CHAPTER V

Noise Reduction by using Modified Wiener Filter in Digital Hearing Aid for Speech Signal Enhancement

5.1 INTRODUCTION

To improve the speech intelligibility, hearing impaired people widely use quality of life Digital hearing aids. The performance of the hearing aid is usually degraded due to feedback, acoustic where other problems are generated. The propagation of sound from the loudspeaker to the microphone produces this phenomenon. This causes high-frequency oscillation and instability which can be perceived by hearing-impaired people if its level increases their thresholds of hearing. The maximum gain will be limited by these effects which the hearing aid can perform, and sound quality can decrease when the gain is close to limit. The acoustic feedback can be reduced several methods based on adaptive algorithms which have been used in [93] for the reduction of feedback. For speech noise reduction Number of techniques can be used like Acoustic feedback reduction based on Finite Impulse Response (FIR) and Infinite Impulse Response (IIR) Adaptive Filters in Digital hearing aids [95], Non-local diffusion filters [94], and Noise reduction Wiener filter [96]. Using the filters, the noise can be suppressed in the above techniques such as adaptive filters, Wiener filter. Using the adaptive filters the Least Mean Squares (LMS), Recursive least squares (RLS), Kalman filters can be implemented. Adaptive filters are optimal in that they minimise the mean squared estimation error [97] and in real-time, the computation was made. The disadvantages of adaptive filters are assumed the dynamic process is linear, provide only point estimate, can only handle processes with the noise of additive and unimodal.

For noise, the technique of spectral subtraction is a known method [98]. From the domain of time into the frequency domain the noisy speech signal is first transformed by means of the Fast Fourier Transform FFT, the noise spectrum is then determined in the speech pauses and from the frequency spectrum of the noisy speech signal subtracted before the noisy speech signal is reconverted from the domain of frequency into the time domain by means of the Inverse Fast Fourier Transformation IFFT. On the accuracy of determination, the result was depended on the noise spectrum. In the
case of stationary noise good results are achieved in practice that noises are not stationary, and the results, achievable are unsatisfactory. On the other hand, the properties of the signal are explored by the Weiner filter. The output error can be controlled and also straightforward to design. Among all techniques, the optimal Wiener filter [96], [98] is one of most fundamental approaches to noise reduction which has been delineated in different forms and adopted in various applications. The spectral property knowledge of the original signal and for designing the Wiener filter, the noise should be known. The availability of certain parameters of statistical function like mean and original speech signal correlation and unwanted additive noise Wiener filter is designed in the Figure 5.1. The noise signal effect can be reduced according to some statistical criterion the noisy speech signal is fed as input to Wiener filter. By mean square error reduction, the unwanted noise signal can be avoided by using the Wiener filter [104].

![Figure 5.1: General Form of Wiener Filter](image)

The Wiener filter may cause some detrimental effects to the speech signal between noise reduction and speech distortion to show an inherent relationship, few efforts have been reported. From the disadvantage of the noise reduction method using a Wiener filter, the object of altering the noise estimation using the Wiener filter and transforming of rules with the noisy speech signals from the time domain into the frequency domain and an adoption is allowed to the nonlinear transmission behaviour of the human ear.

The unwanted noise signal can be reduced from the original speech signal by the effective noise degradation architecture was proposed. The overall block diagram for noise degradation is shown in Figure 5.2. The continuous time domain signal is segmented into the overlapping chunks called frames, and by a window function, the
frames are multiplied for removing the spectral artifacts for the performance of frequency domain processing of the speech signal in the system. By using the processor of FFT, the signal in time domain is converted into a frequency domain, and the noise signal gets reduced from the speech signal the Wiener filter is used. By using Inverse Fast Fourier Transform processor, the output signal from Wiener filter is converted into the time domain, and then multiplied with the same window function and to create a continuous output signal the frames are then overlapped [100][106].

![Overall Block Diagram for Noise Degradation](image)

**Figure 5.2: Overall Block Diagram for Noise Degradation**

The system becomes more efficient for the conversion of the domain instead of using two separate processors for FFT and IFFT. The proposed design made it within a single processor. The FFT is defined over data with complex, but the input is real in many applications. The algorithm of real-valued FFT takes advantage of the symmetry properties of the FFT and over the algorithm of the complex has a speed advantage of the same length. The throughput and area are higher and lower latency in the Real-
Valued Fast Fourier Transform. The efficiency of Computation is low for the implementation. Hence the implementation of RFFT is preferred rather than the processor of FFT for the design since the signal of input is a real-valued signal. The implementation of the modified Wiener filter in the proposed design incorporates a more efficient power spectrum and the technique of energy analysis. To contribute the power reduction in the proposed design Low power floating point adders and multipliers are adopted. Also in VLSI floating point division [99] is a slow process and for the computation, more clock cycles are taken, and to speed up the computations and the operations of the floating point, the division is replaced with an operation of reciprocal and a multiplier. The architecture is pipelined so that the system performance is increased.

