http://ansinet.com/itj

ITJ

ISSN 1812-5638

INFORMATION TECHNOLOGY JOURNAL

ANSInet
Asian Network for Scientific Information
308 Lasani Town, Sargodha Road, Faisalabad - Pakistan

OPEN ACCESS Information Technology Journal

ISSN 1812-5638
DOI: 10.3923/itj.2020.1.11

Research Article
Implementation of Wavelet-based Architecture for Optimization Image Filtering

Ogunlere Samson Ojo, Ajaegbu Chigozirim and Oladejo Daniel Oluwaninyo

Department of Computing and Engineering Sciences, Babcock University, Ilishan-Remo, Ogun State, Nigeria

Abstract

Background and Objective: Existing implementations of wavelet-based image filtering architecture shad design complexities which translated to implementation complexities and low clock frequency. As a result, they failed to meet the requirement for real-time applications. Implementation of image filtering cannot be separated from optimization, hence, optimizations must be continually devised to meet real-time requirements of wavelet-based image filtering. This research aimed at the optimization of existing wavelet-based hardware architecture for embedded image filtering. **Materials and Methods:** This work presents a two-dimensional (2-D) discrete wavelet transform (DWT) architectural design using Verilog Hardware Description Language (Verilog HDL). To reduce implementation complexities, a basic linear algebra approach is used for the Haar transform. A register transfer level (RTL) optimization is proposed to shorten the worst-case filter path delay, thus increasing the maximum operating frequency. Validation of the proposed optimization involved, the design of a circuit for the DWT computation, simulation and implementation in field programmable gate array (FPGA). **Results:** Result achieved shows substantial decreased hardware cost, high signal-to-noise ratio (SNR) and high operating frequency when compared to the existing works used as benchmark. **Conclusion:** This work validates that compensating for the filtering delay in a multi-level wavelet-based image filtering architecture can optimize speed and resource utilization while obtaining a high SNR.

Key words: Image filtering, wavelet-based architecture, embedded platforms, filter delay compensation, signal-to-noise ratio (SNR)

Citation: Ogunlere Samson Ojo, Ajaegbu Chigozirim and Oladejo Daniel Oluwaninyo, 2020. Implementation of wavelet-based architecture for optimization image filtering. Inform. Technol. J., 19: 1-11.

Corresponding Author: Ogunlere Samson Ojo, Department of Computing and Engineering Sciences, Babcock University, Ilishan-Remo, Ogun State, Nigeria

Competing Interest: The authors have declared that no competing interest exists.

Data Availability: All relevant data are within the paper and its supporting information files.

INTRODUCTION

The use of digital images by imaging systems has increased rapidly over the last decade. This rate of increase is directly proportional to the rate of increase in the clock frequency of embedded imaging frameworks over the same period of time[1].

Various fields such as object tracking, machine vision, medical imaging and forensic science require image filtering. In these fields, images need to be analyzed before making decisions. "In medical imaging for instance, high quality imaging is a requirement for analyzing images of unique events. In forensic science, high quality imaging is a requirement where potentially useful photographic evidence is sometimes of extremely bad quality[2]."

All applications of image processing struggle with disturbances which occur when an image is superimposed by another unwanted signal. Digital image processing algorithms, such as filtering algorithms, accept inputs from real-world digitized images and mathematically process them to remove such disturbances. In mathematics, image filtering is a function approximation problem which entails finding the best approximation of the original image in other to distinguish it from a noised version[2].

Image analysis, before filtering, can be effectively carried out by splitting the image into components referred to as wavelets. Wavelet transform of an image is the process of analyzing an image using wavelets. As the name might suggest, a wavelet is a little piece of a wave[3]. Wavelet transform is inherently parallel and it lends itself to hardware implementation on embedded platforms[4]. Most previously proposed architectures have design complexities which translated to implementation complexities (complex control logic) and low clock frequency. As a result, they failed to meet the requirement for real-time applications[5].

The implementation of image filtering hardware architecture cannot be separated from optimization which must be continually devised to meet real-time requirements of wavelet-based image filtering. However, a gap common to these architectures is that the considerable group delays introduced by the wavelet filter are treated as insignificant in the design. These group delays get accumulated when the transform level is increased. Consequently, this causes a significant reduction in maximum operating frequency. Given that computer science is about insight and processing speed is critical to it, there is constantly an increasing need to perform image filtering in the fastest way possible. Thus, current research could be of great importance to the research community of engineers and designers of high-speed, resource-efficient wavelet-based architectures in terms of achieving an optimal and efficient implementation of image filtering.

