Global Journal of Computer Science and Technology: F
Graphics & Vision
Volume 14 Issue 2 Version 1.0 Year 2014
Type: Double Blind Peer Reviewed International Research Journal
Publisher: Global Journals Inc. (USA)
Online ISSN: 0975-4172 & Print ISSN: 0975-4350

Modified Distributive Arithmetic based 2d-Dwt for Hybrid (Neural Network-Dwt) Image Compression

By Mr. Murali Mohan.S & Dr. P. Satyanarayana
Sri Venkatweswara College of Engineering & Technology, India

Abstract- Artificial Neural Networks (ANN) is significantly used in signal and image processing techniques for pattern recognition and template matching. Discrete Wavelet Transform (DWT) is combined with neural network to achieve higher compression if 2D data such as image. Image compression using neural network and DWT have shown superior results over classical techniques, with 70% higher compression and 20% improvement in Mean Square Error (MSE). Hardware complexity and power dissipation are the major challenges that have been addressed in this work for VLSI implementation. In this work, modified distributive arithmetic DWT and multiplexer based DWT architecture are designed to reduce the computation complexity of hybrid architecture for image compression. A 2D DWT architecture is designed with 1D DWT architecture and is implemented on FPGA that operates at 268 MHz consuming power less than 1W.

Keywords: DWT, neural network, image compression, VLSI implementation, high speed, low power, modified DAA.

GJCST-F Classification: F.1.1

Strictly as per the compliance and regulations of:

Modified Distributive Arithmetic based 2d-Dwt for Hybrid (Neural Network-Dwt) Image Compression

Mr. Murali Mohan.S [α] & Dr. P. Satyanarayana [σ]

Abstract- **Artificial Neural Networks (ANN) is significantly used in signal and image processing techniques for pattern recognition and template matching. Discrete Wavelet Transform (DWT) is combined with neural network to achieve higher compression if 2D data such as image. Image compression using neural network and DWT have shown superior results over classical techniques, with 70% higher compression and 20% improvement in Mean Square Error (MSE). Hardware complexity and power dissipation are the major challenges that have been addressed in this work for VLSI implementation. In this work, modified distributive arithmetic DWT and multiplexer based DWT architecture are designed to reduce the computation complexity of hybrid architecture for image compression. A 2D DWT architecture is designed with 1D DWT architecture and is implemented on FPGA that operates at 268 MHz consuming power less than 1W.**

Keywords: DWT, neural network, image compression, VLSI implementation, high speed, low power, modified DAA.

I. Introduction

Image compression is one of the most promising subjects in image processing. Images captured need to be stored or transmitted over long distances. Raw image occupies memory and hence need to be compressed. With the demand for high quality video on mobile platforms there is a need to compress raw images and reproduce the images without any degradation. Several standards such as JPEG200, MPEG-2/4 recommend use of Discrete Wavelet Transforms (DWT) for image transformation [1] which leads to compression with when encoded. Wavelets are a mathematical tool for hierarchically decomposing functions in multiple hierarchical sub bands with time scale resolutions. Image compression using Wavelet Transforms is a powerful method that is preferred by scientists to get the compressed images at higher compression ratios with higher PSNR values [2]. It is a popular transform used for some of the image compression standards in lossy compression methods. Unlike the discrete cosine transform, the wavelet transform is not Fourier-based and therefore wavelets do a better job of handling discontinuities in data. On the other hand, Artificial Neural Networks (ANN) for image compression applications has marginally increased in recent years. Neural networks are inherent adaptive systems [3][4][5][6]; they are suitable for handling nonstationaries in image data. Artificial neural network can be employed with success to image compression. Image Compression Using Neural Networks by Ivan Vilovic [7] reveals a direct solution method for image compression using the neural networks. An experience of using multilayer perceptron for image compression is also presented. The multilayer perceptron is used for transform coding of the image. Image compression with neural networks by J. Jiang [8] presents an extensive survey on the development of neural networks for image compression which covers three categories: direct image compression by neural networks; neural network implementation of existing techniques, and neural network based technology which provide improvement over traditional algorithms. Neural Networks-based Image Compression System by H. Nait Charif and Fathi. M. Salam [9] describes a practical and effective image compression system based on multilayer neural networks. The system consists of two multilayer neural networks that compress the image in two stages. The algorithms and architectures reported in these papers sub divided the images into sub blocks and the sub blocks are reorganized for processing. Reordering of sub blocks leads to blocking artifacts. Hence it is required to avoid reorganization of sub blocks. One of the methods was to combine neural networks with wavelets for image compression. Image compression using wavelet transform and a neural network was suggested previously [10]. Wavelet networks (WNs) were introduced by Zhang and Benveniste [11], [12] in 1992 as a combination of artificial neural networks and wavelet decomposition. Since then, however, WNs have received only little attention. In the wavelet networks, the basis radial functions in some RBF-networks are replaced by wavelets. Szu et al. [13], [14] have shown usage of WNs for signals representation and classification. They have explained how a set of WN, "a super wavelet", can be produced and the original ideas presented can be used for the assortment of model. Besides, they have

Author α: Associate Professor, Dept. of ECE, Sri Venkatweswara College of Engineering & Technology Chittoor, A.P., India.
e-mail: muralimohan.vlsi.dsp@gmail.com
Author σ: Professor, College of Engineering, S.V.University, Tirupati, A.P., India. e-mail: satyamp1@yahoo.com

mentioned the big compression of data achieved by such a representation of WN's. Zhang [15] has proved that the WN's can manipulate the non-linear regression of the moderately big dimension of entry with the data of training. Ramanaiah and Cyril [16] in their paper have reported the use of neural networks and wavelets for image compression. Murali et al. [17] reports use of neural networks with DWT improves compression ratio by 70% and MSE by 20%. The complexities of hardware implementation on VLSI platform are not discussed in this paper. Murali et. al [18] reports the use of FPGA for implementation of neural network and DWT architecture, the design operates are 127 MHz and consumes 0.45 mW on Virtex-5 FPGAs. Sangyun et. al., [19] in their work have proposed a new logic for distributive arithmetic algorithm and have designed for FIR filters. The develop logic is optimized for low power applications. In their work, the LUT coefficients are computed based on a suitable number system, and are stored in LUT. Low power techniques such as block enabling logic, memory bank logic and clock gating logic have been used for optimization. However, the work does not consider the FPGA resources for power optimization; as well the developed architecture is suitable for higher order filter coefficients. Hence there is a need for customized architecture for DWT filters that can efficiently utilize the FPGA resources. Cyril P. Raj, et. al., [20] in their work have developed a parallel and pipelined distributive arithmetic architecture for DWT, the design achieves higher throughput and lower latency, but consumes large area on FPGA. The symmetric property of DWT coefficients have not been used to reduce hardware complexities. Chengjun Zhang, Chunyan Wang, and M. Omair Ahmad [21] propose a scheme for the design of pipeline architecture for fast computation of the DWT is developed. The goal of fast computation is achieved by minimizing the number and period of clock cycles. The main idea used for minimizing these two parameters is to optimally distribute the task of the DWT computation among the stages of the pipeline and to maximize the inter- and intra-stage parallelisms of the pipeline. In this paper 2D-DWT architecture is designed and implemented on VLSI platform for optimizing area, timing and power. Section II presents theoretical background on neural networks and DWT. Section III discusses the image compression architecture using DWT and ANN technique, section IV presents VLSI implementation of DWT architecture and conclusion is presented in section V.