During the process of communication, the Speech signals are usually affected by noises. Wiener filter is adapted for suppressing the noise signal, and the speech signal is combined. In the suppression of noise and enhancement, Weiner filter plays an important role in the estimation of a relation between the power spectrums of the noise affected signal of speech and the noise signal. Consumption of Power and the requirement of hardware are the important problems in adapting the Weiner filter for the major process of communication systems. The proposed implementation is an efficient Wiener filter, and along with a processor of Real-valued FFT/IFFT (RFFT/RIFFT), its application for noise is suppressed. The process of pipelined is adopted for raising the system performance. The proposed Wiener filter is designed in such a way to eliminate the problems of iteration in the conventional Wiener filter. The operation of division is replaced by an efficient inverse and multiplication operation in proposed design. For matrix inversion, a modified architecture with low computation complexity is implemented. The complete design computation is based on IEEE-754 standard single precision floating point numbers. The Wiener filter and system design were implemented and designed in Field Programmable gate Array (FPGA) platform and simulated to validate the results in Xilinx Integrated Synthesis Environment (ISE) tools. An efficient power and area reduction is obtained by adapting the proposed method for speech signal noise degradation.

Section 5.2 includes the approach and with the proposed method which includes the proposed noise degradation system with blocks of RFFT/RIFFT processor, the design of Wiener filter, Subtraction of matrix, matrix multiplier, Power spectral density,
matrix-vector multiplier and inversion of a matrix using given rotation. The results were discussed in section 5.3, and the conclusion of the work is made in section 5.4.

5.2 PROPOSED METHOD

For suppressing the noise signal that is combined with the speech signal Wiener filter is adapted [96, 98, and 101]. Weiner filter plays an important role in noise suppression and enhancement by estimating the relation between autocorrelation of noise affected speech signal and the noise signal. Five main processing units are included in the architecture of Wiener filter which are Power Spectral Density (PSD), Matrix Inverter, Matrix Subtractor, Matrix Multiplier and Matrix-Vector Multiplier. The special case of Cross-Correlation is Auto-Correlation which computes a signal with itself [109]. This provides the relationship between the noise and noise free signal. For the researchers, the Matrix inversion is an important challenge while designing any system of signal processing as it consumes a vast area and power [110].

Considering the problem of the matrix inversion in the design of Wiener filter the noise degradation system for noisy speech signals in Digital Hearing Aid is implemented which incorporates a powerful and efficient hardware Wiener filter. The pre-processing and post processing of the Wiener filter for the conversion of the input speech and noise signal in the domain of time is transformed to the frequency domain and done by the pre-designed real-valued FFT processor by contributing it to modify as a processor of RFFT/RIFFT [102, 103, and 111]. The main contribution is a low power and hardware efficient matrix inversion module design has used in the decomposition of QR [112] with given rotation. As the process of real-valued FFT/IFFT is done, by using a single processor and also by the utilisation of a modified analytic method and design which is an efficient system of speech signal enhancement. The block diagram of the proposed system is shown in Figure 5.3.

The blocks of the proposed noise degradation system are:

1) Efficient Pipelined Architecture for Real-Valued FFT/IFFT Processor and

2) A Modified Wiener Filter.

5.2.1 Real Valued FFT/IFFT Processor

For noise degradation, the fundamental process in the proposed technique is the transformation of input signals in time domain to the frequency domain. Since speech and noise signals are real-valued signals, the conventional FFT architecture for the
conversion of a domain can be replaced with modified low power pipelined architecture so as to make the complete hardware architecture efficient in terms of area and power consumption. The main difference between the FFT and IFFT is used for FFT/IFFT processor design.

Figure 5.3: Overall Block Diagram of the Proposed System

The basic FFT Equation and its inverse of FFT are shown in equation (5.1) and (5.2).

\[
X(\omega) = \sum_{\phi=0}^{M-1} x(\phi) W_{M}^{\phi\omega}, \quad \omega = 0,1,........M - 1
\]

(5.1)

Where \( W_{M} = e^{-j(2\pi/M)} \) is the twiddle factor
On the other hand, the Inverse FFT Equation is

\[ x[\phi] = \frac{1}{M} \sum_{\omega=0}^{M-1} X[\omega]W^{-\phi\omega} \]  

(5.2)

Where, \( \phi = 0, 1, \ldots, M-1 \)

Equation (5.2) can be written as complex conjugate form is shown,

\[ x[\phi] = \frac{1}{M} \sum_{\omega=0}^{M-1} [X[\omega]^*W^{-\phi\omega}]^* \]

(5.3)

By using the relation between the Equation (5.2) and (5.3), the basic structure of our FFT/IFFT processor as shown in Figure 5.4.

