Madam Aravind Kumar* and Kamsali Manjunatha Chari

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

https://doi.org/10.1515/jisys-2017-0509
Received October 9, 2017; previously published online April 5, 2019.

Abstract: Speech signals are usually affected by noises during the communication process. For suppressing the noise signal that is combined with the speech signal, a Wiener filter is adapted in digital hearing aids. Weiner filter plays an important role in noise suppression and enhancement by estimating the relation between the power spectra of the noise-affected speech signal and the noise signal. Power consumption and the hardware requirement are the important problems in adapting Weiner filter for major communication systems. In this work, we implemented an efficient Wiener filter and applied it for noise suppression along with a real-valued fast Fourier transform (FFT)/real-valued inverse FFT processor in digital hearing aids. The pipelined process was adopted for increasing the performance of the system. The proposed Wiener filter was designed to remove the iteration problems in the conventional Wiener filter. The division operation was replaced by an efficient inverse and multiplication operation in the proposed design. A modified architecture for matrix inversion with low computation complexity was implemented. The complete design computation was based on IEEE-754 standard single-precision floating-point numbers. The Wiener filter and the whole system architecture was implemented and designed on a Field Programmable Gate Array platform and simulated to validate the results in Xilinx ISE tools. An efficient reduction in power and area was obtained by adapting the proposed method for speech signal noise degradation. The performance of the proposed design was found to be 50.01% more efficient than that of existing designs.

Keywords: Noise, Wiener filter, real-valued FFT, power spectrum, floating point.

1 Introduction

Digital hearing aids are widely used by hearing-impaired people to improve their speech intelligibility and quality of life. However, hearing aid performance is usually degraded due to acoustic feedback, which generates other problems. This phenomenon is produced when the sound propagates from the loudspeaker to the microphone. This causes instability and a high-frequency oscillation that can be perceived by hearing-impaired people if its level exceeds their hearing thresholds. Also, these effects limit the maximum gain that the hearing aid can perform and reduce the sound quality when the gain is close to the limit. To reduce acoustic feedback, several methods based on adaptive algorithms have been used [20] for feedback reduction. A number of techniques are used for speech noise reduction, like non-local diffusion filters [18], acoustic feedback reduction based on finite impulse response and infinite impulse response adaptive filters in digital hearing aids [14], and noise reduction Wiener filter [5]. In the above techniques, the noise is suppressed using filters such as adaptive filters or Wiener filter. Many documented implementations (least mean square, recursive least square, Kalman filters) have been done by using adaptive filters. Adaptive filters are optimal in that they minimize the mean squared estimation error [13] and they can also be computed in real time.

***Corresponding author: Madam Aravind Kumar,** Electronics and Communication Engineering, Grandhi Varalakshmi Venkata Rao Institute of Technology Engineering College, Bhimavaram, Andhra Pradesh 534207, India, e-mail: aravindkumarm0883@gmail.com
Kamsali Manjunatha Chari: Electronics and Communication Engineering, Gandhi Institute of Technology and Management University, Hyderabad, Telangana 530045, India

However, the drawbacks of adaptive filters are that they assume that the process dynamics are linear, only provide a point estimate, and can only handle processes with additive, unimodal noise.

The spectral subtraction technique is a generally known method for noise reduction [24]. In this method, the noisy speech signal is first transformed from the time domain into the frequency domain by means of the fast Fourier transform (FFT). The noise spectrum is then determined in the speech pauses and subtracted from the frequency spectrum of the noisy speech signal before the noisy speech signal is reconverted from the frequency domain into the time domain by means of the inverse FFT (IFFT). The result depends essentially on the accuracy of the determination of the noise spectrum. Although good results are achieved in the case of stationary noise, in practice noises are not stationary and the achievable results are therefore unsatisfactory. In contrast, Weiner filter exploits the signal properties. It controls output error and is also straightforward to design. Among the numerous techniques that have been developed, the optimal Wiener filter [5, 24] can be considered one of the most fundamental noise reduction approaches, which has been delineated in different forms and adopted in various applications. The knowledge of the spectral properties of the original signal and the noise should be known for designing the Wiener filter. To use the availability of certain statistical parameters like mean and correlation function of the original speech signal and unwanted additive noise, the Wiener filter is designed. To reduce the effect of noise signal according to some statistical criterion, the noisy speech signal is fed as input to the Wiener filter. By minimizing the mean square error, the unwanted noise signal can be removed using the Wiener filter.

Although it is not a secret that the Wiener filter may cause some detrimental effects to the speech signal, few efforts have been reported to show the inherent relationship between noise reduction and speech distortion. From the described disadvantages of the noise reduction method using a Wiener filter, the object of altering the noise estimation by means of the Wiener filter and the rules for transforming the noisy speech signals from the time domain into the frequency domain and vice versa were considered, so as to permit an adaption to the non-linear transmission behavior of the human ear.