II. Neural Networks and DWT

In this section, neural network architecture for image compression is discussed. Feed forward neural network architecture and back propagation algorithm for training is presented. DWT based image transformation and compression is also presented in this section. Compression is one of the major subject of research, the need for compression is discussed as follows [17]: Uncompressed video of size 640 x 480 resolution, with each pixel of 8 bit (1 bytes), with 24 fps occupies 307.2 Kbytes per image (frame) or 7.37 Mbytes per second or 442 Mbytes per minute or 26.5 Gbytes per hour. If the frame rate is increased from 24 fps to 30 fps, then for 640 x 480 resolution, 24 bit (3 bytes) colour, 30 fps occupies 921.6 Kbytes per image (frame) or 27.6 Mbytes per second or 1.66 Gbytes per minute or 99.5 Gbytes per hour. Given a 100 Gigabyte disk can store about 1-4 hours of high quality video, with channel data rate of 64Kbits/sec – 40 – 438 secs/per frame transmission. For HDTV with 720 x 1280 pixels/frame, progressive scanning at 60 frames/s: 1.3Gb/s – with 20Mb/s available – 70% compression required – 0.35bpp. In this work we propose a novel architecture based on neural network and DWT [18].

a) Feed forward neural network architecture for image compression

An Artificial Neural Network (ANN) is an information- processing paradigm that is inspired by the way biological nervous systems, such as the Brian, process information [16]. The key element of this paradigm is the novel structure of the information processing system. The basic architecture for image compression using neural network is shown in fig. 1. The network has input layer, hidden layer and output layer. Inputs from the image are fed into the network, which are passed through the multi layered neural network. The input to the network is the original image and the output obtained is the reconstructed image. The output obtained at the hidden layer is the compressed image. The network is used for image compression by breaking it in two parts as shown in the Fig. 1. The transmitter encodes and then transmits the output of the hidden layer (only 16 values as compared to the 64 values of the original image). The receiver receives and decodes the 16 hidden outputs and generates the 64 outputs. Since the network is implementing an identity map, the output at the receiver is an exact reconstruction of the original image.

Three layers, one input layer, one output layer and one hidden layer, are designed. The input layer and output layer are fully connected to the hidden layer. Compression is achieved by designing the network such that the number of neurons at the hidden layer is less than that of neurons at both input and the output layers. The input image is split up into blocks or vectors of 8 X8, 4 X 4 or 16 X 16 pixels. Back-propagation is one of the neural networks which are directly applied to image compression coding [20][21][22].

In the previous sections theory on the basic structure of the neuron was considered. The essence of the neural networks lies in the way the weights are updated. The updating of the weights is through a

definite algorithm. In this paper Back Propagation (BP) algorithm is studied and implemented.

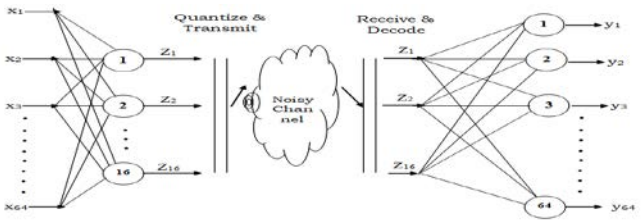


Figure 1 : Feed forward multilayered neural network architecture [19]

b) DWT for Image Compression

The DWT represents the signal in dynamic sub-band decomposition. Generation of the DWT in a wavelet packet allows sub-band analysis without the constraint of dynamic decomposition. The discrete wavelet packet transform (DWPT) performs an adaptive decomposition of frequency axis. The specific decomposition will be selected according to an optimization criterion. The Discrete Wavelet Transform (DWT), based on time-scale representation, provides efficient multi-resolution sub-band decomposition of signals. It has become a powerful tool for signal processing and finds numerous applications in various fields such as audio compression, pattern recognition, texture discrimination, computer graphics [24][25][26] etc. Specifically the 2-D DWT and its counterpart 2-D Inverse DWT (IDWT) play a significant role in many image/video coding applications. Fig. 2 shows the DWT architecture, the input image is decomposed into high pass and low pass components using HPF and LPF filters giving rise to the first level of hierarchy. The process is continued until multiple hierarchies are obtained. A1 and D1 are the approximation and detail filters.

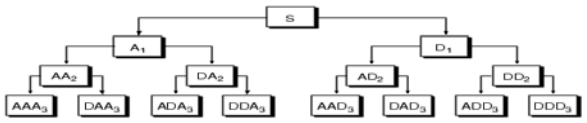
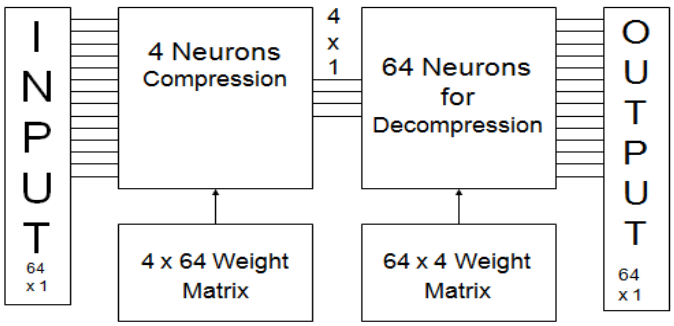


Figure 2 : DWT decomposition

Fig. 3 shows the decomposition results. The barbera image is first decomposed into four sub bands of LL, LH, HL and HH. Further the LL sub band is decomposed into four more sub bands as shown in the fig.. The LL component has the maximum information content as shown in fig. 3, the other higher order sub bands contain the edges in the vertical, horizontal and diagonal directions. An image of size N X N is decomposed to N/2 X N/2 of four sub bands. Choosing the LL sub band and rejecting the other sub bands at the first level compresses the image by 75%. Thus DWT assists in compression. Furhter encoding increases compression ratio.

Figure 3 : DWT decomposition of barbera image into hierarchical sub bands

III. ANN with DWT for Image Compression



Network size	Size of hidden Layer	Compression Ratio
64-64-64	64	0%
64-32-64	32	50%
64-16-64	16	75%
64-08-64	08	87.5%
64-04-64	04	93.75%
64-01-64	01	98.5%

Figure 4 : Neural Network based Image Compression

Basic architecture for image compression using neural network is shown in the above fig. 4. The input image of size 64 x 1 is multiplied by 4 x 64 weight matrixes to obtain the compressed output of 4 x 1, at the receiver 4 x 1 is decompressed to 64 x 1 by multiplying the compressed matrix by 64 x 4. The table in fig. 4 shows the compression ratio that can be achieved by choosing the sizes of hidden layer.

