A Multiplierless VLSI Architecture of QR Decomposition Based 2D Wiener Filter for 1D/2D Signal Processing With High Accuracy

Anirban Chakraborty
Electronics & Telecommunication Engineering
IIEST, Shibpur
Howrah, India
chakraborty.sab@gmail.com

Ayan Banerjee
Electronics & Telecommunication Engineering
IIEST, Shibpur
Howrah, India
ayan@telecom.becs.ac.in

***Abstract*—Now-a-days real-time signal processing attracts growing interests from the researchers all over the world due to its advantageous nature in solving various hindrances that frequently occur in various significant signal processing applications. Specifically, digital images suffer from noise contamination and blurring effect which poses difficulties in extracting useful information from those images. This necessitates the removal of noises from those digital images as well as de-blurring of such images in real-time. In this article, we have proposed a low area and highly accurate VLSI architecture of 2D Wiener filter which can be applied for any1D/2D real-time signal efficiently. The applicability of inherently highly accurate Wiener filter throttles due to its computational complexities. Our focus in this article is to overcome the barrier by reducing the computational complexity using Toeplitz matrix formation and its QR decomposition. We have proposed an area efficient multiplier-less VLSI architecture for realizing 2D Wiener filter. We have exploited the concept of Givens rotation based QR decomposition and also we have utilized the CORDIC algorithm to achieve high performance of our design. We have also applied our proposed hardware in real-time audio signal and image denoising. The supremacy of our proposed design can be proved from the analysis of pictorial and also numerical results.**

***Keywords*—2D Wiener Filter, VLSI Architecture, QR Decomposition, Autocorrelation, Image De-noising.**

I. Introduction

Modern signal processing finds its applications in such significant areas like satellite image processing, medical image processing that requires real-time operations frequently. Digital images or audio signal are prone to be degraded by impulse or Gaussian noise easily. These kinds of noise contaminations pose severe hindrance to get valuable information accurately from those contaminated digital images or audio signals. Obviously, there are so many methods to get rid of such unwanted noises. [1] There also exists several sophisticated software to take care of such noise contaminations. But all those methods and software are unable to tackle any undesired noise or blurring effect if it is required to operate in real-time as is the case for a number of significant applications. [2] That is why it is required to implement any signal processing algorithms in terms of hardware so that those algorithms can be effectively used in real-time applications. Usage of Adaptive Filters (AF) are one of the most viable solutions for signal denoising and de-blurring as this kind of filters are immune to dynamic situations. [3] Now, among many AFs we have focused on Wiener Filter (WF) in this article. This is because of several advantages of WF over other AFs. The primary advantage of WF is that it is a non-recursive type of filter unlike all other AFs. That is why WF is inherently faster than any other AFs. Moreover, WFs produce accurate output and so it is considered as the optimum solution. [3] That is why, in this article we have focused on utilizing WF in audio signal and also image processing to achieve a fast method that produces results with great accuracy. But, the main problem of using WF in real-time signal processing applications is that it involves extensive mathematical calculations which make it incapable to be used in any real-time applications. This barrier that forces researchers to reluctantly focus on other filters is successfully overcome by us. In this article, we have proposed a method to reduce the cumbersome mathematical calculations of WF and also we have proposed one high speed, area efficient hardware for the same. Novelty of the work is in the fact that we have efficiently reduce the computational load of WF by using some useful mathematical techniques as well as we have been able to judiciously design our method in terms of low area and high speed hardware that can be applied on significant real-time audio signal as well as image processing applications.

The rest of the paper is organized as follows. Section II describes the mathematical description of WF. Section III presents existing hardware implementations of WF till date. We have put down the detailed description of our proposed method and its architecture in Section IV. The results of our proposed VLSI architecture of WF and discussions are given in Section V. The output of some real-time imagery applications using our proposed proto-type WF is also provided in Section V. Finally, we conclude in Section VI.