Some wavelet-based architecture developed and implemented for image filtering was studied. "The following is the summary of closely related works, León *et al.*[6] studied a hardware implementation of the DWT algorithm in 2-D for FPGA, using the Daubechies filter family of order 2 (db2). Algorithm blocks were individually implemented and analyzed before combining then into one system. The complete system comprised of a general control block that determined when each block is activated, thereby generating system monitoring and synchronization of each stage of design. The architecture was implemented using Xilinx ISE software, version 11.2. The processing time of a complete image of 256×256 pixels stored in memory is 36 msec with a working frequency of 50 MHz. Huang *et al.*[7] studied a high-speed architecture for wavelet-based image filtering. The proposed architecture included the input interface module, the high/low filter module and the output interface module. The results of high-pass and low-pass filters were selected to output by a multiplexer module. High-pass and low-pass filters are parallel for optimizing processing speed." The design was synthesized and implemented on an Altera Cyclone II 2C35 development board. A well-known image, Lena, was selected as the verification object. The maximum clock frequency of 114.10 MHz was achieved. Input test image was stored in the Static RAM (SRAM). Hsieh *et al.*[8] studied the efficient architecture with forward-and-inverse DWT. "Folded transformation is used to systematically determine the control circuits. By executing multi-level analysis operations on a single stage, the number of functional unit in the implementation is reduced. It required four adders only for multi-level analysis and achieves higher hardware utilization. The architecture was coded in Verilog HDL, verified by the platform of Quartus-II and implemented in a field-programmable gate array (FPGA). The proposed folded architecture did not require extra memories which are instead required in the existing work used as a benchmark. Basheer and Mohammed[9] investigated an architecture depended on 2-D DWT and a threshold stage. This architecture was made up of a control unit, a processor unit, two on-chip internal memories to speed up system operations and an on-board off-chip external memory. An operating speed of 52.84 MHz was achieved. The proposed architecture was designed and synthesized with the VHDL language and then implemented on the FPGA Spartan 3E starter kit (XC3S500E) to check validation of the results and performance of the design. Jeon *et al.*[2] studied an FPGA architecture to improve filtering speed using total variation algorithm for image noise removal. The image filtering was successful after 10 cycles of operation. Although the operation is complex and requires considerable computing power, noise removal performance is better than prior results. The proposed system achieved up to 250 MHz processing speed with no loss of accuracy. A hardware implementation of image noise filtering using Vertex FPGA[10]. The architecture can compute a large set of filters for both the 2-D forward and inverse transforms. The FPGA has access to two independent banks of SRAM. Experimental results showed that this method proved a fast and easy hardware implementation with few and simple arithmetic computations and occupied less memory storage." A high throughput rate compared with other implementations was achieved, with slight increase in the number of occupied slices. The design of 2-D DWT architecture implemented in VLSIC using Verilog HDL[11]. The wavelet decomposition is verified and obtained in Xilinx 14.2 version software using Verilog HDL. Circuit is planned, modeled (simulated) in Verilog HDL and finally the result is validated in Spartan 6 FPGA to get the output 2D-DWT computed image back. "The result showed that the proposed architecture provides good performance with respect to execution time. Hardware utilization was also efficient which precisely used block memory space. An FPGA implementation of image noise removal using haar wavelet transform (HWT)[12]. The maximum clock frequency of 206.605 MHz was achieved. The performance comparison of existing technique with proposed technique was based on PSNR and hardware utilization. The results show that PSNR value of proposed algorithm is higher compared to the existing algorithm used as a benchmark. It was also observed that the proposed algorithm used lesser hardware resources due to the simplicity of the proposed architecture. Overall, the system used large amounts of external memory for storing the image." This is seen by the authors as a drawback of their architecture. Chuma *et al.*[13] investigated an FPGA implementation of noise filtering using Haar wavelet transform. In this work, an image with noise was applied at a level 5 for haar wavelet transform (DWT) through a threshold and then through an Inverse Wavelet transform (IDWT). The programming of the algorithm in behavioral mode was made in the VHDL. The code was organized in building blocks and the first block is the finite state machine (FSM) that is responsible for control flow and the clock rate. The design was implemented using Altera Quartus 15 development environment. The result of the implementation shows a maximum frequency of 295.6 MHz. Based on the results obtained by the simulation of the implementation, it is acceptable to conclude that the FPGA is a fast and reliable platform. Hasan *et al.*[1] studied an FPGA implementation of Haar Wavelet Hardware Architecture. In their work, they presented 2-D DWT hardware architecture design using (VHDL) hardware description language. To achieve considerable reduction in design complexity, linear algebra approach was adopted to represent the signal flow graph of 2-D DWT architecture. "In order to validate the proposed scheme, a circuit for the DWT computation was designed, simulated and implemented in FPGA." This architecture was described and synthesized with the VHDL based methodology. The VHDL module successfully performed 2-D DWT on images of different dimensions and is capable of varied transform levels. The result was achieved at a substantially decreased hardware space cost block elements using linear equations instead of matrix approach.

The design specifications of existing works studied had no provisions for compensating filter delay at the register transfer level (RTL) of the architecture. The difference between the shortest and worst-case delay of filter paths is not effectively minimized in the existing works studied. "Not compensating for this delay will have a significant effect in reducing the operational frequency of the system.

The study concentrated on registered transfer level (RTL) architectural optimization. The research made use of rectangular 2-D grey-level images corrupted with Gaussian noise. Thus, the purpose of this study is to optimize and implement the wavelet-based architecture for embedded image filtering.

MATERIALS AND METHODS

The finite scale multi-resolution representation of a discrete function is known as discrete wavelet transform (DWT). It is a fast linear operation on a data vector whose length is an integer power of 2. Discrete wavelet transform is invertible, where the inverse transform is the transpose of the transform matrix. The mother wavelet function is quite localized in space, while individual wavelet functions are localized in frequency similar to sine and cosine in Fourier transform."The wavelet function is given as in Kanithi[3]:

$$\psi_{(j, k)}(x) = 2^{j}{}_{2}\psi(2^{j}x-k) \tag{1}$$

And the scaling function is given as:

$$\varphi_{(j,\,k)}\,(x) = 2^{j}{}_{2}\varphi(2^{j}x\text{-}k) \tag{2}$$

where, ψ is called the wavelet function and j and k are integers that scale the wavelet function. The factor 'j' in above equations is known as the scale index, which indicates the wavelet's width. "The location index k provides the position.