Prior to use of NN for compression it is required to perform training of the network, in this work we have used back propagation training algorithm for obtaining the optimum weights and biases for the NN architecture. Based on the training, barbera image is compressed and decompressed; Fig. 5 shows the input image, compressed image and decompressed image.

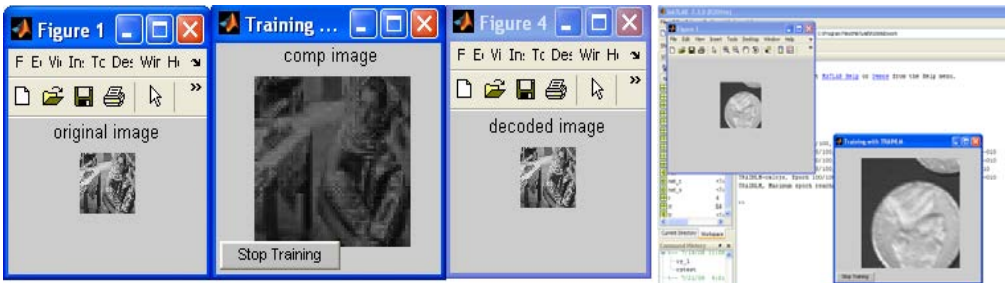


Figure 5 : NN based Image Compression and Decompression

Fig. 5 also shows the input image and the decompressed image of coins image using neural network architecture. From the decompressed results shown, we find the checker blocks error, which exists on the decompressed image. As the input image is sub divided into 8 x 8 blocks and rearranged to 64 x 1 input matrixes, the checker block arises. This is one of the limitations of NN based compression. Another major limitation is the maximum compression ration which is less than 100%, in order to achieve compression more than 100% and to eliminate checker box errors or blocking artifacts we proposed DWT combined with NN architecture for image compression.

a) Image Compression using DWT-ANN

Most of the image compression techniques use either neural networks for compression or DWT (Discrete wavelet Transform) based transformation for compression. In order to overcome the limitations of NN architecture in this work, DWT is used for image decomposition and an N X N image is decomposed using DWT into hierarchical blocks the decomposition is carried out until the sub block is of size 8 x 8. For a image of size 64 x 64, first level decomposition gives rise to 32 x 32 (four sub bands) of sub blocks, further decomposition leads to 16 x 16 (sixteen sub bands), which can further decamped to 8 x 8 at the third hierarchy. The third level of hierarchy there are 64 sub blocks each of size 8 x 8. Fig. 6 shows the decomposition levels of input image of size 64 x 64.

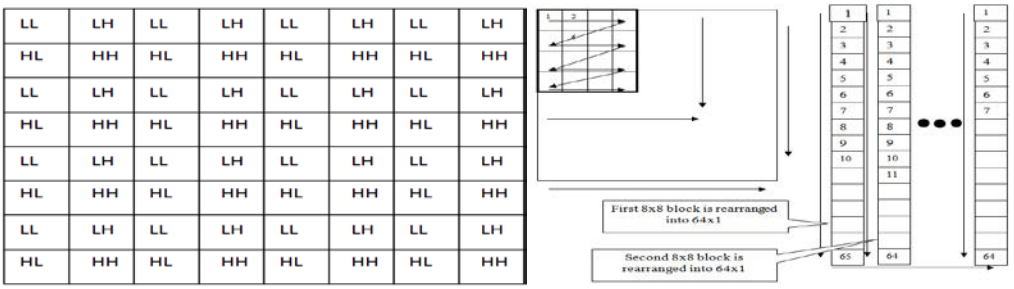


Figure 6: Decomposition of image into sub blocks using DWT

Sub blocks of 8 x 8 are rearranged to 64 x 1 block are combined together into a rearranged matrix size as shown in fig. 6. The rearranged matrix is used to train the NN architecture based on back propagation algorithm. In order to train the NN architecture and to obtain optimum weights it is required to select appropriate images [17][18]. The training vectors play a vital role in NN architecture for image compression. The NN architecture consisting of input layer, hidden layer and output layer. The network functions such as tansig and purelin are used to realize feed forward neural network architecture [18]. In this work, hybrid neural network architecture is realized using DWT combined with ANN. The hybrid architecture is discussed in [Ramanaiah and Cyril]. The NN based compression using analog VLSI is presented in [Cyril and Pinjare]. Based on the two different papers neural network architecture is developed and is trained to compress and decompress multiple images. The DWT based image compression algorithm is combined with neural network architecture. There are several wavelet filters and neural network functions. It is required to choose appropriate wavelets and appropriate neural network functions. In this work an experimental setup is modeled using Matlab to choose appropriate wavelet and appropriate neural network function. Based on the above parameters chosen the Hybrid Compression Algorithm is developed and is shown in Fig. 7.

Figure 7 : Proposed hybrid algorithms for image compression [16]

Several images are considered for training the network, the input image is resized to 256 x 256, the resized image is transformed using DWT, 2D DWT function is used for the transformation. There is several wavelet functions, in this work Haar and dB4 wavelet functions are used. The input image is decomposed to obtain the sub band components using several stages of DWT. The DWT process is stopped until the sub band size is 8 x 8. The decomposed sub band components are rearranged to column vectors; the rearranged vectors are concatenated to matrix and are set at the input to the neural network. The hidden layer is realized using 4 neurons and tansig function. The weights are biases obtained after training are used to compress the input to the required size and is further processed using weights and biases in the output layer to decompress. The decompressed is further converted from vector to blocks of sub bands. The sub band components are grouped together and are transformed using inverse DWT. The transformation is done using multiple hierarchies and the original image is reconstructed. The input image and the output image are used to compute MSE, PSNR. A detailed discussion on DWT with NN for image compression and the performance results are presented in [17][18]. One of the major challenges in this work is the hardware complexity of DWT and NN architecture. In order to reduce the computation complexity on hardware platform, in this work a modified architecture for DWT is proposed, design, modeled and implemented on VLSI platform. Next section discusses the modified architecture.

IV. Distributive Arithmetic Architecture for FIR Filters

DWT is realized using low pass and high pass FIR filters. In an FIR filter, the incoming signal is processed by the filter coefficients to produce the output samples. The filters coefficients are designed or identified based on the required specifications and are used in design of filter architecture. Fig. 9 shows the basic block of FIR filter. The relation between input, output and filter coefficients are related using convolution sum.

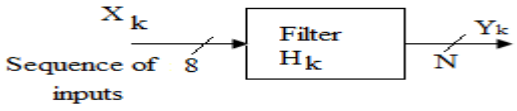


Figure 9 : FIR filter

The convolution algorithm is given by equation (1)

$$Y_k = H_k * X_k \text{ for all values of k.} \quad (1)$$

The convolution operation is basically sum of products. Thus the convolution operation in equation (1) can be expressed as in equation (2)

$$Y_k = \sum_{k-1}^{N} H_k X_k \quad (2)$$

X_k = Input samples, H_k = Filter coefficients, Y_k = Output and N = Filter order length

In general X_k and Y_k are represented using 2's complement number system. Thus representing both positive and negative values of input and filter samples. In 2's complement format X_k is represented as, $X_k = \{b_{k0}, b_{k1}, b_{k2} \ldots \ldots b_{kL-1}\}$, where L is the number of bits or word length. In 2's complement number system

MSB = 1 implies it is a negative number and thus sign extension is carried out. For analysis X_k can be mathematically represented as in equation (3)

$$X_k = -b_{k0} + \sum_{n=1}^{L-1} b_{kn}\, 2^{-n} \quad (3)$$

Where b_{k0}= sign bit and b_{kn} = binary bits representing magnitude.