II. OVERVIEW OF WIENER FILTERING

WFs are usually applied in the frequency domain. Given a degraded image $x(n,m)$ one takes the Discrete Fourier Transform (DFT) to obtain $X(u,v)$ .The original image

978-1-5386-6947-1/18/$31.00 ©2018 IEEE

spectrum is estimated by taking the product of $X(u,v)$with the transfer function of WF, i.e.,$G(u,v)$[2]:

$$S(u,v) = G(u,v)X(u,v) \tag{1}$$

The inverse DFT is then used to obtain the image signal from its spectrum. The transfer function of WF is defined as follows [2]:

$$G(u,v) = \frac{H^*(u,v)P_S(u,v)}{|H(u,v)|^2 P_S(u,v) + P_n(u,v)} \tag{2}$$

Where,

$H(u,v)$ =Fourier transform of Point Spread Function (PSF).

$P_S(u,v)$ =Power spectrum of the signal, obtained by taking the Fourier transform of the signal autocorrelation.

$P_n(u,v)$ =Power spectrum of the noise, obtained by taking the Fourier transform of the noise autocorrelation.

For the case of additive white noise and no blurring, the WF simplifies to [2]:

$$G(u,v) = \frac{P_S(u,v)}{P_S(u,v) + \sigma_n^2} \tag{3}$$

Where, σ_n^2 is the noise variance.

In order to derive the coefficients of the WF, consider the signal $x[n]$fed to the input of WF of order n having filter coefficients $\{a_0, a_1, \dots \dots .., a_N\}$. The output of the filter is denoted by $y[n]$ which is given by the expression [2]:

$$y[n] = \sum_{i=0}^{N} a_i x(n-i) \tag{4}$$

The residual error is denoted by $e[n]$ and calculated as follows.

$$e[n] = y[n] - s[n] \tag{5}$$

Where, $s[n]$ is the desired output.

The WF is designed so as to minimize MMSE criteria which can be mathematically stated as follows:

$$a_i = argmin[E\{e^2(n)\}] \tag{6}$$

Where, $E[.]$ denotes the expectation operator.

In the general case, the coefficients a_i may be complex and may be derived for the case where $x[n]$ and $s[n]$are complex as well. With a complex signal, the matrix to be solved is a Hermitian Toeplitz matrix [2], rather than a symmetric Toeplitz matrix. For simplicity, we consider only the case where all these quantities are real. The mean square error (MSE) may be rewritten as [3]:

$$\begin{aligned} E\left[e^2[n]\right] &= E[(y[n]-s[n])^2] \\ &= E\left[\left(\sum_{i=0}^{N} a_i x[n-i]\right)^2\right] + E\left[s^2[n]\right] \\ &\qquad - 2E\left[\sum_{i=0}^{N} a_i x[n-i]\, s[n]\right] \end{aligned} \tag{7}$$

To find the vector $\{a_0, a_1, \dots \dots .., a_N\}$ which minimizes the expression above, calculate its derivative with respect to each a_i

$$\begin{aligned} \frac{\delta}{\delta a_i}\left\{E\left[e^2[n]\right]\right\} &= \frac{\delta}{\delta a_i}\left\{E[(\textstyle\sum_{i=0}^{N} a_i x[n-i])^2] + E\left[s^2[n]\right] - 2E\left[\textstyle\sum_{i=0}^{N} a_i x[n-i]\, s[n]\right]\right\} \\ &= 2\,(\textstyle\sum_{j=0}^{N} E[x[n-j]x[n-i]a_j]) - 2E[x[n-i]s[n]] \end{aligned} \tag{8}$$

Assuming that $x[n]$and s[n] are each stationary and jointly stationary, the sequence $R_{xx}[m]$ and $R_{xs}[m]$ known respectively as the autocorrelation of $x[n]$ and the cross-correlation between $x[n]$ and $s[n]$ can be defined as follows:

$$R_{xx}(m) = E[x[n]x[n+m]] \tag{9}$$

$$R_{xs}(m) = E[x[n]s[n+m]] \tag{10}$$

The derivative of the MSE may, therefore, be rewritten as [2]:

$$\frac{\delta}{\delta a_i}\left\{E\left[e^2[n]\right]\right\} = 2\left(\sum_{j=0}^{N} R_{xx}[j-i]a_j\right) - 2R_{xs}(i) \tag{11}$$