There are a number of functions that can be used as the mother wavelet for a wavelet transformation. The choice of mother wavelet depends on the application." One suitable for image is not suited for audio signal. "Therefore, the details of the particular application should be taken into account and the appropriate mother wavelet should be chosen for effectiveness[14]. The choice of the mother wavelet is, however, not unique. The chosen one is in accordance with the problem to be solved. The simplicity of operation and representation plays a crucial role. Haar wavelet is one of the oldest and simplest wavelet. It is fast and memory-efficient since it is calculated without the use of any temporary array. It is mostly used in image signal processing.

Multi-resolution analysis using filter banks: The resolution of the image, which is a measure of the amount of detail information in the image, is determined by filtering operations. DWT is computed by successive low-pass and high-pass filtering of the discrete time-domain image signal as shown in Fig. 1. The samples generated by the high-pass filters are completely decomposed. Other samples generated by the low-pass filters are applied to the next level of computation for further decomposition. This process of computation is referred to as Mallat algorithm[14].

In the Fig. 1, the signal is denoted by the sequence x[n], where n is an integer. The low pass filter is denoted by G_o while the high pass filter is denoted by H_o. At each level, the high pass filter produces detail information d[n], while the low pass filter associated with scaling function produces coarse approximations, a[n]. The filtering and decimation-by-2 process is continued until the desired level is reached. The maximum number of levels depends on the length of the signal[14].

Wavelet shrinkage technique: Because the wavelet transform decomposes the signal neatly into approximations (low frequency) and details (high frequency) sub-bands, the detail values will contain much of the noise." This suggests a method for filtering the image: simply reduce the size of the detail values before using them to synthesize the signal. This approach is called thresholding[9].

Thresholding types: Thresholding types include Hard and Soft thresholding. Hard thresholding can be described as the process of setting to zero the elements whose absolute values are lower than the threshold. "Soft thresholding is an extension of hard thresholding. This works by first setting to zero the elements whose absolute values are lower than the threshold and then shrinking the non-zero values toward zero. The best kind of thresholding depends on the application. Soft thresholding is the best type of thresholding for image filtering[15].

Hard and soft thresholding are given in Eq. 3 and 4 respectively:

$$\text{Hard threshold}: \begin{cases} y = x \text{ if } |x| > \lambda \\ y = 0 \text{ if } |x| < \lambda \end{cases} \tag{3}$$

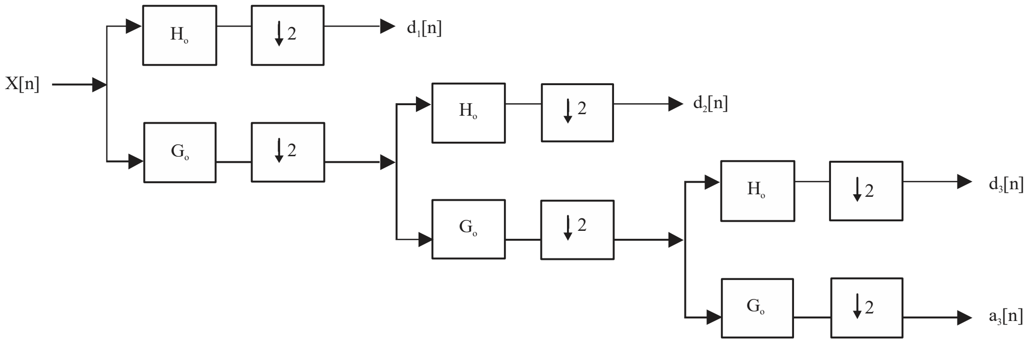


Fig. 1: Three-level wavelet analysis tree

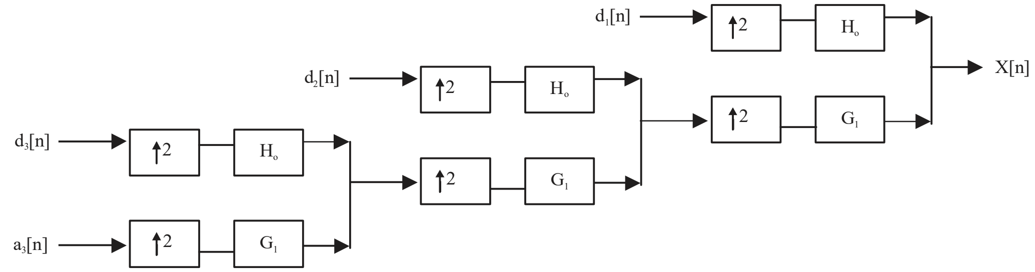


Fig. 2: Three-level wavelet synthesis tree

$$\text{Soft threshold: } \{y = \text{sign}(x)\,(|x|\text{-}\lambda) \qquad (4)$$

where, x is the input signal, y is the signal after thresholding and λ is the threshold value."

Multi-resolution synthesis using filter banks: Figure 2 illustrates the synthesis of the original image from the wavelet values. Basically, the synthesis is the reverse process of the analysis. The approximation and detail values at each level are up-sampled by two, passed through the low-pass and high-pass synthesis filters and then summed up. "This process is continued through the same number of levels as in the analysis process to obtain the original signal. The Mallat algorithm works equally well if the analysis filters, G_0 and H_0, are exchanged with the synthesis filters, G_1 and H_1[14].