Substituting (3) in (2), equation (1) can be expressed as in equation (4)

$$Y = \sum_{k=1}^{N} H_k[-b_{k0} + \sum_{n=1}^{L-1} b_{kn}\, 2^{-n}] \quad (4)$$

Rearranging . equation (4), equation (5) is obtained,

$$Y = \sum_{n=1}^{N}\left[\sum_{k=1}^{k} H_k b_{kn}\right] 2^{-n} + \sum_{k=1}^{k} H_k(-b_{k0})\, k = 1 \quad (5)$$

equation (5) has two terms, the first term is with the magnitude and the second term is with the sign bit. Considering 1st term in equation (5)

$Y = \sum_{n=1}^{N}\left[\sum_{k=1}^{k} H_k b_{kn}\right]2^{-n}$ Which can be expanded for every value of n. Assuming k=4 and expanding term $Y = \left[\sum_{k=1}^{k} H_k b_{k1}\right]2^{-n}$ can be written as

$$Y = H_1 b_{11} 2^{-1} + H_2 b_{21} 2^{-1} + H_3 b_{31} 2^{-1} + H_4 b_{41} 2^{-1} \quad \text{for n=1} \quad (6)$$
$$Y = H_1 b_{12} 2^{-2} + H_2 b_{22} 2^{-2} + H_3 b_{32} 2^{-2} + H_4 b_{42} 2^{-2} \quad \text{for n=2} \quad (7)$$
$$Y = H_1 b_{13} 2^{-3} + H_2 b_{23} 2^{-3} + H_3 b_{33} 2^{-3} + H_4 b_{43} 2^{-3} \quad \text{for n=3} \quad (8)$$
$$Y = H_1 b_{14} 2^{-4} + H_2 b_{24} 2^{-4} + H_3 b_{34} 2^{-4} + H_4 b_{44} 2^{-4} \quad \text{for n=4} \quad (9)$$

From the above two equation the following are the observations

1. The coefficients remain constant as they are fixed coefficients
2. The b_{11} term represents the first MSB bit of the first input sample X1, b_{21} represents the first MSB of second input sample X2 and so on. From equation. (6), it is understood that the MSB bits of X1, X2, X3 and X4 are multiplied with the filter coefficients H1, H2, H3 and H4. As the binary bits can be '1' or '0', there are 16 possible partial products of filter coefficients. Thus the 16 possible partial products of filter coefficients can be pre-computed and stored in a memory, the first MSB bits of input samples can

be used as address to the memory and can be used to access the memory contents. Thus avoiding multiplication process.

3. The equation (7) is similar to equation (6), the only difference is the binary bits are the second MSB bits of input samples. Similarly, as discussed previously there are 16 possible combinations of partial products that can be accessed.
4. Comparing equation (6) to equation (7), each bit of input samples are used in accessing the memory contents and have to be added with the previous partial products. Before every addition is performed the partial products are to be right shifted by 1 bit position as the terms 2^{-1}, 2^{-2} and so on.

Thus the term $\left[\sum_{k=1}^{k} H_k b_{kn}\right]2^{-n}$ has 2^k possible values. The coefficients are fixed and hence 2^k combination of coefficients can be pre-computed and stored in a LUT (ROM). The LUT depth is2^k, and width of LUT can be (L+1), where L is the maximum width of filter coefficients

Fig. 10 shows the top level block diagram of DA algorithm. The DA architecture consists of input registers, which are SISO registers that can be sequentially loaded with the input samples.

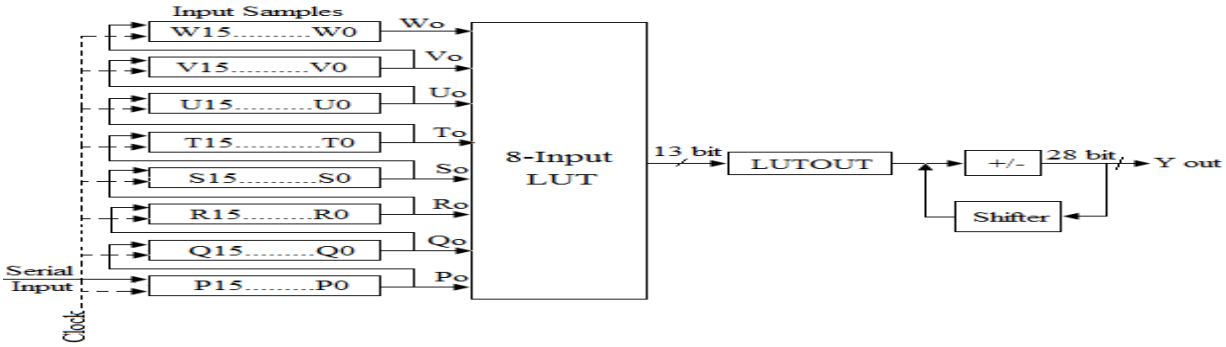


Figure 10 : Distributive Arithmetic Architecture

The input samples X = [W, V, U, T, S, R, Q, P], each of 16-bit width is loaded serially into the serial in serial out shift register. The LSBs of the SISO register forms the address to the LUT. As there are 8 SISO (number of SISO is decided by the order of the filter), there will be 8 LSB bits, hence 8 address lines. The depth of LUT will be 256, and the width of LUT will be (L+1) bit. At the output side of LUT, there is an accumulator unit along with right shift register. The input sample is of size 16 bits, hence it requires 16*8 clock cycles to load the SISO register serially. At 17th clock cycle the LSB form the address of LUT, the first partial product is fetched and accumulated with the right shift register contents. As the width of each SISO is 16 bit, the SISO registers are serially shifted and hence requires 16 clock cycles. Thus to compute one output sample it requires 16*8 clock cycles for loading and 16 clock cycles for reading partial products and accumulation. Hence the first output sample is available after 16*8 + 16 clock cycles. The second output sample is computed by further loading a new sample at the bottommost SISO register, which requires 16 clock cycles. Once new sample is loaded, it requires another 16 clock cycles to accumulate partial products. Thus the latency is 144 clock cycles and throughput is 32 clock cycles.

a) Modified DA based DWT architecture

The limitations of DA architecture is that as the number of inputs increase from 8 to 16, the size of LUT is 216. In order to reduce the LUT size, the inut sample can be split into two halves of 4 each. The first 4 SISO register accesses the top LUT and the bottom 4 SISO accesses the bottom LUT. The size of top and bottom LUT is 24, and thus the total size of the LUT is 32. As the LUTs are split into two sections, the output of each LUT is independent and the accumulated data is further added to compute the final output. In the split DA architecture shown in Fig. 11 the LUT size is reduced from 28 to 2*24 and the number of adders are increased from 1 to 3 compared with the basic DA logic.