Letting the derivative be equal to zero results in [1]:

$$\sum_{j=0}^{N} R_{xx}[j-i]a_j = R_{xs}(i) \tag{12}$$

which can be rewritten in matrix form [1]:

$$\begin{bmatrix} R_{xx}[0] & \cdots & R_{xx}[n] \\ \vdots & \ddots & \vdots \\ R_{xx}[n] & \cdots & R_{xx}[0] \end{bmatrix} \begin{bmatrix} a_0 \\ \vdots \\ a_N \end{bmatrix} = \begin{bmatrix} R_{xs}[0] \\ \vdots \\ R_{xs}[N] \end{bmatrix} \tag{13}$$

These equations are known as the Wiener-Hopf equations [3]. The matrix T appearing in the equation is a symmetric Toeplitz matrix [2]. Under suitable conditions on$R$, these matrices are known to be positive definite and therefore non-singular yielding a unique solution to the determination of the WF coefficient vector [3],

$$a = T^{-1}v \tag{14}$$

III. EXISTING WF ARCHITECTURES

As WF is computationally complex and thus conventionally unsuitable for implementing in terms of hardware, researchers, till date, were reluctant to utilize WF in case of real-time signal processing applications. That is why; there are very few VLSI architectures of WF that are proposed so far. Notable and recent existing architectures of WF are briefly described here in the following paragraphs.

The work of A. Khireddine et.al [4] depicts that 2D WF produces the best-restored output. The output from WF also bears the least error. They performed the analysis of the performance in an efficient way. But, they didn't try to overcome the computational load of the WF. That is why, though their WF can be extended to a higher dimension, it can never be implemented in terms of hardware. So, their algorithm of WF is of no use in case of real-time signal processing applications

H. Y. Jheng et.al [5] proposed VLSI architecture of 1D WF. They have established their superiority by means of the quality of the restored audio signal and the amount of noise removed from the signal. But, their architecture is not area efficient as it involves a number of multipliers including matrix multiplier also.

IV. PROPOSED ARCHITECTURE

As we know WF is an optimum solution that produces output with great accuracy. So, WF is the best solution for any signal processing problem where accuracy is of prime importance but, as discussed before, WF is computationally complex which restricts researchers to implement it in hardware for real-time signal processing. In this article, our focus is to fill up this gap by proposing one high speed, area efficient architecture of WF. From the equations of Section II, we know that computation of Wiener coefficients composed of certain steps. Those steps are depicted in Fig. 1 below.

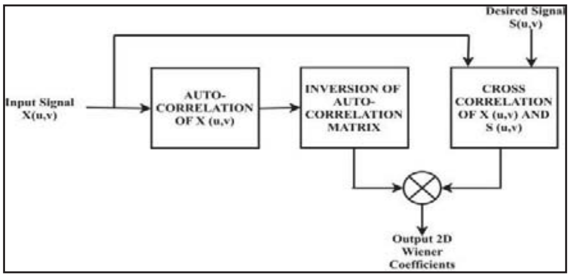
Desired Signal
S(u,v)
Input Signal
X(u,v)
AUTO-CORRELATION OF X (u,v)
INVERSION OF AUTO-CORRELATION MATRIX
CROSS CORRELATION OF X (u,v) AND S (u,v)
Output 2D Wiener Coefficients

Fig. 1. General steps of computation of Wiener coefficients

From Fig. 1, we can see that the calculation for generating Wiener coefficients comprises auto-correlation of 2D input signal $X(u,v)$ followed by the inversion of that auto-correlation matrix. The cross-correlation of the input signal $X(u,v)$ and the desired signal $S(u,v)$ is then multiplied with the inverted auto-correlation matrix to get the final Wiener coefficients. So, extensive computational load based operations like 2D auto-correlation and cross correlation, matrix inversion and matrix multiplication are present in the process of the Wiener coefficient calculation.

In this Section, we present our proposed method by which we have been able to reduce the computational load of the above mentioned operations significantly. Block diagram of our proposed method is shown in Fig. 2.