![Figure 5.4: The Basic Structure of Our FFT/IFFT Processor](image)

The architecture of RFFT architecture includes four stages and each stage includes the butterfly unit. The block schematic for 2-parallel pipelined architecture for 16 points radix2 RFFT is shown in Figure 5.5.

![Figure 5.5: Block Schematic for 2-Parallel Pipelined Architecture for 16 Point Radix 2 RFFT](image)

**Stage 1**

The pair of real samples \( x(\phi) \) and \( x(\phi + M/2) \) process can be performed by the butterfly unit in the stage1. The 2:1 multiplexer consists of one selector line \( S \) in the
butterfly unit. The input value computation is started by the butterfly unit if inputs are real, then the value set to the selector line is 1. The selection line value is set to 0 when the input is complex, and then the multiplexer just passes the input without computation.

**Stage 2**

The stage 2 consists of three parts shuffling units, the butterfly unit and twiddle factor $W^\phi$. The Butterfly unit works as explained in the stage 1. To transform the order of the data the unit of shuffling is provided which required from the stage 1 to stage 2, also 2:1 multiplexer was present and at the input and output of the multiplexer two delay elements are included. The architecture of $W^\phi$ module was shown in figure 5.6. Four twiddle factors are included in the stages as $W^0, W^1, W^2$ and $W^3$ with the values of real and imaginary values and tabulated in the in Table 5.1.

![Figure 5.6: Architecture of $W^\phi$ Module](image)

<table>
<thead>
<tr>
<th>Twiddle factor($W^k$)</th>
<th>Real</th>
<th>Imaginary</th>
</tr>
</thead>
<tbody>
<tr>
<td>$W^0$</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$W^1$</td>
<td>0.9239</td>
<td>0.3827</td>
</tr>
<tr>
<td>$W^2$</td>
<td>0.7071</td>
<td>0.7071</td>
</tr>
<tr>
<td>$W^3$</td>
<td>0.3827</td>
<td>0.9239</td>
</tr>
</tbody>
</table>
From the Table 5.1 the selector line S set to 0 and the value of $W^0$ is 1, without any complex multiplication the input passes to the output. The selector line set to 1 for twiddle factors $W^1$ and $W^3$ the multiplexer is allowed for complex multiplication. Canonical –Signed-Digit (CSD) is introduced for the number of additions and shifting reduction. In the calculation of Canonical –Signed-Digit (CSD) the twiddle factor coefficient have to convert from binary to Canonical-Signed-Digit as shown in table 5.2. The steps for conversion of binary to CSD is given as,

**Step 1**: In the sequence of binary the consecutive numbers of 1's are checked.

**Step 2**: Replace the ‘0’ before the first ‘1’ in the sequence with ‘+’ or ‘1’.

**Step 3**: Replace the last ‘1’ in the sequence with ‘-’.

**Table 5.2: Twiddle Factor Coefficients for M=16**

<table>
<thead>
<tr>
<th>Decimal</th>
<th>Binary</th>
<th>CSD</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.9239</td>
<td>0011111101101100</td>
<td>0100000-10-10-00</td>
</tr>
<tr>
<td>0.3827</td>
<td>0011111101100011</td>
<td>010000-10-00010-</td>
</tr>
<tr>
<td>0.7071</td>
<td>0011111100110101</td>
<td>010000-010-0101</td>
</tr>
</tbody>
</table>

For the value of real and imaginary of $W^2$ are similar hence the modified shift and the module of add/subtract can be used.

**Stage 3**

At stage 3, the order of the data is transformed into the shuffling unit 1 which required from stage2 to stage3. Computational samples are shuffled by the shuffling unit 2 also from the butterfly unit, the computed samples get shuffled. In the case of shuffling unit 2, for 2l clock cycles and rest for the clock cycles the initial selector signal is ‘0’ and the operation is similar to the shuffling unit 1. In the block of twiddle factor $W^2$ in the stage3 is used and hence same as before stage 2 is performed.

**Stage 4**

At stage 4 the shuffling unit transforms the samples from the stage 3 to stage 4, and then the samples are computed by the butterfly unit and get the output sample $X(\omega)$.  