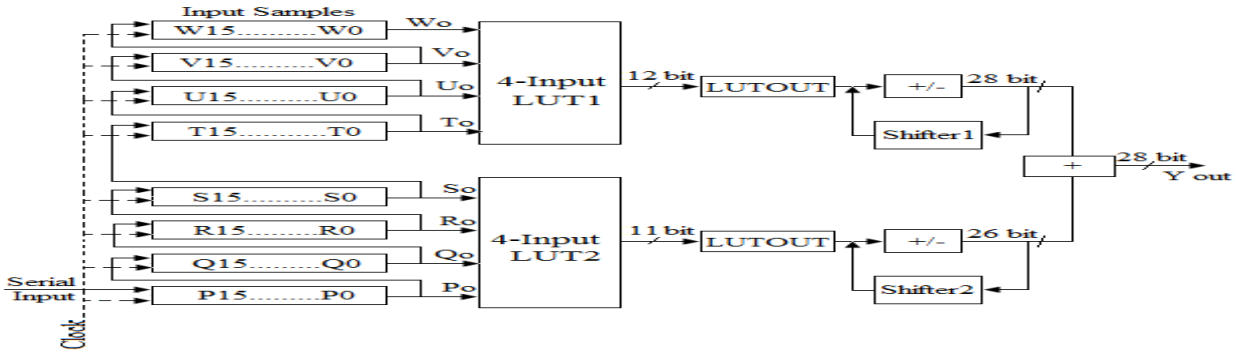


Figure 11 : Split DA architecture

In this work, 9/7 filter based DWT is chosen for modulation and demodulation. Table 1 shows the 9/7 filter coefficients. As there exist symmetry in the 9/7 filter coefficients, the modified equations for high pass and low pass filters can be expressed as follows:

Table 1 : Low pass and high pass filter coefficients for 9/7 filter

Order	Co-efficient	Values
4	0.0267487	27
3	-0.0168641	-17
2	-0.0782232	-8
1	0.02668641	273
0	0.6029490	617
8	0.2668641	273
7	-0.0782232	-80
6	-0.0168641	-17
5	0.0267487	27

Order	Co-efficient	values
3	0.0912717	47
2	-0.0575435	29
1	-0.5912717	-303
0	1.11150870	569
6	-0.5912717	-303
5	-0.0575435	29
4	0.0912717	47

From the Table 1, as there are 9 low pass filter coefficients, and 7 high pass filter coefficients, the output samples can be expressed as in equation (10) and equation (11) respectively.

$$Y_L = X_0h_0 + X_1h_1 + X_2h_2 + X_3h_3 + X_4h_4 + X_5h_5 + X_6h_6 + X_7h_7 + X_8h_8 \quad (10)$$

$$Y_H = X_0g_0 + X_1g_1 + X_2g_2 + X_3g_3 + X_4g_4 + X_5g_5 + X_6g_6 \quad (11)$$

In order to realize the low pass and high pass filters using DA logic the depth of low pass LUT will be 2^9 and high pass LUT will be 2^7. In order to optimize the size of LUT, the symmetric property of filter coefficients are considered and the equation (10) and equation (11) are rewritten as in equation (12) and equation(13),

$$Y_L = X_0h_0 + (X_1 + X_8)h_1 + (X_2 + X_7)h_2 + (X_3 + X_6)h_3 + (X_4 + X_5)h_4 \quad (12)$$

$$Y_H = X_0g_0 + (X_1 + X_6)g_1 + (X_2 + X_5)g_2 + (X_3 + X_4)g_3 \quad (13)$$

Thus in order to realize the filter the low pass LUT depth is 2^5 and high pass LUT depth is 2^4. Thus the total depth of LUT for DWT computation is (2^5+2^4) compared to the original LUT depth of (2^9+2^7). Thus the memory size is reduced by 97.5%. However, it is observed that the number of adders required is 5 and 4 for the low pass and high pass filter respectively. In this research work, one of the major contributions is the design of DA architecture that combines split DA logic with symmetric property of filters. The modified DA architecture is shown in Fig. 12.

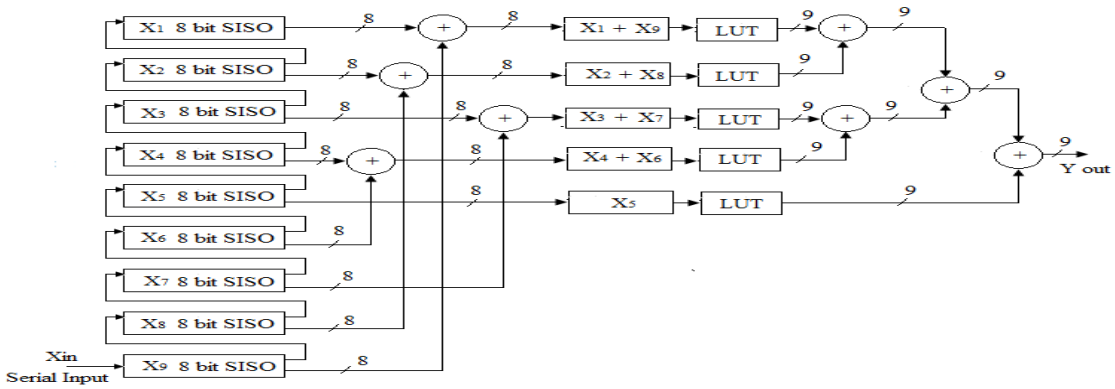


Figure 12 : Modified DA algorithm for low pass filter

In the modified DA logic, the input samples are sequentially loaded into the SISO registers, it requires 8*9 clock cycles (the width of input samples are considered to be 8 bit wide), after the initial load operations are performed, the input samples are added using the first stage adders and the out of the adder is stored in the second stage PISO register, the addition and loading of second stage PISO register requires one clock cycle. The PISO registers in the second stage are split into two halves, and are further used in accessing the LUTs. As two PISO registers accesses one LUT, the LUT depth is 4. The bottom LUT is accessed by 3 PISO registers, and thus the depth is 8. The total LUT size (depth) is 12 (8 + 4). The output of each LUT is accumulated to compute the final output of the low pass filter used in DWT. Thus the number of adders required for low pass output filter computation is 7 and the LUT depth is 12. Similarly the architecture for high pass filter using modified DA logic can be designed. Fig. 13 shows the modified DA logic for high pass filter used in DWT. The depth of LUT is 8 (4 + 4), and the number of adders required are 6.