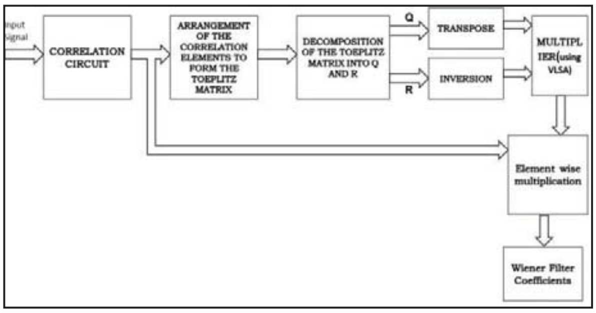
Input Signal
CORRELATION CIRCUIT
ARRANGEMENT OF THE CORRELATION ELEMENTS TO FORM THE TOEPLITZ MATRIX
DECOMPOSITION OF THE TOEPLITZ MATRIX INTO Q AND R
Q
R
TRANSPOSE
INVERSION
MULTIPLIER(using VLSA)
Element wise multiplication
Wiener Filter Coefficients

Fig. 2: Our proposed block diagram for 2D WF

From this block diagram, we can realize that the time consuming and computationally inefficient operations like matrix inversion and cross correlation are judiciously avoided in our proposed method by means of QR decomposition and the transpose of the matrix which reduces the complexity of the process to a great extent. Multiplications throughout the whole process are abolished by realizing multiplication in terms of shift and add method

A. Auto-correlation circuit

The first step in our block diagram is auto-correlation of input signal. We assume that the signal is an N point signal so we have to take N special type of shifters as shown in Fig. 3, that shift the sequence by one bit. Another input is m that is taken to compare how many shifts are required before the correlation is computed. The SC block shifts a sequence $x[n]$ by one bit to produce the sequence $x[n-1]$. As per the given value of m that value of the Comparator is activated and the buffer turns on allowing signal $x[n-m]$ to pass through and then they are multiplied with that value of the signal at that particular time instant. This process is fed into a serial adder untill the final result is obtained. The terminating condition is when all the samples are exhausted. The signal $x[n]$ is a time varying signal so with every passing instant the value of $x[n]$ changes at the input. It is this change that translates to yield a different product each time $x[n]x[n-l]$. This is summed over all samples to yield the value of auto correlation.

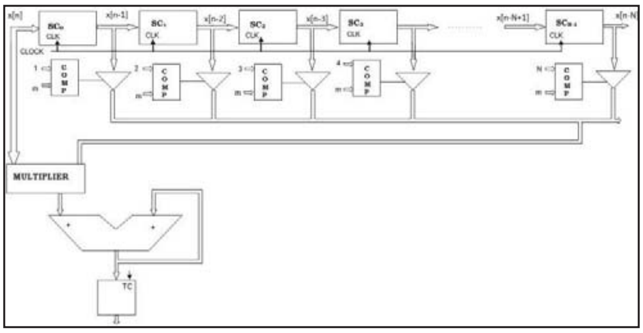

Fig. 3: Internal architectures of auto-correlation block.

B. Matrix inversion Circuit

The matrix inversion circuit is designed by us in a way which would reduce the area and computational load to a great extent. QR decomposition is performed to the matrix to be inverted. QR decomposition is an elementary operation, which decomposes a matrix into an orthogonal and a triangular matrix. QR decomposition of any real square matrix A is a decomposition of A as $A = QR$, where Q is an orthogonal matrix ($Q^TQ = I$) and R is an upper triangular matrix. The inversion of A is found out as (15)-(17) below in our case:

$$A^{-1} = (QR)^{-1} \quad (15)$$

$$A^{-1} = (R)^{-1}(Q)^{-1} \quad (16)$$

$$A^{-1} = (R)^{-1}(Q)^{T} \quad (17)$$

Among various methods of QR decomposition we have judiciously chosen Givens rotation based QR decomposition so that we can utilize hardware efficient CORDIC algorithm [6] to realize the rotation. Givens Rotation process utilizes a cycle of rotation whose function is to nullify an element in the sub-diagonal of the matrix forming the QR matrix. Q matrix is obtained by concatenating all the Givens Rotation. R is to be found from three rotations where each element is obtained from each rotation. All the rotations mentioned above are realized using the CORDIC algorithm [6] in our case. The internal architecture for QR decomposition is shown in Fig. 4 below where,G_1, G_2 and G_3 are the cyclic Givens rotation realized using CORDIC [6].