Image filtering algorithm: This section describes the image filtering algorithm. The algorithm is very simple to implement and computationally more efficient. It has following steps[16]:

Step 1 : Perform multi-scale wavelet transform of the noisy image using Haar wavelet
Step 2 : For each level, compute the scale parameter
Step 3 : Compute threshold λ
Step 4 : Apply soft or hard thresholding to the noisy pixel values
Step 5 : Filter the high frequency sample values
Step 6 : Merge low frequency coefficient with the filtered high frequency coefficient
Step 7 : Invert the multi-scale DWT to synthesize the filtered image

Filter performance measurement

Mean square error: Mean square error (MSE) is defined as:

$$\text{MSE} = \frac{1}{\text{mn}}\sum_{0}^{m-1}\sum_{0}^{n-1}\| f(i,\ j) - g(i,\ j)\|^2 \qquad (5)$$

where, f represents the matrix data of the original image, g represents the matrix data of the degraded image, m represents the numbers of rows of pixels of the image and i represents the index of that row, n represents the number of columns of the pixels of the image and j represents the index of that column[17].

Peak signal-to-noise ratio (PSNR): The peak signal to noise ratio (PSNR) is the value of the noisy image with respect to that of the original image. This ratio is often used as a quality measurement between the original and the reconstructed image." The higher the PSNR, the better the quality of the reconstructed image[17]:

$$\text{PSNR} = 10 * \log_{10}\frac{1}{\text{MSE}} \qquad (6)$$

System model: An overview of the proposed optimized system architecture is given in Fig. 3. The proposed architecture is designed based on the Harvard architecture.

Module description: The hardware modules that make up the proposed architecture are described as follows:

DWT processing module: The DWT processing module is the central part of the proposed optimized architecture. It is responsible for the row and column operations. The thresholding is also incorporated in this module. In generating each low-pass and high-pass filter components, the HWT algorithm performs averaging a and differencing d on a pair of noisy image pixel values using Eq. 7 and 8. "Then the algorithm shifts over by two values and calculates another average and difference on the next pair of values":

$$\text{Low-pass } a = \frac{1}{2}(x(a) + x(b)) \qquad (7)$$

$$\text{High-pass } \ b = \frac{1}{2}(x(a) - x(b)) \qquad (8)$$

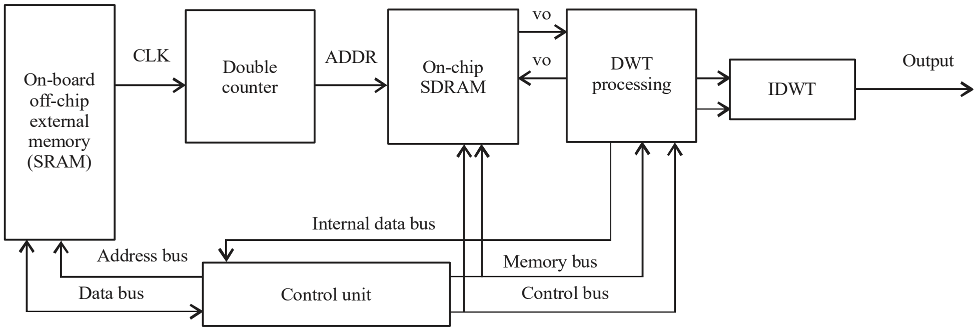


Fig. 3: Top-level model of the proposed architecture

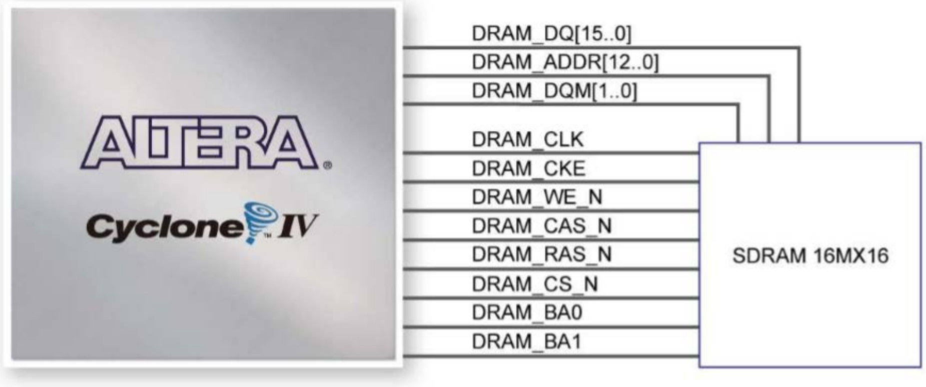


Fig. 4: Connections between FPGA and SDRAM

The above linear equations are adopted to reduce complexity of the design and enhance computational speed. In addition it minimizes hardware expenditure and entails a reversible sequence of additions, subtractions and divisions. This makes it compatible with energy-efficient hardware implementation.

This module is a combinational hardware incorporated with adders and right shifters which generate the average components. Appropriate number of multiplexers (without any additional logic gates) is used to implement the right shifters that accept 4-bit strings and generate 4-bit strings. The output is fed back into a subtractor to generate the difference components. “The combinational hardware is supplied with data from input memory (SRAM).

On-chip internal memory: This is the embedded synchronous dynamic random access memory (SDRAM) device providing 32 MB with a 16-bit data line connected to the FPGA. All signals are registered on the positive edge of the clock signal, DRAM_CLK. Other SDRAM interface signals include clock enable (CKE), write enable (WE), column address select (CAS), row address select (RAS), chip select (CS) and bank address (BA). This module is used with the DWT processing module for writing wavelet transform results in the addressed memory locations. A 16-bit word can be written into the SDRAM by entering the address of the desired location, specifying the data to be written.

