/10.1007/s00034-017-0555-0.

Bhairannawar, S. S., Raja, K. B., & Venugopal, K. R. (2016). An efficient reconfigurable architecture for fingerprint recognition. *Journal of VLSI Design*,. https://doi.org/10.1155/2016/9532762.

Bhardwaj, A., & Khunteta, A. (2017). Video watermarking equations using DWT and DCT sub-sub bands for secure transmission over communication channels: A research paper. In *IEEE international conference on power, control, signals and instrumentation engineering (ICPCSI)* (pp. 435–440). https://doi.org/10.1109/ICPCSI.2017.8392333.

Chakraborty, A., & Banerjee, A. (2020). Area and memory efficient tunable VLSI implementation of DWT filters for image decomposition using distributed arithmetic. *International Journal of Electronics*,. https://doi.org/10.1007/s00034-019-01328-2.

Datasheet of Digilent ATLYS FPGA board. [Online] Available at https://reference.digilentinc.com/_media/atlys:atlys:atlys_rm.pdf. Accessed 26 March 2020.

Elakkiya, S., & Audithan, S. (2014). Feature based object recognition using discrete wavelet transform. In *Second IEEE international conference on current trends in engineering and technology (ICCTET)* (pp. 1–4). https://doi.org/10.1109/ICCTET.2014.6966323.

Gafsi, M., Ajili, S., Hajjaji, M. A., & Mtibaa, A. (2016). XSG for hardware implementation of a robust watermarking system. In *17th IEEE international conference on sciences and techniques of automatic control and computer engineering (STA)* (pp. 117–122). https://doi.org/10.1109/STA.2016.7952031.

Gonzalez, R. C., & Woods, R. E. (2008). *Digital image processing* (3rd ed.). London: Pearson Education.

Gonzalez, R. C., Woods, R. E., & Eddins, S. L. (2009). *Digital image processing using MATLAB* (2nd ed.). Knoxville, TN: Gatesmark Publishing.

Gupta, V., Mahle, R. & Shriwas, R. S. (2013). Image denoising using wavelet transform method. In *Tenth IEEE international conference on wireless and optical communications networks (WOCN)* (pp. 1–4). https://doi.org/10.1109/WOCN.2013.6616235.

Hajjaji, M. A., Gafsi, M., Abdelali, A. B., & Mtibaa, A. (2019). FPGA implementation of digital images watermarking system based on discrete Haar wavelet transform. *Journal of Security and Communication Networks*,. https://doi.org/10.1155/2019/1294267.

Harender, & Sharma, R. K. (2017) EEG signal denoising based on wavelet transform. In *IEEE international conference of electronics, communication and aerospace technology (ICECA)* (pp. 758–761). https://doi.org/10.1109/ICECA.2017.8203645.

Hasan, K. K., Ngah, U. K., & Salleh, M. F. M. (2013). Multilevel decomposition discrete wavelet transform for hardware image compression architectures applications. In *IEEE International conference on control system, computing and engineering* (pp. 315–320). https://doi.org/10.1109/ICCSCE.2013.6719981.

IEEE 754 format for floating number. [Online] Available https://en.wikipedia.org/wiki/Single-precision_floating-point_format. Accessed 26 March 2020.

Jayaraman, S., Esakkirajan, S., & Veerakumar, T. (2009). *Digital image processing*. London: Tata McGraw Hill.

Karris, S. T. (2006). *Introduction to simulink with engineering applications*. Newton: Orchard Publications.

Kaur, A., & Toor, C. K. (2017). Implementation of parallel prefix adders using FPGA's. *Australian Journal of Basic and Applied Sciences*, *11*(8), 100–106.

Khan, S., Nazir, S., Hussain, A., Ali, A., & Ullah, A. (2019). An efficient JPEG image compression based on Haar wavelet transform, discrete cosine transform, and run length encoding techniques for advanced manufacturing processes. *Journal of Measurement and Control*, *52*, 1532–1544. https://doi.org/10.1177/0020294019877508.

Kogge, P. M., & Stone, H. S. (1973). A parallel algorithm for the efficient solution of a general class of recurrence relations. *IEEE Transactions on Computers*, *22*(8), 786–793. https://doi.org/10.1109/TC.1973.5009159.

Koyada, B., Meghana, N., Jaleel M. O., & Jeripotula, P. R. (2017). A comparative study on adders. In *IEEE international conference on wireless communications, signal processing and networking (WiSPNET)* (pp. 2226–2230). https://doi.org/10.1109/WiSPNET.2017.8300155.

Lee, B. D., & Oklobdzija, V. G. (1990). Optimization and speed improvement analysis of carry-lookahead adder structure. In *Twenty-fourth asilomar conference on signals, systems and computers* (pp. 359–362). https://doi.org/10.1109/ACSSC.1990.523471.