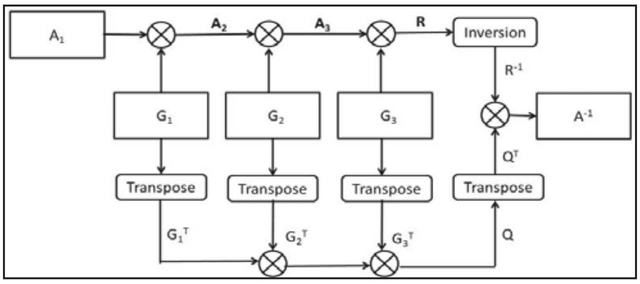

Fig. 4. The internal architecture of the QR decomposition.

After the decomposition of the matrix, we have concentrated to invert the upper triangular matrix, i.e.,R. We have designed an efficient hardware for the inversion of a generalized $N\ X\ N$ upper triangular matrix R . Let R is defined as below:

$$R = \begin{bmatrix} r_{1,1} & \cdots & r_{1,n} \\ \vdots & \ddots & \vdots \\ 0 & \cdots & r_{n,n} \end{bmatrix} \quad (18)$$

Let, B is the inverted matrix. So,

$$B = (R)^{-1} \quad (19)$$

We observe that the inverse of the upper triangular matrix is also an upper triangular matrix with the diagonal elements reciprocal of the diagonal elements of the original matrix. The inverse of the other elements is calculated using our proposed method recursively. The following circuit in Fig. 5, computes the reciprocal of the diagonal elements of the upper triangular matrix by accessing only the diagonal elements of the matrix (the input matrix is given by A and the resultant matrix with only the inverse of the diagonal elements is stored in matrix B.

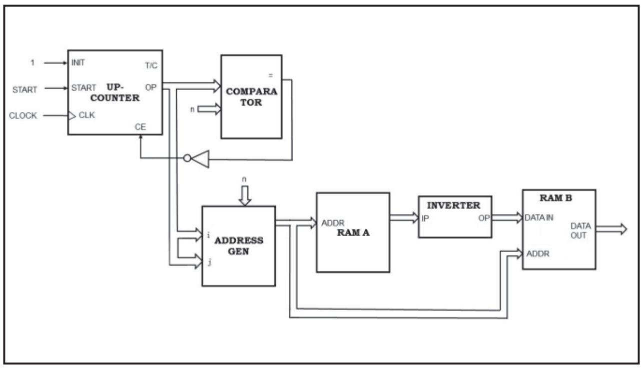

Fig. 5: The internal architecture of inversion of diagonal elements of R

As in the upper triangular matrix the $r_{i,j} = 0$ for $i < j$; there would no need for storing them as zeroes individually in the RAM, instead we could just omit the zeroes and find the position in the RAM corresponding to inputs (i, j) that is $r_{i,j}$would be given and a corresponding location in the RAM would be obtained in our mechanism where zeroes are not stored, the address in the RAM for $r_{i,j}$would be equal to $n(i-1) + j - \frac{i(i-1)}{2} - 1$

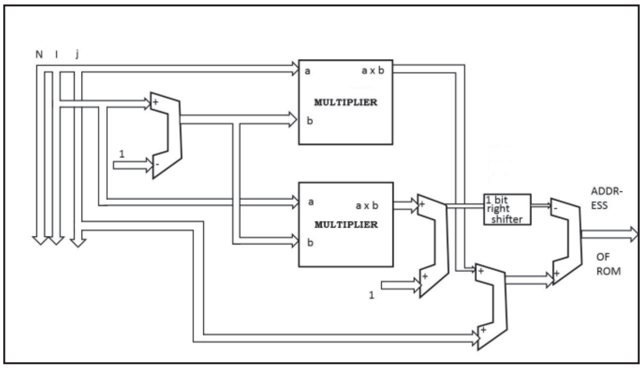

Fig. 6: Circuit for RAM addresses generation

The following circuit in Fig. 7, computes the inverse of the non-diagonal elements of the upper triangular matrix by accessing the non-diagonal elements of the matrix (the input matrix is given by A and the resultant matrix with only the inverse of the diagonal elements is stored in matrix B. This computes the inverse using the algorithm demonstrated previously. The Address generation block ADDR is already shown in Fig.6.