93
5.2.2 Wiener Filter Design

For designing an efficient Wiener filter some assumptions was made [105][108]. First, the noise affected signal of input speech is single channel (from one source) and the speech signal and the noise are uncorrelated. If the noise affecting speech signal is additive then the Equation 5.5 is shown below,

\[ x(t) = s(t) + a(t) \]  \hspace{2cm} (5.5)

Where, \( x(t) \) - Noisy speech signal in time domain.
\( s(t) \) - Original speech signal in time domain.
\( a(t) \) - Additive noise in the time domain.

In the input signal to detect noise or interference is very difficult by merely observing the samples of the time domain. By mapping the signals in the frequency domain the detection and analysis of such signals become easy. So the conversion is necessary from time domain noise affected the signal to the frequency domain. By taking real valued Fast Fourier Transform is shown in the Equation (5.6).

\[ X(f) = S(f) + A(f) \]  \hspace{2cm} (5.6)

From the noisy signal, the original speech signal \( S(f) \) can be extracted by multiplying the noisy speech signal \( X(f) \) with the function of Wiener filter \( W(f) \).

\[ S(f) = W(f)X(f) \]  \hspace{2cm} (5.7)

In Equation (5.7) in frequency domain \( W(f) \) represent the Wiener filter and can be estimated as,

\[ W(f) = \frac{|S(f)|^2}{|S(f)|^2 + |A(f)|^2} \]  \hspace{2cm} (5.8)

Where, \( |S(f)|^2 \) is the original speech signal power spectral density
\( |A(f)|^2 \) is the noisy speech signal power spectral density

There is no idea about original speech and noise spectrum, since device with input is noisy speech signal and noisy signal, from the noisy speech signal and the estimated noise signal the Wiener filter is approximated as,

\[ W(f) \approx \frac{|X(f)|^2 - |N(f)|^2}{|X(f)|^2} \]  \hspace{2cm} (5.9)

Where, \( |X(f)|^2 \) is the input noisy speech signal Power Spectral density.
\[ |N(f)|^2 \] is the Power Spectral density of input noisy signal.

By applying Inverse Fast Fourier Transform the original signal \( s(t) \) can be written as

\[
s(t) = \text{RIFFT}\{S(f)\}
\]

(5.10)

For Speech Signal Enhancement to implement an efficient system of de-noising can be realised by the Equations from (5.5) to (5.10) with the units of optimal computation where the various criteria like low power, less area consumption, and high throughput were adapted. The complete architecture of the proposed noise degradation system was shown in Figure 5.7. The proposed noise degradation system consists of RFFT, power spectral density, Matrix inverter, Matrix subtractor, Matrix multiplier, Matrix-vector multiplier and Real valued Inverse Fast Fourier Transform.

![Figure 5.7: The Proposed Noise Degradation System](image)

### 5.2.3 Matrix Subtraction

For the matrix addition and subtraction, the same algorithm is used. Two input matrices are added together or subtracted from each other is the main idea similarly addressed. In the output matrix of the same address cell, the result was placed. The A, B, and C with three 4x4 matrices, and pseudo code is followed which describes the matrix subtraction. To describe addition by replacing the “-” sign with a “+” sign the same pseudo code can be used.

```plaintext
for i = 1: 4
  for j = 1: 4
    C[i, j] = A[i, j] - B[i, j];
  end for;
end for;
```
Figure 5.8: Block Architecture of Matrix Subtraction

From the implementation of hardware values from two RAM modules will be passed to the module of adder or subtractor is shown in Figure 5.8. The adder/subtractor result will then be written to the third module of RAM. It will be used to address the three modules of RAM while the registers will delay the signal of the address to the output RAM. The delay caused by the logic of adder/subtractor will compensate by the register. To ensure that the result is written to the proper address this is important.
5.2.4 Matrix Multiplier

Basically, the first the matrix every row must be multiplied by every column of the second matrix. For a row-column pair, by a similarly addressed element of the column, every element of the row gets multiplied. One element of the resulting matrix forms by the sum of the four products. The pseudo code described in the following process:

```plaintext
for k = 1:4
    for j = 1:4
        for i = 1:4
            C[k, j] = C[k, j] + A[j, i]*B[i, j];
        end for;
    end for;
end for;
```

From the hardware implementation of the matrix multiplier, the A and B counters will control the addressing of matrices A and B respectively. Counter A will increment at one-fourth the frequency of counter B. Counter C is a two-bit counter that selects one of the four temporary RAM modules for writing. From the decoder, each of the output bits is multiplexed with a low logic and transmits to write enable of one of the temporary memories.

If write enabled, four temporary memories written are determined by the decoder. The behaviour of the decoder is described in the following Truth Table 5.3. The input of the decoder is the two-bit counter. There are two addressing modes in Counter D, depending on the state of read/write of the temporary memory. In the write mode, the counter incremented for every four clock cycles to ensure that each module of temporary memory is written to before a change of address. In read mode, counter increments for every clock cycle. This scheme of addressing is used to read from the volatile memories, and to the final result, memory writing takes place.