The SDRAM controller is designed using the Verilog HDL language to be appropriate for the proposed design. A universal asynchronous receiver transmitter (UART) design is used for reading SDRAM contents. This design is loaded on the FPGA chip to provide serial data communication between the DE0 Nano board and the personal computer. Connections between FPGA and SDRAM is presented in Fig. 4.

The DWT processing block is given the start address of the memory space filled by the current line vector of pixels. The line type is determined through SDRAM_CAS control signal.” The line type also includes the style employed to store image data (either row by row or column by column). The SDRAM_CAS signal is set to 0 to carry out the horizontal pass initialization in SDRAM_RAS. The process of vertical pass is started, after all the lines in the horizontal pass are completed and ready = '1'. The width of the current line is equal to the image size at level one. The image size is subsequently divided

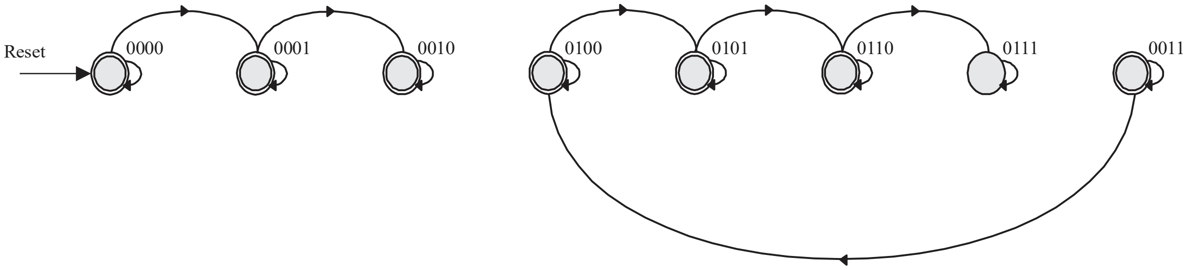


Fig. 5: Deterministic FSM

in two at each new level. "Following FDWT implementation on one line vector of the input image pixels, the module generates an internal (Ready) signal in waiting for the next line." Afterwards, the 2-D-DWT module gives the requisite information for the next line. The process of vertical pass is started, after all the lines in the horizontal pass are completed. The Low-pass and High-pass wavelet components for the FDWT are computed and saved in the course of the horizontal and vertical pass phases.

IDWT module: This module is responsible for performing inverse discrete wavelet transform on the stored wavelet values in order to synthesize the filtered image."The hardware comprises adders devoid of the shift to the right operation.

"The computed FDWT data is written in the SRAM, with the help of multiplexers, where the IDWT begins." The IDWT output image is stored in a text file in the SDRAM. Matlab software is used to read that file. This reconstructed image is validated on the target FPGA by sending it bit by bit to the board. This data is returned as a text file by the FPGA.

External memory entity: This is a static random access memory (SRAM) mega function created in Verilog to specify how the memory read and write operation is implemented for the 256×256 Lena test image data. A phase locked loop (PLL) is used to distribute precisely timed clock pulses within the RAM. "A PLL is a control system that comprises of an oscillator and a phase detector. The oscillator generates a periodic signal of 50 MHz as output while the phase detector compares this output with the phase of the periodic input image signal to keep them in lock (keep input and output frequencies the same). This is called signal synchronization. The input image signal is stored in this memory using the DE0 control panel.

Control unit module: The major purpose of this module is to develop control signals that are needed to gain access to the memory as well as the essential signals for 2-D DWT processing. The most powerful FSM, a deterministic (unique) FSM, incorporated with two counters is chosen for this module. The state machine bubble diagram in Fig. 5 shows the operation of eight-state automation that accepts or rejects strings of four bits and only produces a unique computation of the automation for each input string.

In the Fig. 5, there are eight states denoted graphically by circles. For each state, there is a transition arrow leading out to the next state. Upon reading a symbol, the state machine jumps deterministically from one state to another by following the transition arrow.

Counter machine module: This is an abstract machine used to model the discrete time-steps in the DWT computation in relation to memory accesses for each respective computational step to avoid the simultaneous writing operation by two or more SRAM threads to the same memory address. It is used in the process of designing the parallel HWT algorithm. It comprises of a set of one or more unbounded registers, each of which can hold a single non-negative integer and a list of (usually sequential) arithmetic and control instructions for the machine to follow. It is a sequential digital logic circuit with an input line called the clock. The value of the output line represents a number in the binary or BCD number system. Each pulse applied to the clock input increments (up counts) the number in the counter. The counter is a type of memory to store a single natural number (initially zero) and can be arbitrarily long. The counter is used along with the FSM. The role of the FSM is to check whether the counter is zero and increment the counter by one.

Proposed optimization: The matrix multiply-accumulate (MAC) approach for computing the HWT algorithm requires a long chain of adders and multipliers." This approach elongates the worst-case delay path and limits the maximum frequency of the system. To meet the requirements for real-time application, this work proposes a unique combination of design parameters to shorten the worst-case delay and increase the maximum frequency of the system.

In this architecture, linear algebra approach is used in place of the MAC approach. The implementation of this algorithm involves the use of adders and right shifters. The absence of multipliers reduces the worst-case delay to just adder delays. The right shifters are implemented using multiplexers. This ensures the reuse of hardware resource without additional logic gates. A PLL is introduced in the architectural design for clock signal synchronization. A double-counter based deterministic FSM is used to model the control logic of the system. This unique combination enhanced the maximum operating frequency while minimizing hardware cost.