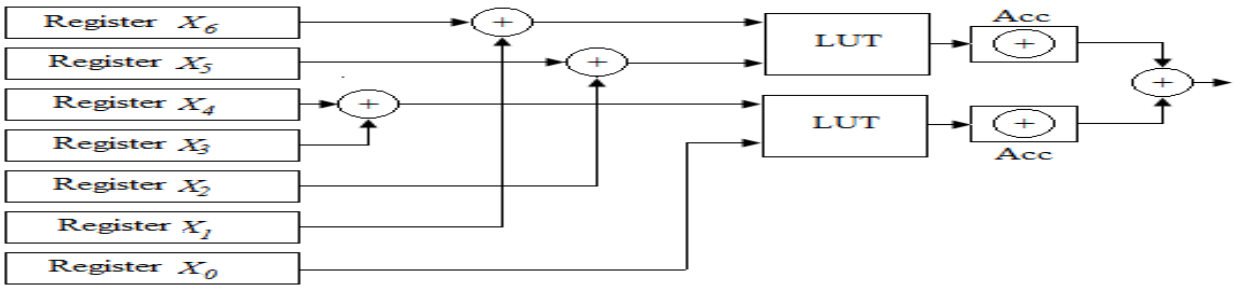


Figure 13 : Modified DA algorithm for high pass filter

To load the low pass SISO and high pass SISO it requires 9*8 clock cycles and 7*8 clock cycles respectively. After loading the first stage SISO, it is required to add the samples and store the samples into the second stage PISO, which requires one clock cycle. The PISO registers are used in accessing the LUTs and thus it requires 9 clock cycles (the width of input samples is 8 bit, after addition the width of each sample is 8+1 bit, thus 9 clock cycles are required to access the LUTs). Thus the first output from low pass filter is available at 9*8 + 1 + 9 clock cycles and the first output from high pass filter is available at 7*8 + 1 + 9 clock cycles. Hence the latency is 82 and 66 clock cycles respectively. The first stage and second stage adders are isolated with the use of SISO and PISO registers, thus the addition in the first stage and the accumulation in the second stage can be performed simultaneously. Thus the loading of SISO register can be done in parallel, thus reducing one clock cycle, and the throughput in low pass and high pass filter output computation is found to be 9 and 9 clock cycles respectively. Table 2 shows the comparison of various DA algorithms for DWT computation.

Table 2 : Comparison of computation complexity in DA architecture

	High Pass			Low Pass		
	DA	Split DA	Modified DA	DA	Split DA	Modified DA
LUT Size	$2^7 = 128$	$2^4 + 2^3 = 24$	8	2^9	$2^5 + 2^4$	12
Latency	8*9+8	8*7+8	7*8+1+9=66	8*9+8	8*9+8	9*8+1+9
Throughput	16	16	9	16	16	9
Adders	1	3	6	1	3	7
SISO	Required	Required	Required	Required	Required	Required
PISO	Not required	Not required	Required	Not required	Not required	Required

From the Table 3, it is found that the proposed DA logic reduces the LUT size from 512 to 12 in low pass and 128 to 8 in high pass filter computation. The number of adders is increased; however, the throughput is 9 for both low pass and high pass computation. The proposed architecture is modeled using Verilog HDL and is verified for its functionality using suitable test cases. A test environment is developed to test the logic correctness of the proposed DA logic. From the simulation results obtained in ModelSim, the developed HDL model is found to produce correct results for all possible test vectors. The proposed model is implemented using Xilinx ISE and is targeted on Virtex devices. The implementation results are discussed in detail in next chapter. Another approach for DWT computation is using multiplexers based approach. Next section discusses the multiplexers based approach with DA for DWT computation.

b) Multiplexer based DA for DWT

The split DA logic discussed in the previous section uses two LUT structure to store the pre-computed partial products, which are accessed by the

SISO registers. Exploiting the symmetric property of DWT filter coefficients the split DA logic was further modified and the LUT size is reduced. In the modified DA logic discussed in the previous section, a PISO register is introduced between the first stage and second stage adders, thus this may increase the memory size and add to area complexity. In order to eliminate PISO registers, MUX based logic is proposed and designed in this work.

The modified DA logic based architecture is optimized for area and speed performances, however, when the design is implemented on FPGA, there are limitations. FPGA consists of Configurable Logic Blocks (CLBs), dedicated RAM (block RAM), dedicated multipliers and routing resources. CLB consists of LUTs, registers (flip flop), multiplexers, fast carry adders and buffers. As the modified DA logic uses LUTs and adders, the multiplexer logic and registers are not utilized within a CLB. Thus for implementation of modified DA logic more number of CLBs is utilized and every CLB resource is not completely utilized. Hence in order to utilize the resources fully within a CLB, a novel FIR filter architecture is proposed and implemented.

$$Y_k = \sum_{K=0}^{N-1} H_k X_k \text{ N=9 (or) 7 for DWT filters.} \quad (14)$$

The above equation can be expanded and written as equation (15),

$$Y_k = \sum_{k=0}^{4} H_k X_k + \sum_{k=5}^{8} H_k X_k \quad (15)$$

Equation (15) can be realized using DA algorithm and mux based logic in order to fully utilize the FPGA resources. Equation (15) consists of two terms, the first term is realized using mux based logic and the second term is realized using split DA logic. The Term $\sum_{k=5}^{8} H_k X_k$ is realized using split DA logic and is as shown in Fig. 14.

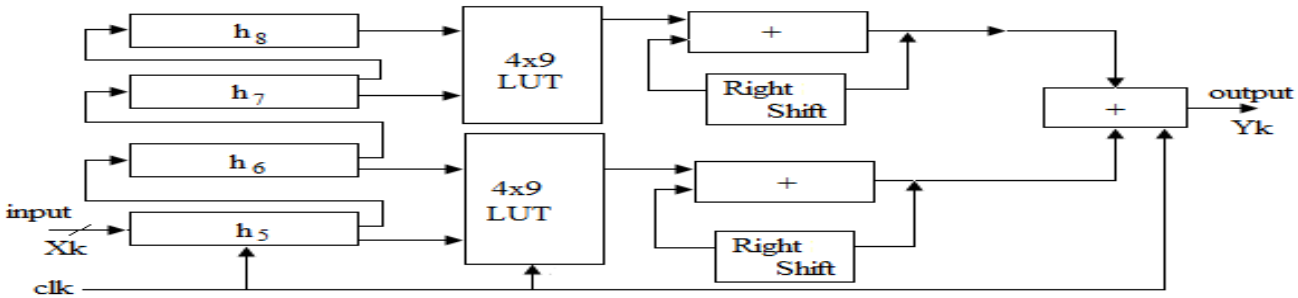


Figure 14 : Split DA logic architecture for second term of Eq (15)

The first term in equation(15) is realized using mux based logic. Consider the first term $Y_{k1} = \sum_{k=0}^{4} H_k X_k$, Expanding the term equation (16) is obtained,

$$Y_{k1} = H_0X_0 + H_1X_1 + H_2X_2 + H_3X_3 + H_4X_3 \quad (16)$$

As the filter parameters (H) are fixed coefficients, and X_k being binary number the term H_0X_0 can be expressed as $H_0X_0 = H_0[X_0^7\ X_0^6\ X_5^5\ X_0^4\ X_0^3\ X_0^2\ X_0^1\ X_0^0]$, where X_0^7 is the MSB and X_0^0 is LSB. Multiplication of H_0X_0 is performed by checking individual bits ofX_0, if X_0^0 is '1' then H_0 is the first partial product else if X_0^0 is '0' then the partial product is all zeros. Similarly every bit of X_0 is checked for its weight and the H_0 coefficient is added with the previous bit partial product. Prior to addition $H_0X_0^0$ partial product should be shifted right by 1 bit and added with X_0^1 H_0 partial product. In order to realize equation (16) using multiplexer, as there are five terms, five 2:1 multiplexers are required. One input of the multiplexers is the filter coefficient $H_0\ H_1\ H_2\ H_3\ H_4$ and the other input is all zeros. The X_0^n bit forms the select line of the multiplexer. If X_0^n bit is 1 then the output of mux is zero else the output of the mux is the corresponding filter coefficient. After every output is chosen from the mux for every bit of input sample, the outputs are accumulated and the final product is computed. Fig. 16 shows the mux based filter design for the first term of equation (15). The use of multiplexers and adders in computing the filter output eliminates the use of LUTs and hence at the input of every multiplexer two registers are required one stores the filter coefficient and the other is hardwired to ground as shown in Fig. 15. The multiplexer based logic is combined with split DA logic in computation of low pass filter outputs.