<table>
<thead>
<tr>
<th>Counter</th>
<th>Decoder</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>0001</td>
</tr>
<tr>
<td>01</td>
<td>0010</td>
</tr>
<tr>
<td>10</td>
<td>0100</td>
</tr>
<tr>
<td>11</td>
<td>1000</td>
</tr>
</tbody>
</table>

Table 5.3: Truth Table of Decoder
To compensate for the combinational logic delay of the adders the address bus controlling the RAM of output and must be pipelined. To add all the products, two stages of adders are used, and the element of the result matrix is formed. Two adders are used in the first stage, and each adds two of four products. The second stage consists of one adder where the outputs of the previous adders are added together. The third adder output is an element of the product of the final matrix and is written to the output RAM. The above operation is shown in Figure 5.9.
5.4.5 Power Spectral Density

The PSD computation from the block level architecture the power spectral density of the input sequence can be calculated, using the estimator of N-point FFT, window filter, Absolute Square Multiple Accumulator Circuits (ASMAC) is shown in Figure (5.10)

![Figure 5.10: Block Architecture for Power Spectral Density](image)

The input signal is divided into N/2 samples from the block diagram. To each N/2 sample, the FFT is applied. To calculate the subset of the dataset the block of Windowing is used to avoid the complexity of the full calculation of the dataset. The functions of the ASMAC circuit are to periodograms computation and average them over L segments. The address decoder and multiplexer are controlled by the control block to accumulate the outputs of periodograms over different sectors correctly. The power spectral density of the input signal is represented as in the Equation (5.11).

\[
\text{PSD} = \frac{1}{K} \sum_{k=1}^{K} |X_k|^2
\]  

(5.11)

5.2.6 Matrix Vector Multiplier

From the block architecture of the matrix vector multiplier is shown in Figure 5.11. A is a matrix and C is a vector. These matrix and vector are multiplied and the output of the multiplier as vector G=AC. In the accumulator, the output vector is stored. The output of the multiplier is given as one input of the adder. The output of the previous adder is feedback as the second input to the adder. The adder output will be stored in the output RAM while the process completes.
5.2.7 Matrix Inversion Using Given Rotation

For matrix inversion, the method used is called as Given Rotation. Matrix inversion is done by the decomposition QR using Given Rotation [107] is shown in equation 5.12.

The orthogonal matrix which is denoted by Q and R is an upper triangular matrix. From the block diagram, until to get the upper triangular matrix R the input signals of given rotation gets rotated. Based on the given algorithm the rotation can be done from the left GA, a given rotation matrix G (i, j, θ), multiplies another matrix A, only rows i and j of are affected. Given a and b, to find c = cosθ and s = sinθ such that

\[
\begin{bmatrix}
  c & -s \\
  s & c
\end{bmatrix}
\begin{bmatrix}
  a \\
  b
\end{bmatrix}
= 
\begin{bmatrix}
  r \\
  0
\end{bmatrix}
\]

(5.12)

Where c is the length of the vector (a, b), c = a/r, s = -b/r. the upper triangular matrix R can found out using this algorithm. By multiplying the transpose values of G, the Q value can be calculated by using the Equation (5.13)

\[ Q = G_1^T G_2^T ....... G_k^T \]

(5.13)

The transpose of Q and inverse of R has to multiply to get the inverse of A.

```matlab
function BACKSOLVE (U, b)
%Find the solution to Ux = b, where U is an n x n upper triangular matrix
X_n = b_n / u_n
for i = n-1:-1:1
    sum = 0.0;
    for j = i+1:n
        sum = sum + U_{ij} X_j;
    end for;
    x(i) = (b(i) - sum) / U_{ii};
end for;
return x;
end function
```
By simple matrix inversion, the inverse of \( R \) can be calculated using the method of Back-Substitution. S that Identical matrix \( I \) is given as input the simple matrix inversion is shown in the Figure 5.12. Back substitution pseudo code is given below,

```
Figure 5.12: Block Diagram of Matrix Inversion using Given Rotation Method
```

Simple registers are required in the transpose matrix \( Q^T \), due to the usage of scheduling and the calculation is not required for calculation. For transpose of 3×3, the pseudo code is given below,

- Let the input matrix be \( A \)
- Let the matrix holding the transposed be called Transpose.
- For I in 1…3 loop
  - For J in 1…3 loop
    - Transpose (I, J) = A (J, I)
- Return matrix Transpose.