System development tools: The software tools used in the development of this system include the MATLAB software and the electronic design automation tools of the DE0 Nano evaluation board.

The MATLAB program is used for the following:

- To add noise to the original image
- To convert noisy image to vector form
- To display the restored image after noise removal
- To compute the PSNR

The EDA tools of the DE0 Nano FPGA board are used for this work. The tools include:

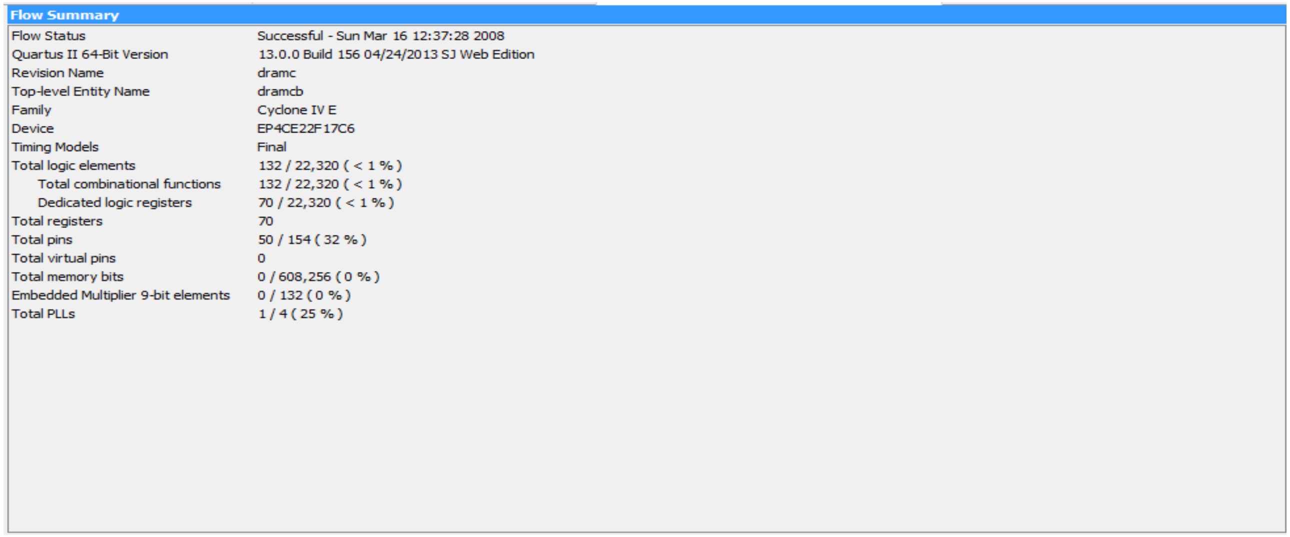
- Verilog HDL: This is used for entry of the data that represents the project design
- Modalism altera: This is used for simulation to generate the machine code directly from the compiler. "Thus, faster compilation and runtime speed-up are enabled
- Quartus II: The Quartus II software, student edition version 10.2, provided by Altera Corporation is used for synthesis and implementation of the design on the target FPGA

RESULTS

The DE0 Nano board's programmable logic has different resources available in different quantities for the implementation of re-configurable designs. The flow summary of synthesis of the proposed architecture is shown in Fig. 6.

Filtering performance analysis result: In other to test the noise removal performance of the implementation, the output image file is downloaded from the DE0 Nano board to the PC and read into an array by a MATLAB script. This script then computes the PSNR of the output image. The filtering performance analysis result comparison of the proposed optimized architecture, resulted in a higher value of 40.46 peak signal to noise ratio (PSNR) for the same.

Maximum operating frequency result: The maximum operating frequency result shown in Fig. 7a-c, represented the output image file downloaded from the DE0-Nano board to the PC with the original image. From Fig. 7, Image (c) shows clearer picture when compared with (b) after the introduction of Gaussian noise to the original image (a). Results of time quest timing analyzer are presented in Fig. 8, while showing the Fmax summary, the results was low Fmax 232.99 MHz, with restricted Fmax of 232.99 MHz and highest 351.99 MHz while restricted Fmax is 250 MHz.



Flow Summary

Flow Status	Successful - Sun Mar 16 12:37:28 2008
Quartus II 64-Bit Version	13.0.0 Build 156 04/24/2013 SJ Web Edition
Revision Name	dramc
Top-level Entity Name	dramcb
Family	Cyclone IV E
Device	EP4CE22F17C6
Timing Models	Final
Total logic elements	132 / 22,320 (< 1 %)
Total combinational functions	132 / 22,320 (< 1 %)
Dedicated logic registers	70 / 22,320 (< 1 %)
Total registers	70
Total pins	50 / 154 (32 %)
Total virtual pins	0
Total memory bits	0 / 608,256 (0 %)
Embedded Multiplier 9-bit elements	0 / 132 (0 %)
Total PLLs	1 / 4 (25 %)