Mallat, S. (2008). *A wavelet tour of signal processing: The sparse way* (1st ed.). London: Academic Press.

Mamatha, G., Sumalatha, V., & Lakshmaiah, M. V. (2015). FPGA implementation of satellite image fusion using wavelet substitution method. In *IEEE international science and information conference (SAI)* (pp. 1155–1159). https://doi.org/10.1109/SAI.2015.7237290.

Marques, O. (2012). *Practical Image and Video Processing Using MATLAB* (1st ed.). London: Wiley-IEEE Press.

Mhamunkar, N. S., & Gayal, B. S. (2015). Design and implementation of generic 2-D biorthogonal discrete wavelet transform on FPGA. In *IEEE international conference on energy systems and applications* (pp. 622–627). https://doi.org/10.1109/ICESA.2015.7503424.

Mohammadi, S., Omidi, R., & Lotfinejad, M. (2010). Low-power area-efficient fault tolerant adder in current mode multi valued logic using Berger codes. *Journal of Electronic Testing*, *36*, 555–563. https://doi.org/10.1007/s10836-020-05887-0.

Nashat, A. A., & Hussain, N. M. H. (2016). Image Compression based upon wavelet transform and a statistical threshold. In *IEEE international conference on optoelectronics and image processing (ICOIP)* (pp. 20–24). https://doi.org/10.1109/OPTIP.2016.7528492.

Nedunuri, S., Cheung, J. Y., & Nedunuri, P. (2006). Design of low memory usage discrete wavelet transform on FPGA using novel diagonal scan. In *IEEE international symposium on parallel computing in electrical engineering (PARELEC)* (pp. 1–6). https://doi.org/10.1109/PARELEC.2006.29.

Palnitkar, S. (2003). *Verilog HDL: A guide to digital design and synthesis* (2nd ed.). London: Pearson Education.

Penchalaiah, U. & Siva Kumar, V. G. (2018). Design of high-speed and energy-efficient parallel prefix Kogge–Stone adder. In *IEEE international conference on system, computation, automation and networking (ICSCAN)* (pp. 1-7). https://doi.org/10.1109/ICSCAN.2018.8541143.

Prasad, L., & Iyengar, S. S. (1997). *Wavelet analysis with applications to image processing* (1st ed.). London: CRC Press.

Rajasekhar, V., Vaishnavi, V., Koushik, J. & Thamarai, M. (2014). An efficient image compression technique using discrete wavelet transform (DWT). In *IEEE international conference on electronics and communication systems (ICECS)* (pp. 1–4). https://doi.org/10.1109/ECS.2014.6892826.

Roth, C. H. (1992). *Fundamentals of logic design* (1st ed.). Mumbai: Jaico Publishing House.

Roth, C. H. (2006). *Digital system design using VHDL*. Boston, MA: Cengage Learning.

San, A. M., & Yakunin, A. N. (2018). Reducing the hardware complexity of a parallel prefix adder. In *IEEE conference of Russian young researchers in electrical and electronic engineering (EIConRus)* (pp. 1348–1351). https://doi.org/10.1109/EIConRus.2018.8317346.

Sateesh Kumar, H. C., Sarkar, S., Bhairannawar, S. S., Raja, K. B., & Venugopal, K. R. (2015). FPGA implementation of moving object and face detection using adaptive threshold. *International Journal of VLSI Design and Communication Systems*, *6*(5), 15–35. https://doi.org/10.5121/vlsic.2015.6502.

Shilpa, K. C., Shwetha, M, Geetha, B. C, Lohitha, D. M., Navya & Pramod, N. V. (2018). Performance analysis of parallel prefix adder for datapath VLSI design. In *Second IEEE international conference on inventive communication and computational technologies (ICICCT)* (pp. 1552–1555). https://doi.org/10.1109/ICICCT.2018.8473087.

Shrivastava, A., Churhe, S., Bhagat, H., & Wamankar, R. (2017). Design and estimation of delay, power and area for parallel prefix adders. *International Journal of Engineering Research and Application*, *7*(4), 1–8. https://doi.org/10.9790/9622-0704050108.

Siddmal, R. R. R. S. V. (2017). Hardware implementation of optimized Kogge–Stone adder. *International Journal of Emerging Technology in Computer Science and Electronics,14*(2), 1–8.

Singh, A., & Srinivasan, S. (2003). Digital signal processing implementations: Using DSP microprocessors—with examples from TMS320C54xx. Nelson Engineering.

Smith, M. J. S. (2004). *Application specific integrated circuit* (1st ed.). London: Pearson Education.

Soares, L. B., da Rosa, M. M. A., Diniz, C. M., da Costa, E. A. C., & Bampi, S. (2019). Design methodology to explore hybrid approximate adders for energy-efficient image and video processing accelerators. *IEEE Transactions on Circuits and Systems I: Regular Papers*, *66*(6), 2137–2150. https://doi.org/10.1109/TCSI.2019.2892588.