After the process of transpose of \( Q \) and the inverse of \( R \), these values are multiplied by the multiplier and the final output is produced as shown in Equation 5.14.

\[
A^{-1} = R^{-1} Q^T
\]  

(5.14)
5.3 RESULTS AND DISCUSSIONS
The experimental results of the proposed method are presented below. The speech signal and a random noise signal are mixed and then fed as input to the MATLAB model of in the proposed noise degradation system and the resulting signal and produce a noise free signal.

![Figure 5.13: Output Signal, Noisy Signal and Original Signal](image)

The Figure 5.13 shows the input signal and the noise added to the original speech signal and the output signal produced is the filtered output.

![Figure 5.14: Spectrogram for Original Speech Signal](image)
The spectrogram of original speech is shown in Figure 5.14 and filtered speech signal in Figure 5.15. The proposed noise degradation system was implemented in the Xilinx 14.5 using the Verilog language with the family of virtex4, device XC4VFX12, package SF363 and speed -12. The power, area and simulation report can be evaluated by the Xilinx. The RTL schematic of the noise degradation system was shown in Figure 5.16. RTL schematic opens an NGR file and viewed as a schematic of gate-level. After the HDL synthesis phase, the schematic is generated by the synthesis process. The design is pre-optimized and represented in terms of generic symbols, such as adders, multipliers, counters, AND gates, and OR gates, that are independent of the targeted Xilinx device. The RTL schematic includes the modules, used in the noise degradation system. The RFFT, power spectral density, matrix subtractor, matrix inverter, matrix multiplier and matrix-vector multiplier are included in the proposed system of noise degradation.

### 5.3.1 Power

The power result can be calculated by the XPower Analyzer which is an interactive graphical tool. This tool helps in analysis of consumption of power for Xilinx at the targeted device, thermal information like junction temperature can be reported, for the different supply of voltage the static and dynamic power can be calculated and the design of consumption is generated.
Figure 5.16: RTL Schematic for Proposed Noise Degradation System

Figure 5.17: Power Report of RFFT
The Figure 5.17 shows the power report of RFFT in which the dynamic power obtained is 0.000W and quiescent power is 0.027W and the total power obtained in the RFFT module is 0.027W. The power report of matrix inverter was shown in the Figure 5.18. The total power obtained in the matrix inverter is 0.181W in which the Dynamic power is 0.014W and quiescent power is 0.167. The Figure 5.19 shows the power report of proposed noise degradation system in which the dynamic and quiescent power has the value of 0.027W and 0.167W respectively and total obtained power is 0.194W. The Dynamic, Quiescent and total power value of RFFT, power spectral density, matrix subtractor, matrix inverter, matrix multiplier, vector multiplier and proposed technique is tabulated in Table 5.4.

### Table 5.4: Power Consumption by Each Module

<table>
<thead>
<tr>
<th>Modules</th>
<th>Dynamic Power (W)</th>
<th>Quiescent Power (W)</th>
<th>Total Power (W)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RFFT</td>
<td>0.000</td>
<td>0.027</td>
<td>0.027</td>
</tr>
<tr>
<td>Power Spectrum</td>
<td>0.006</td>
<td>0.166</td>
<td>0.172</td>
</tr>
<tr>
<td>Matrix Subtractor</td>
<td>0.009</td>
<td>0.167</td>
<td>0.176</td>
</tr>
<tr>
<td>Matrix Inversion</td>
<td>0.014</td>
<td>0.167</td>
<td>0.181</td>
</tr>
<tr>
<td>Matrix Multiplier</td>
<td>0.018</td>
<td>0.167</td>
<td>0.185</td>
</tr>
<tr>
<td>Vector Multiplier</td>
<td>0.007</td>
<td>0.167</td>
<td>0.174</td>
</tr>
<tr>
<td>Proposed noise degradation system</td>
<td>0.027</td>
<td>0.167</td>
<td>0.194</td>
</tr>
</tbody>
</table>
Figure 5.20: Bar Chart of Power Consumption by Each Module

The bar chart of power consumption of each module is shown in Fig. 5.20 which clearly explains the each module dynamic, quiescent and total power.

5.3.2 Area

The area report includes the number of slices, number of LUT, number of input and output bonds and the number of flip-flops.