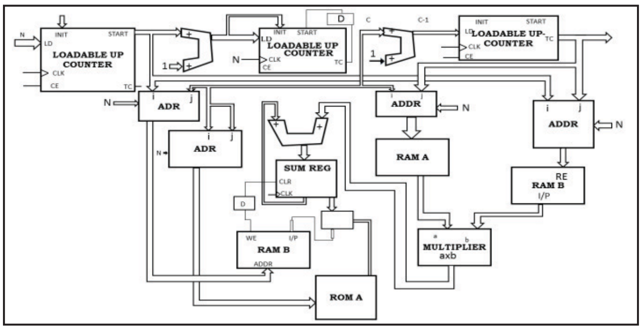

Fig. 7: The internal architecture of inversion of non-diagonal elements of R.

V. RESULTS AND DISCUSSIONS

We have simulated the proposed hardware of the 2D WF in Xilinx 14.7. The simulated waveforms are analyzed to verify the quality of the proposed prototype hardware. All the input signal values are taken as 24 bit fixed point binary numbers. After post placement simulation, we have implemented the design onto our Spartan-6 series XC6SLX45T FPGA.

After implementation, we have established high-speed Ethernet connection between our FPGA and Personal Computer (PC). For that, we have manually configured the in-built interfacing blocks of the FPGA. Having established the connection, we have used MATLAB software in our host PC to send 1D and 2D input noisy and blurred signals to the FPGA. The prototype WF inside FPGA is then used to restore those input signals and the output from the FPGA is sent back to the host PC for display. The input noisy signals and the restored signals are shown below both for 1D and 2D case.

We have analysed the performance of our proposed hardware in two real-time scenarios for 1D and 2D.

- Addition of bird chirping to the human voice and then subsequently denoising the human voice and just obtaining the human voice from the noisy samples. This case is somewhat different from the AWGN case because it has a known PSD (Power Spectral Density) but bird chatter is completely random with frequent pauses and hence the challenge of the WF is to identify the human voice from the bird chatter.
- We have added AWGN to the popular 'Lena' image. Then we have used our 2D WF to restore the image.

In all these scenarios, the performance of our proposed hardware is presented in the pictorial form in Fig. 8(a) to (c), Fig. 9(a) to (b) and Fig. 10 (a) to (c).

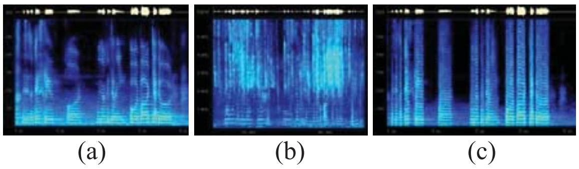

Fig. 8: (a) Original audio signal; (b) Audio signal mixed with AWGN; (c) Restored audio signal by WF

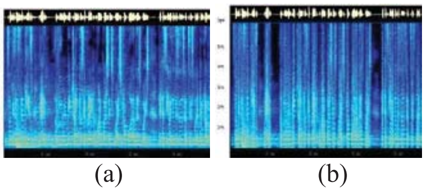

Fig. 9: (a) Original 1D voice mixed with bird chatter; (b) Restored voice

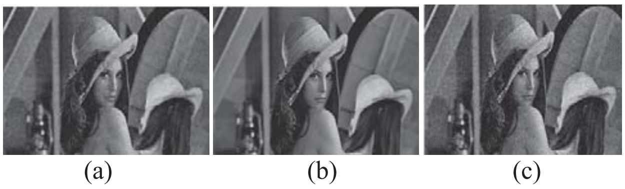

Fig. 10: (a) Original 2D image 'Lena'; (b) Corrupted image; (c) Restored image by WF