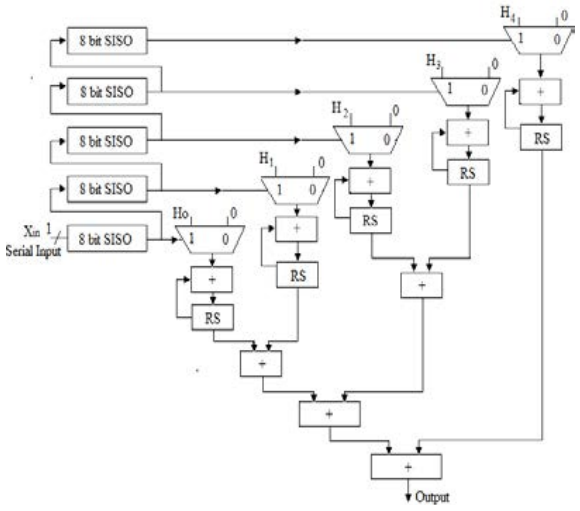


Figure 15 : Mux based architecture for first term of Eq. (15)

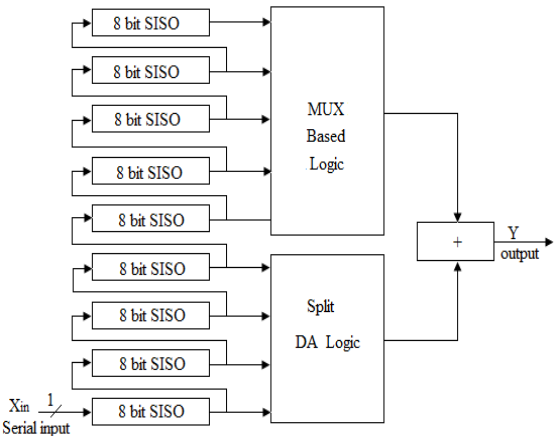


Figure 16 : Novel architecture for DWT using split DA and mux logic

Figure17 shows the novel architecture that combines mux based logic with split DA logic. The serial input is used for sequentially loading the SISO register. It requires 8 clock cycles to serially load each register. Thus to load all the 9 registers it requires 9*8 clock cycles. The LSB outputs of top five registers are used as select lines to multiplexer logic, and the LSBs of the other four registers are used as addresses to the split DA logic. The mux based logic and split DA logic uses 8 clock cycles to compute the output samples and at the end of 9th clock cycle the final output is computed by adding the partial products of mux based logic with split DA logic. Similarly, the novel architecture for high pass filter can be designed and is shown in Fig. 17.

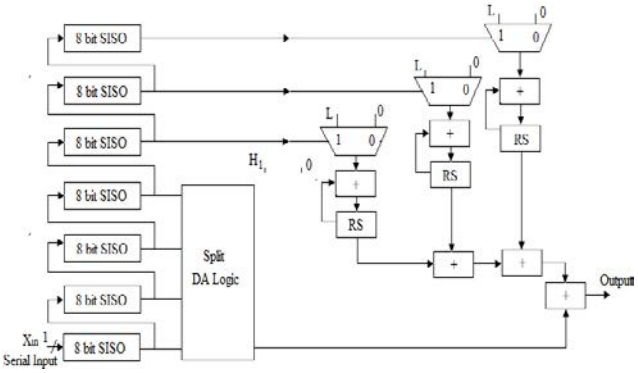


Figure 17 : Novel architecture for high pass filter using split DA and mux logic

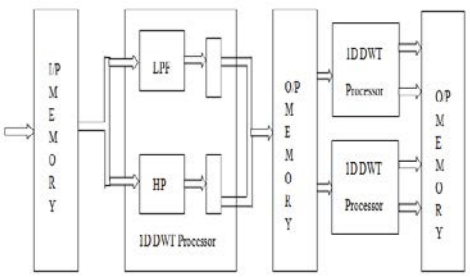


Figure 18 : 2D DWT architecture

Symmetric property of 9/7 filter coefficients can be further used in optimizing the area and speed performances of the DWT architecture. Table 5 presents the performance parameters of the novel DWT architecture designed using mux with split DA logic.

Table 4 : Performance parameters of novel architecture

Performance Parameters	Low pass filter	High pass filter
LUT size	2 LUTs of size 4x9	2 LUTs of size 4x9
Number of multiplexers	4	3
Number of adders	5	3
Number of accumulators	6	4
Throughput	17	17
Latency	9*8 + 8 +1	7*8 + 8 + 1
CLB utilization	100%	100%

The advantage of novel algorithm for DWT computation is that it fully utilizes the CLB resources and hence the area occupancy on FPGA is optimized. As the filter coefficients are biorthogonal, the IDWT processor can be realized just by interchanging the high pass and low pass filters used for DWT computation. The designed 1D DWT architecture is used to compute 2D DWT for the input image. The top level architecture for 2D DWT processor is implemented using the modified 1D-DWT architecture discussed in Fig. 12 and Fig. 13. The 2D-DWT processor consists of input memory, output memory and three 1D-DWT processors as shown in Fig. 18.

HDL code for 1D DWT processor, input memory and output memory is developed and are integrated to top module. The top module is verified using test bench written in Verilog and with know set of input vectors. The simulation results and synthesis results are obtained using Xilinx ISE. The synthesis results obtained are verified with various constraints options provided in the tool. The default options were producing best results. The area report in terms of slices, the power report and timing report have been generated and are reported in this work. Conventional DWT architecture was realized in [19] on Spartan device hence the results reported have been used for comparison. In order to compare the performance improvements in the proposed architecture, the conventional DWT architecture is modeled using HDL and implemented on Virtex-5 device. The results obtained are reported in table 6.