Table 5.5: Design Summary of RFFT

<table>
<thead>
<tr>
<th>Device Utilization Summary</th>
<th>Used</th>
<th>Available</th>
<th>Utilization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Slice Flip Flops</td>
<td>117</td>
<td>1,536</td>
<td>7%</td>
</tr>
<tr>
<td>Number of 4 input LUTs</td>
<td>32</td>
<td>1,536</td>
<td>2%</td>
</tr>
<tr>
<td>Number of occupied Slice</td>
<td>69</td>
<td>768</td>
<td>8%</td>
</tr>
<tr>
<td>Number of Slice containing only related logic</td>
<td>69</td>
<td>69</td>
<td>100%</td>
</tr>
<tr>
<td>Number of Slice containing unrelated logic</td>
<td>0</td>
<td>69</td>
<td>0%</td>
</tr>
<tr>
<td>Total Number of 4 input LUTs</td>
<td>32</td>
<td>1536</td>
<td>2%</td>
</tr>
<tr>
<td>Number of bonded IOBs</td>
<td>100</td>
<td>124</td>
<td>80%</td>
</tr>
<tr>
<td>Number of MULT18x18s</td>
<td>4</td>
<td>4</td>
<td>100%</td>
</tr>
<tr>
<td>Number of BUFGMUXs</td>
<td>1</td>
<td>8</td>
<td>12%</td>
</tr>
<tr>
<td>Average Fanout of Non-clock Nets</td>
<td>2.51</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Table 5.6: Design Summary of Matrix Inverter

<table>
<thead>
<tr>
<th>Slice Logic Utilization</th>
<th>Used</th>
<th>Available</th>
<th>Utilization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Slice Flip Flops</td>
<td>91</td>
<td>10,944</td>
<td>1%</td>
</tr>
<tr>
<td>Number of 4 input LUTs</td>
<td>311</td>
<td>10,944</td>
<td>2%</td>
</tr>
<tr>
<td>Number of occupied Slice</td>
<td>218</td>
<td>5472</td>
<td>3%</td>
</tr>
<tr>
<td>Number of Slice containing only related logic</td>
<td>218</td>
<td>218</td>
<td>100%</td>
</tr>
<tr>
<td>Number of Slice containing unrelated logic</td>
<td>0</td>
<td>218</td>
<td>0%</td>
</tr>
<tr>
<td>Total Number of 4 input LUTs</td>
<td>314</td>
<td>10,944</td>
<td>2%</td>
</tr>
<tr>
<td>Number used as logic</td>
<td>303</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number used as a route-thrus</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number used as Shift registers</td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of bonded IOBs</td>
<td>69</td>
<td>240</td>
<td>28%</td>
</tr>
<tr>
<td>Number of BUFG/BUFGCTRLs</td>
<td>1</td>
<td>32</td>
<td>3%</td>
</tr>
<tr>
<td>Number used as BUFGs</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of DSP 48s</td>
<td>21</td>
<td>32</td>
<td>65%</td>
</tr>
<tr>
<td>Average Fanout of Non-clock Nets</td>
<td>1.52</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The Table 5.5 shows the design summary of RFFT and the design summary of matrix inverter were shown in Table 5.6.

### Table 5.7: Design Summary of Proposed Degradation System

<table>
<thead>
<tr>
<th>Slice Logic Utilization</th>
<th>Used</th>
<th>Available</th>
<th>Utilization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Slice Flip Flops</td>
<td>641</td>
<td>10,944</td>
<td>5%</td>
</tr>
<tr>
<td>Number of 4 input LUTs</td>
<td>933</td>
<td>10,944</td>
<td>8%</td>
</tr>
<tr>
<td>Number of occupied Slice</td>
<td>722</td>
<td>5472</td>
<td>13%</td>
</tr>
<tr>
<td>Number of Slice containing only related logic</td>
<td>722</td>
<td>722</td>
<td>100%</td>
</tr>
<tr>
<td>Number of Slice containing unrelated logic</td>
<td>0</td>
<td>722</td>
<td>0%</td>
</tr>
<tr>
<td>Total Number of 4 input LUTs</td>
<td>1,018</td>
<td>10,944</td>
<td>9%</td>
</tr>
<tr>
<td>Number used as logic</td>
<td>887</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number used as a route-thrus</td>
<td>85</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number used as Shift registers</td>
<td>46</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of bonded IOBs</td>
<td>136</td>
<td>240</td>
<td>56%</td>
</tr>
<tr>
<td>Number of BUFG/BUFGCTRLs</td>
<td>2</td>
<td>32</td>
<td>6%</td>
</tr>
<tr>
<td>Number used as BUFGs</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of DSP 48s</td>
<td>11</td>
<td>32</td>
<td>34%</td>
</tr>
<tr>
<td>Average Fanout of Non-clock Nets</td>
<td>1.99</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The Table 5.7 shows the design summary of proposed degradation system which includes the number of flip-flops, slices and LUT. The utilization of the proposed noise degradation system also obtained at particular FPGA device. The graph for various module power and area is shown in Figure 5.21