Standard Video Database. [Online] Available at https://media.xiph.org/video/derf/. Accessed 26 March 2020.

Talukder, K. H., & Harada, K. (2007). Haar wavelet based approach for image compression and quality assessment of compressed image. *IAENG International Journal of Applied Mathematics*, *36*(1), 1–8.

Talukder, S., Singh, R., Bora, S., & Paily, R. (2020). An efficient architecture for QRS detection in FPGA using integer Haar wavelet transform. *Journal of Circuits, Systems, and Signal Processing*, *39*, 3610–3625. https://doi.org/10.1007/s00034-019-01328-2.

Tapasvi, B., Bala Sinduri, K., Lakshmi, B. G. S. S. B., & Udaya Kumar, N. (2015). Implementation of 64-bit Kogge–Stone carry select adder with ZFC for efficient area. In *IEEE international conference on electrical, computer and communication technologies (ICECCT)* (pp. 1–6). https://doi.org/10.1109/ICECCT.2015.7226154.

Tyagi, A. (1993). A reduced-area scheme for carry-select adders. *IEEE Transactions on Computers*, *42*(10), 1163–1170. https://doi.org/10.1109/12.257703.

Vaidyanathan, P. P. (1993). *Multirate systems and filter banks* (1st ed.). London: Pearson Education.

Vamsi, A. K., Kumar, N. U., Sindhuri, K. B., & Teja, G. S. C. (2018). A systematic delay and power dominant carry save adder design. In *IEEE international conference on smart systems and inventive technology (ICSSIT)* (pp. 918–922). https://doi.org/10.1109/ICSSIT.2018.8748789.

Vijendra, V., & Kulkarni, M (2016). ECG signal filtering using DWT haar wavelets coefficient techniques. In *IEEE international conference on emerging trends in engineering, technology and science (ICETETS)* (pp. 1-6). https://doi.org/10.1109/ICETETS.2016.7603040.

Wang, Z., Jullien, G. A., Miller, W. C., Wang, J. (1993). New concepts for the design of carry lookahead adders. In *IEEE international symposium on circuits and systems* (pp. 1837–1840). https://doi.org/10.1109/ISCAS.1993.394104.

Weste, N. H. E., & Eshraghian, K. (1994). *Principles of CMOS VLSI design* (2nd ed.). London: Addison-Wesley.

Wolf, W. (2004). *FPGA-based system design*. Prentice: Prentice Hall.

Xiang, L. M., Zabidi, M. M. Ah. Awab, A. H., & Ab Rahman, Ab A. (2018). VLSI implementation of a fast Kogge–Stone parallel-prefix adder. *Journal of Physics: Conference Series, 1049*, 1–11. https://doi.org/10.1088/1742-6596/1049/1/012077.

Xilinx ISE In-Depth Tutorial. [Online] Available at https://www.xilinx.com/support/documentation/sw_manuals/xilinx14_1/ise_tutorial_ug695.pdf. Accessed 26 March 2020.

Zhang, J., & Zhang, Q. (2015). Image fusion algorithm based on wavelet transform. In *4th IEEE international conference on advanced information technology and sensor application (AITS)* (pp. 47–50). https://doi.org/10.1109/AITS.2015.19.

Sayantam Sarkar is currently working as Assistant Professor in the Department of Electronics and Communication Engineering at Vijaya Vittala Institute of Technology, Bangalore. He obtained B.E. degree in Electronics and Communication Engineering from Sambhram Institute of Technology, Bangalore and M.Tech Degree in VLSI Designs and Embedded Systems from Dayananda Sagar College of Engineering, Bangalore. Currently he is pursuing Ph.D degree in VLSI design from Visvesvaraya Technological University, Belagavi. He has over 15 research publications in refereed International Journals and Conference Proceedings.His research interests include VLSI Architectures for Image Processing, Biometrics, VLSI forsignal processing applications, Video Processing, VLSI Design and Circuit. He also served as reviewer in many refereed International Journals and Conferences.

Satish S. Bhairannawar is currently working as Dean of Industry Institute Interface (III) and Professor of Electronics and Communication Engineering, Shri Dharmasthala Manjunatheshwara College of Engineering and Technology, Dharwad. Also he is working as a visiting faculty in IIIT, Dharwad. He obtained B.E degree in Electronics and Communication and M.E. Degree in Electronics and Communication from University Visvesvaraya College of Engineering, Bangalore. He also obtained Ph.D. in Computer Science and Engineering at Bangalore University, Karnataka. He has over 30 research publications in refereed International Journals and Conference Proceedings. He also wrote a chapter in a book published by Elsevier on medical image processing. His research interests include VLSI Architectures for ImageProcessing, Biometrics, VLSI for signal processing applications, Video Processing, VLSI Design and Circuit. He is a senior member of IEEE for personal and professional commitment to achievement of technology.