Table 5 : Comparison of 2-D DWT architecture

Parameters	Conventional DWT [19] (on Spartan)	Conventional DWT	Proposed Design
No of Slices	566 out of 768	31105 out of 69120 45%	7235 out of 69120 12%
No of gates	37K	31105 out of 69120 45%	7235 out of 69120 42%
Clock Speed	36MHZ	237 MHz	268 MHz
Power dissipation	51mW	1.37 W	0.9 W

From the comparison results it is demonstrated that the proposed architecture consumes very less resources, as the multipliers are replaced with shift operations, the operating frequency is increased to 268 MHz and power dissipation is reduced by setting the low power constraints. One of the major challenges in the design is data synchronization in DWT computing, as the shift operations are used for multiplication operation, it is mandatory to carefully design the control unit to keep track of the data output and read the data into register for further computation and hence there is need for a predesigned control logic to monitor the data flow logic.

V. Conclusion

Use of NN for image compression has superior advantage compared with classical techniques, however the NN architecture requires image to be decomposed to several blocks of each 8 x 8, and hence introduces blocking artifact errors and checker box errors in the reconstructed image. In order to overcome the checker errors in this work, we have used DWT for image decomposition prior to image compression using NN architecture. In this work, we proposed a hybrid architecture that combines NN with DWT and the input image is used to train the network. The network architecture is used to compress and decompress several images and it is proven to achieve better MSE compared with reference design. The hybrid technique uses hidden layer consisting of tansig function and output layer with purelin function to achieve better MSE. In order to reduce the computation complexity of DWT architecture in this work two different architectures for DWT computation is proposed, designed and implemented on FPGA. The modified DA algorithm and the Multiplexer based DA algorithm is designed to reduce the number of logic gates and to improve throughput on FPGA platform. The 2D DWT architecture is designed with the proposed 1D DWT architecture and the design is implemented on FPGA that operates at a maximum speed of 268 MHz with power consumption less than 1W. The proposed design can be integrated with NN architecture for hybrid architecture for image compression.

References Références Referencias

1. Sang Yoon Park, and Pramod Kumar Meher, Low-Power, High-Throughput, and Low-Area Adaptive FIR Filter Based on Distributed Arithmetic, IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS-II: EXPRESS BRIEFS, pp.1-5.
2. Chengjun Zhang, Chunyan Wang, M. Omair Ahmad: A Pipeline VLSI Architecture for Fast Computation of the 2-D Discrete Wavelet Transform. IEEE Trans. on Circuits and Systems 59-I(8): 1775-1785 (2012).
3. Cyril Prasanna Raj P., Review of 2D VLSI architectures for image Compression, SASTech Journal, Vol. 2, Issue 4, pp:23_27, 2006.
4. Zang, Wavelet Network in Nonparametric Estimation. IEEE Trans. Neural Networks, 8(2):227-236, 1997.
5. Q. Zang and A. Benveniste, Wavelet networks. IEEE Trans. Neural Networks, vol. 3, pp. 889-898, 1992.
6. A. Grossmann and B. Torrésani, Les ondelettes, Encyclopedia Universalis, 1998.
7. R. Baron. Contribution à l'étude des réseaux d'ondelettes, Thèse de doctorat, Ecole Normale Supérieure de Lyon, Février 1997.
8. C. Foucher and G. Vaucher. Compression d'images et réseaux de neurones, revue Valgo n°01-02, 17-19 octobre 2001, Ardèche.
9. J. Jiang. Image compressing with neural networks – A survey, Signal processing: Image communication, ELSEVIER, vol. 14, n°9, 1999, pp. 737-760.
10. S. Kulkarni, B. Verma and M. Blumenstein. Image Compression Using a Direct Solution Method Based Neural Network, The Tenth Australian Joint Conference on Artificial Intelligence, Perth, Australia, 1997, pp. 114-119.
11. G. Lekutai. Adaptive Self-tuning Neuro Wavelet Network Controllers, Thèse de Doctorat, Blacksburg- Virgina, Mars 1997.
12. R.D. Dony and S. Haykin. Neural network approaches to image compression, Proceedings of the IEEE, V83, N°2, Février, 1995, pp.288-303.
13. A. D'souza Winston and Tim Spracklen. Application of Artificial Neural Networks for real time Data Compression, 8th International Conference On Neural Processing, Shanghai, Chine, 14-18 Novembre 2001.
14. Ch. Bernard, S. Mallat and J-J Slotine. Wavelet Interpolation Networks, International Workshop on CAGD and wavelet methods for Reconstructing Functions, Montecatini, 15-17 Juin 1998.
15. D. Charalampidis. Novel Adaptive Image Compression, Workshop on Information and

Systems Technology, Room 101, TRAC Building, University of New Orleans, 16 Mai 2003.
16. M. J. Nadenau, J. Reichel, and M. Kunt, "Wavelet Based Color Image Compression: Exploiting the Contrast Sensitivity Function",EEE Transactions Image Processing, vol. 12, no.1, 2003, pp. 58-70.
17. K. Ratakonda and N. Ahuja, "Lossless Image Compression with Multiscale Segmentation", IEEE Transactions Image Processing, vol.11, no.11, 2002, pp. 1228-1237.
18. K. H. Talukder and K. Harada, "Haar Wavelet Based Approach for Image Compression and Quality Assessment of Compressed Image", IAENG International Journal of Applied Mathematics, 2007.
19. Bo-Luen Lai and Long-Wen Chang, "Adaptive Data Hiding for Images Based on Haar Discrete Wavelet Transform", Lecture Notes in Computer Science, Springer-Verlag, vol. 4319, 2006, pp. 1085-1093.
20. S. Minasyan, J. Astola and D. Guevorkian, "An Image Compression Scheme Based on Parametric Haar-like Transform", ISCAS 2005. IEEE International Symposium on Circuits and Systems, 2005, pp. 2088-2091.
21. Z. Ye, H. Mohamadian and Y.Ye, "Information Measures for Biometric Identification via 2D Discrete Wavelet Transform", Proceedings of the 3rd Annual IEEE Conference on Automation Science and Engineering, CASE'2007, 2007, pp. 835-840.
22. S. Osowski, R. Waszczuk, P. Bojarczak, "Image compression using feed forward neural networks — Hierarchical approach" Lecture Notes in Computer Science, Book Chapter, Springer-Verlag, vol. 3497, 2006, pp. 1009- 1015.
23. M. Liying and K. Khashayar, "Adaptive Constructive Neural Networks Using Hermite Polynomials for Image Compression", Lecture Notes in Computer Science, Springer-Verlag, vol. 3497, 2005, pp. 713-722.
24. R. Cierniak, "Image Compression Algorithm Based on Soft Computing Techniques", Lecture Notes in Computer Science, Springer-Verlag, vol. 3019, 2004, pp. 609-617.
25. B. Northan, and R.D. Dony, "Image Compression with a multiresolution neural network", Canadian Journal of Electrical and Computer Engineering, Vol. 31, No. 1, 2006, pp. 49-58.
26. S. Veisi and M. Jamzad, "Image Compression with Neural Networks Using Complexity Level of Images", Proceedings of the 5th International Symposium on image and Signal Processing and Analysis, ISPA07, IEEE, 2007, pp. 282-287.
27. I Vilovic, "An Experience in Image Compression Using Neural Networks", 48th International Symposium ELMAR-2006 focused on Multimedia Signal Processing and Communications, IEEE, 2006, pp. 95-98.