### Table 5.8: Area Occupied By Each Module

<table>
<thead>
<tr>
<th>Modules</th>
<th>Slices</th>
<th>LUT</th>
<th>Flip-flop</th>
</tr>
</thead>
<tbody>
<tr>
<td>RFFT</td>
<td>69</td>
<td>32</td>
<td>117</td>
</tr>
<tr>
<td>Power Spectrum</td>
<td>8</td>
<td>16</td>
<td>17</td>
</tr>
<tr>
<td>Matrix Subtractor</td>
<td>16</td>
<td>32</td>
<td>32</td>
</tr>
<tr>
<td>Matrix Inversion</td>
<td>218</td>
<td>311</td>
<td>91</td>
</tr>
<tr>
<td>Matrix Multiplier</td>
<td>582</td>
<td>586</td>
<td>1,080</td>
</tr>
<tr>
<td>Vector Multiplier</td>
<td>62</td>
<td>62</td>
<td>64</td>
</tr>
<tr>
<td>Proposed noise degradation</td>
<td>722</td>
<td>933</td>
<td>641</td>
</tr>
</tbody>
</table>

**Figure 5.21: Bar Chart for Area Occupied By Each Module**
Table 5.9: Comparison of Power and Area with Existing Methods

<table>
<thead>
<tr>
<th>Module</th>
<th>Power(W)</th>
<th>Area</th>
<th>Proposed</th>
<th>Existing</th>
<th>Proposed</th>
<th>Existing</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>LUT</td>
<td>Flip-flops</td>
<td>LUT</td>
<td>Flip-flop</td>
</tr>
<tr>
<td>RFFT</td>
<td>0.027</td>
<td>32</td>
<td>17,793</td>
<td>8998</td>
<td>117</td>
<td>791</td>
</tr>
<tr>
<td>Matrix Inverse</td>
<td>0.181</td>
<td>311</td>
<td>4993</td>
<td>791</td>
<td>91</td>
<td>791</td>
</tr>
<tr>
<td>Noise degradation system</td>
<td>0.194</td>
<td>933</td>
<td>8360</td>
<td>2385</td>
<td>641</td>
<td>2385</td>
</tr>
</tbody>
</table>

The Figure 5.22 shows the variation of various modules versus power is show the proposed module is better than other existing modules. Similarity for LUT and Flip flops are shown in Figure 5.23 and Figure 5.24.

The Table 5.9 shows the comparison of power and area with the proposed and existing technique. The power obtained in RFFT is 0.027W which is less as compared to the existing technique [113][114]. Likewise, the area also compared with the existing technique and found that the proposed method was good in the area reduction and power consumption and provide better performance. Existing methods were implemented in Xilinx ISE 10.1i to synthesize our design. The target device is chosen as Xilinx xc5vlxll0t[115].

The comparison was shown by the bar chart the requirement of LUT in the proposed noise degradation method is less as compared to the existing method. The Proposed
method is implemented in Xilinx ISC 14.5 using Verilog-HDL. The Target device is chosen as Vertex-4(xc4vlx160-11ff1148).

![Figure 5.23: Comparison of LUT with Proposed and Existing Method](image)

![Figure 5.24: Bar Chart for the Comparison of Flip-Flop with Proposed and Existing Method](image)

5.3.3 Performance

The performance was analyzed and tabulated in Table 5.10. The signal to noise ratio value of input noisy signal is 24.5277 dB and the filtered output signal has the SNR value 49.0307 dB and it gains high performance. The PSNR value of the filtered output signal is high as compared to the input noisy signal is shown in the Figure 5.25.

![Figure 5.25: Bar Chart of Noisy Input Signal and Filtered Signal](image)
Table 5.10: Performance Analysis

<table>
<thead>
<tr>
<th>Signal</th>
<th>SNR(dB)</th>
<th>MSE(dB)</th>
<th>PSNR(dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input Noisy Signal</td>
<td>24.5277</td>
<td>0.5762</td>
<td>50.5251</td>
</tr>
<tr>
<td>Filtered Output Signal</td>
<td>49.0307</td>
<td>0.4097</td>
<td>52.0059</td>
</tr>
</tbody>
</table>

5.4 CONCLUSION

The FPGA based de-noising system for the enhancement of speech signal in digital hearing aid using Real-valued FFT/IFFT and Wiener filter is proposed. The proposed modified Wiener filter and processor of Real-valued FFT/IFFT used to reduce the consumption of power and hardware requirement of the system. The computational complexity of the system can be reduced by the Matrix inversion using Given Rotation method is proposed. The proposed system is coded, synthesized and simulated in Xilinx ISE and efficient SNR values are exhibited. The SNR analysis proves that the proposed technique brings better performance than the existing method.