Fig. 6: Summary of synthesis in Altera Quartus

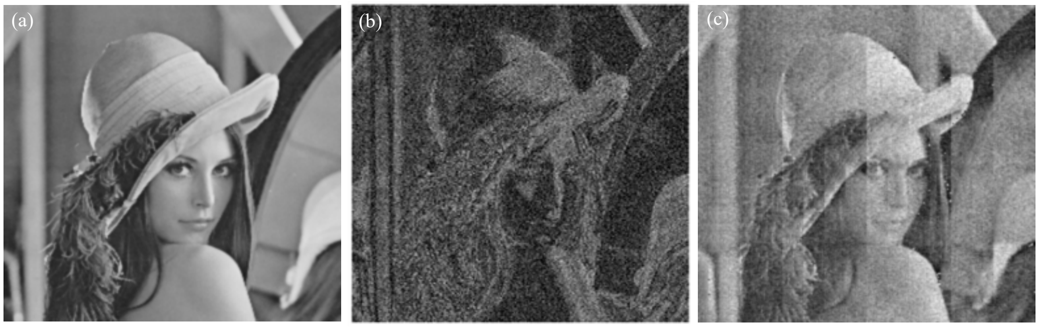
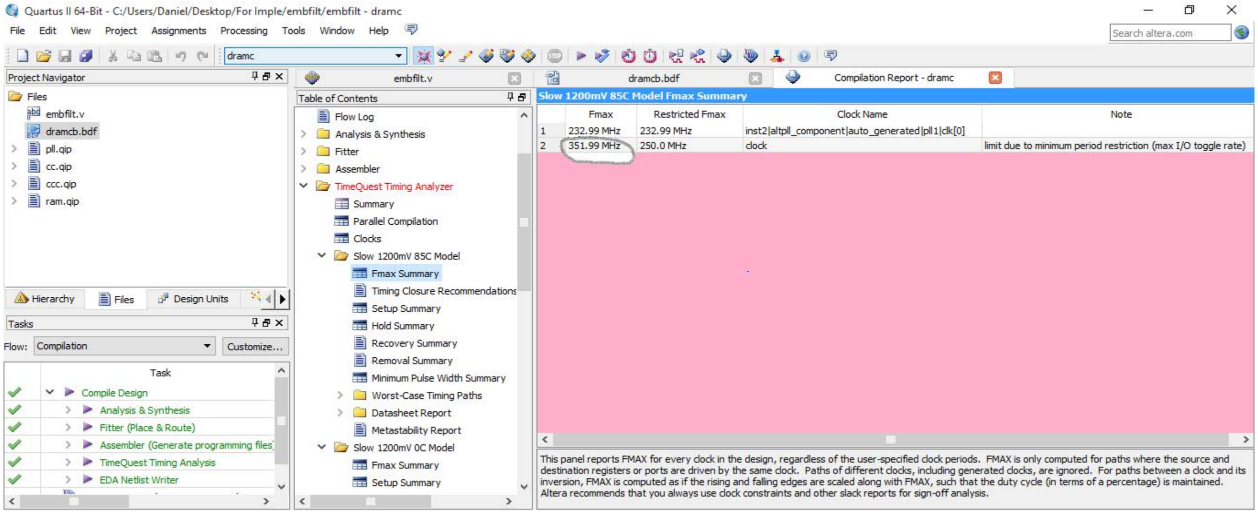


Fig. 7(a-c): Lena test images, (a) Original image, (b) Image with noise and (c) Filtered image


Quartus II 64-Bit - C:/Users/Daniel/Desktop/For Imple/embfilt/embfilt - dramc

Slow 1200mV 85C Model Fmax Summary

	Fmax	Restricted Fmax	Clock Name	Note
1	232.99 MHz	232.99 MHz	inst2|altpll_component|auto_generated|pll1|clk[0]	
2	351.99 MHz	250.0 MHz	clock	limit due to minimum period restriction (max I/O toggle rate)

This panel reports FMAX for every clock in the design, regardless of the user-specified clock periods. FMAX is only computed for paths where the source and destination registers or ports are driven by the same clock. Paths of different clocks, including generated clocks, are ignored. For paths between a clock and its inversion, FMAX is computed as if the rising and falling edges are scaled along with FMAX, such that the duty cycle (in terms of a percentage) is maintained. Altera recommends that you always use clock constraints and other slack reports for sign-off analysis.

Fig. 8: Result of time quest timing analyzer

DISCUSSION

This discussion is based on the key findings and performance analysis comparison to the previously published works from the reviewed literatures.

The FPGA resource usage analysis result comparison shows that the proposed optimized architecture in this work utilized fewer hardware resources of 132/22, 320 (0.59%) than the work used as benchmark[12,2], with a total logic element resources of 725/4, 656 (15.57%). Hasan *et al.*[1] traded off FPGA resource usage operational speed and throughput for hardware cost when compared to the proposed work.

Also, the filtering performance analysis result comparison of the proposed optimized architecture in this work, resulted in a higher value of 40.46 peak signal to noise ratio (PSNR) for the same Lena test image corrupted with Gaussian noise of density 0.15 dB when compared to the benchmark[12] result of 36.46 PSNR ratio. Hsieh *et al.*[8] studied the performance evaluation in terms of hardware utilization only, not in reference to PSNR.

Furthermore, this work achieved a maximum operating frequency of 351.99 MHz as shown in the captured timing analyzer of Fig. 8 which is higher than the value 206.605 MHz achieved[6,7,9,12]. The maximum operating frequency of an output image file downloaded from the DE0-Nano board to the PC. In comparison, image (7c) shows clearer picture when compared with (7b) after the introduction of Gaussian noise to the original image (7a). This is an improvement, with a minimal hardware resources, in the FPGA resource usage analysis result comparison. Hasan *et al.*[1] traded off FPGA resource usage operational speed and throughput for hardware cost when compared to the proposed work.

It should be noted that the design specifications of existing works studied have no provisions for compensating filter delay at the register transfer level (RTL) of the architecture. The difference between the shortest and worst-case delay of filter paths is not effectively minimized in the existing works studied. Not compensating for this delay will have a significant effect in reducing the operational frequency of the system. The performance of the system is depleted as level of transform increases. The operational frequency can be optimized further by a further reduction in design complexities at the RTL.