After applying our proposed proto-type 2D WF in case of these three real-time applications, we have moved on to analyze the performance of our proposed WF numerically in terms of hardware requirements and Signal to Noise Ratio (SNR). We have compared the performance of our work with that of the work of [4, 5]. Analytically we have calculated the hardware requirement of our architecture which is presented in TABLE I below. TABLE II depicts the comparison results of our proposed work with [5] in terms of different performance parameters. TABLE III shows the supremacy of our proposed work with respect to the work of [4] in terms of SNR and computational complexity.

TABLE I. HARDWARE REQUIREMENT OF PROPOSED WORK

Components	Number
Multiplier	0
Adder	30
Register	23
Loadable Up Counter	4
Inverter	1
Comparator	N
Right Shifter	1
RAM	4

TABLE II. NUMERICAL COMPARISON OF WF QUALITY WITH [5]

Works	SNR Achieved	Peak Sampling Rate (kHz)
[5]	30.27 dB	88.62
Proposed	50.28 dB	166. 68

TABLE III. NUMERICAL COMPARISON OF WF QUALITY WITH [4]

Works	SNR Achieved	Computational Complexity For Denoising M X N Image
[4]	30 dB	$O(MNlogN)$
Proposed	57 dB	$O(MN)$

From the above tables, it is clear that our proposed hardware efficient architecture is capable of producing high quality output. The quality of the output signal from our proposed design is calculated in terms of SNR and it is compared with the work of [4, 5] to prove that our architecture is performing the best.

To prove the supremacy of our proposed design in terms of area requirement, we have presented the resource utilization report of our FPGA. We have also compared our resource utilization with the work of [5] which is presented in TABLE IV.

TABLE IV. COMPARISON OF FPGA RESOURCE UTILISATION

Resource Name	**Resource Used in [5]**	**Resource Used in Proposed Work**
No. of Slice Register	19,346	8,245
No. of LUTs	22,126	3,768
No. of DSP48E	36	15

VI. CONCLUSION

In this article, we have proposed a high speed area efficient VLSI architecture of 2D WF and thereby, we have overcome the hindrance posed by the extensive computational load of any WF. By exploiting the concept of Givens rotation based QR decomposition and by judiciously utilizing the hardware efficient CORDIC algorithm, we have been able to achieve the proposed 2D WF. Our proposed design is implemented in Spartan 6 series FPGA. We have applied our VLSI architecture in case of different real-time applications. The outputs of our design for such applications are presented in the article to visually realize the quality of our design. Numerical results of comparison with other similar kinds of works are also given in this article to prove that our proposed design is the best. In future, our research work will be focused on to extend this design in real-time multi-dimensional signal processing including real-time medical and satellite image processing where accuracy of output will be of prime interest.

REFERENCES

[1] S. Haykin, *Adaptive Filter Theory*, Prentice Hall, Upper Saddle River, New Jersey, 1996.

[2] P. S. R. Diniz, *Adaptive Filtering: Algorithms and Practical Implementation*, 4th edition, New York, NY, USA: Springer, 2013.

[3] A. H. Sayed, *Fundamentals of Adaptive Filtering*, John Wiley & Sons, 2003.K. Elissa, “Title of paper if known,” unpublished.

[4] Khireddine, A., Benmahammed, K., & Puech, W. (2007). Digital image restoration by Wiener filter in 2D case. *Advances in Engineering Software, 38*, 513-516.

[5] Jheng, H., Chen, Y., Ruan, S., & Qi, Z. (2011). FPGA implementation of high sampling rate in-car non-stationary noise cancellation based on adaptive Wiener filter. *2011 IEEE/IFIP 19th International Conference on VLSI and System-on-Chip*, 114-1.

[6] Meher, P.K., Valls-Coquillat, J., Juang, T., Sridharan, K., &Maharatna, K. (2009). 50 Years of CORDIC: Algorithms, Architectures, and Applications. *IEEE Transactions on Circuits and Systems I: Regular Papers, 56*, 1893-1907.