CONCLUSION

This work presents a two-dimensional (2-D) discrete wavelet transform (DWT) architectural design using verilog hardware description language (Verilog HDL). The investigations carried out in this work validates that compensating for the filtering delay in a multi-level wavelet-based image filtering architecture can optimize speed and resource utilization while obtaining a high SNR. The filtered image is affected by the fixed phase noise of the PLL frequency synthesizer of the targeted FPGA. It is recommended that the idea in this work be extended to various domains of multi-level wavelet-based image processing such as image compression, estimation and transformation. It is also recommended that embedded platforms with low phase noise PLL synthesizers be used in further studies.

SIGNIFICANCE STATEMENT

This study discovers the optimization of wavelet-based embedded images that can be beneficial for image filtering. This study will help the researcher to uncover the critical areas of high speed, resource-efficient wavelet-based architectures in term of achieving an optimal and efficient implementation of image filtering that many researchers were not able to explore. Thus a new theory on wavelet-based architecture for improving image filtering may be arrived at.

REFERENCES

1. Hasan, K.K., K.J. Ali and M.A. Dhan, 2018. An FPGA implementation of Haar wavelet filter hardware architecture. Proceedings of the International Conference on Electrical, Electronics, Computers, Communication, Mechanical and Computing (EECCMC), January 28-29, 2018, Tamil Nadu, India, pp: 18-23.
2. Jeon, B., S. Lee, J. Jin, D.D. Nguyen and J.W. Jeon, 2013. Design and implementation of hardware architecture for denoising using FPGA. Proceedings of the 2013 IEEE 9th International Colloquium on Signal Processing and its Applications, March 8-10, 2013, IEEE., Kuala Lumpur, Malaysia, pp: 83-88.
3. Kanithi, A., 2011. Study of spatial and transform domain filters for efficient noise reduction. Master Thesis, National Institute of Technology, Rouekela, India.
4. Kamarujjaman, M. Mukherjee and M. Maitra, 2015. An efficient FPGA based de-noising architecture for removal of high density impulse noise in images. Proceedings of the International Conference on Research in computational Intelligence and Communication Networks, Kolkata, India, November 20-22, 2015, IEEE., pp: 262-266.
5. Hidayat, R., S. Aji and Litasari, 2014. Implementation of haar wavelet transform on xilinx spartan-3E FPGA. Proceedings of the Conference on Applied Electromagnetic Technology (AEMT), April 11-15, 2014, Lombok, pp: 167-171.
6. León, M., L. Barba, L. Vargas and C.O. Torres, 2011. Implementation of the 2-D wavelet transform into FPGA for image. J. Phys.: Conf. Ser. Vol. 274, No. 1. 10.1088/1742-6596/274/1/012052.
7. Huang, Q., Y. Wang and S. Chang, 2011. High-performance FPGA implementation of discrete wavelet transform for image processing. Proceedings of the 2011 Symposium on Photonics and Optoelectronics (SOPO), May 16-18, 2011, IEEE., Wuhan, China, pp: 1-4.
8. Hsieh, C.F., T.H. Tsai, C.H. Lai and S.C. Yi, 2013. An efficient architecture of 1-D discrete wavelet transform for noise reduction. Int. J. Adv. Comput. Technol., 5: 412-419.
9. Basheer, N.M. and M.M. Mohammed, 2013. Image denoising using FPGA based 2D-DWT architecture. Int. J. Recent Technol. Eng., 2: 92-97.
10. Ashok, C. and V.S. Anu, 2015. FPGA implementation of image denoising using adaptive wavelet thresholding. Int. J. Adv. Res. Comput. Commun. Eng., 4: 288-292.
11. Khan, G.F. and A.G. Sawant, 2016. Spartan 6 FPGA implementation of 2D-discrete wavelet transform in verilog HDL. Proceedings of the IEEE International Conference on Advances in Electronics, Communication and Computer Technology (ICAECCT), December 2-3, 2016, Pune, India, pp: 139-143.
12. Ramesha, K.P., K.V. Ramanareddy and S. Yellampalli, 2016. FPGA implementation of image de-noising using Haar wavelet transform. Int. J. Eng. Res. Technol., 5: 112-115.
13. Chuma, E.L., L.G.P. Meloni, Y. Iano and L.L.B. Roger, 2017. FPGA implementation of a de-noising using Haar level 5 wavelet transform. Brasileiro Telecomunicações Processamento Sinais, 26: 1189-1192.

14. Sripathi, D., 2003. Efficient implementations of discrete wavelet transforms using FPGA. M.Sc. Thesis, Department of Electrical and Computer Engineering, Florida State University Libraries, USA.
15. Ergen, B., 2012. Signal and Image Denoising Using Wavelet Transform. In: Advances in Wavelet Theory and Their Applications in Engineering, Physics and Technology, Baleanu, D. (Ed.)., InTech, China, ISBN: 978-953-51-0494-0, pp: 495-514.
16. Satpathy, S., M.C. Pradhan and S. Sharma, 2015. Comparative study of noise removal algorithms for denoising medical image using Labview. Proceedings of the International Conference on Computational Intelligence and Communication Networks (CICN), Jabalpur, India, December 12-14, 2015, IEEE., pp: 300-305.
17. Kaur, S., 2015. Noise types and various removal techniques. Int. J. Adv. Res. Electron. Commun. Eng., 4: 226-230.