For reducing the unwanted noise signal from the original speech signal, the digital hearing aid is designed with our proposed effective noise degradation architecture in this paper. The continuous time domain signal is segmented into overlapping chunks called frames, and the frames are multiplied by a window function for avoiding the spectral artifacts, to perform frequency domain processing of the speech signal in our system. The signal in time domain is converted into frequency domain by using an FFT processor, and to reduce the noise signal from the speech signal, the Wiener filter is used in the digital hearing aid. The output signal from the Wiener filter is converted into time domain by using an IFFT processor and then multiplied with the same window function, and the frames are then overlapped to create a continuous output signal.

In our work, to make the system more efficient for the domain conversion instead of using two separate processors for FFT and IFFT, we have designed it within a single processor. The FFT is defined over complex data; however, in many applications, the input is real. Real-valued FFT (RFFT) algorithms take advantage of the symmetry properties of the FFT and have a speed advantage over complex algorithms of the same length. RFFT achieves higher throughput area and lower latency. Computational efficiency is low to implement. Hence, we prefer RFFT implementation rather than an FFT processor for our work, as the input signal is a real-valued signal. The modified Wiener filter implementation in our work incorporates an efficient power spectrum and energy analysis technique. Low-power floating-point adders and multipliers are adopted to contribute to the power reduction of our work. Also, in very large scale integration (VLSI), floating-point division [16] is a slow process and it also takes more clock cycles for computation; thus, in order to speed up our computations, we replaced the floating-point division operations with a reciprocal and a multiplier operation. We also made the whole architecture pipelined so that the performance of the system is increased.

The rest of the paper is organized as follows. Section 2 reviews some related works to our work. Section 3 discusses the motivation for the proposed work along with the modified Wiener filter and the methods adopted for making it efficient. Section 4 explains each block of the proposed noise degradation system, and their outputs are verified individually. Section 5 reports the simulation result compared with other existing methods, and the work is concluded in Section 6.

2 Related Works

A wide range of literature reports about VLSI FFT/IFFT and Weiner filter design are available, and we discuss a few among them in this section. Arunachalam and Raj [1] concentrated on the trivial multiplications in the input stage of the IFFT unit and replaced them by the proposed "pass-logic". In an orthogonal frequency division multiplexing-based digital transmitter [7], the IFFT processing unit consumes the most hardware area and power, especially because of the twiddle multipliers in the Cooley-Tukey-based decimation-infrequency (DIF) IFFT architecture. The replacements can be possible because the inputs are bitwise with binary-phase shift keying (PSK) or quadrature-PSK digital modulation. The input stage of DIF-FFT for 8–128 points (N) were implemented with multipliers and "pass-logics". The performance improvements (PIs) of their proposed FFT/IFFT implementation have been analyzed. For a 64-point FFT in Field Programmable Gate Array (FPGA), the number of slices was reduced by 9% and the total power by 6.5%. The same implementation on an ASIC consumed 28% less power and 27% lesser gates. In 128-point implementation, these PIs are more than those of the 64-point implementation; thus, PI is in upward trend as N increases. A chip for FFT processing as per IEEE 802.11a specifications (64-point, 16-bit data) is designed with pass-logics, which uses 24,947 gates and consumes 6.45 mW at 1.8 V, 20 MHz in 0.18 μm 1P6M complementary metal oxide semiconductor (CMOS) process.

Yu and Yen [25] presented a novel 128/256/512/1024/1536/2048-point single-path delay feedback (SDF) pipeline FFT processor for long-term evolution and mobile worldwide interoperability for microwave access systems. FFT is widely used in digital signal processing and telecommunications, particularly in orthogonal frequency division multiplexing systems, to overcome the problems associated with orthogonal subcarriers. Their proposed design employs a low-cost computation scheme to enable 1536-point FFT, which significantly reduces hardware costs as well as power consumption. In conjunction with the aforementioned 1536-point FFT computation scheme, their proposed design included efficient three-stage SDF pipeline architecture on which to implement a radix-3 FFT. The new radix-3 SDF pipeline FFT processor simplifies its data flow and is easy to control, and the complexity of the resulting hardware is lower than that of existing structures. Their paper also formulated a hardware-sharing mechanism to reduce the memory space requirements of the proposed 1536-point FFT computation scheme. Their proposed design was implemented using 90-nm CMOS technology. Post-layout simulation results revealed a die area of approximately 1.44×1.44 mm^2 with power consumption of only 9.3 mW at 40 MHz.

Tsai and Lin [19] presented a generalized conflict-free memory addressing scheme for memory-based FFT processors with parallel arithmetic processing units made up of radix multi-path delay commutator. Their proposed addressing scheme considers the continuous-flow operation with minimum shared memory requirements. To improve throughput, parallel high-radix processing units are employed. They prove that the solution to non-conflict memory access satisfying the constraints of the continuous-flow, variable-size, higher-radix, and parallel-processing operations indeed exist. In addition, a rescheduling technique for twiddle-factor multiplication is developed to reduce hardware complexity and to enhance hardware efficiency. From the results, they can see that their proposed processor has high utilization and efficiency to support flexible configurability for various FFT sizes with fewer computation cycles than the conventional radix-2/radix-4 memory-based FFT processors.

Biswas et al. [3] discussed about wavelet packet acoustic features that are found to be very promising in unvoiced phoneme classification tasks, but they are less effective in capturing periodic information from voiced speech. This motivated them to develop a wavelet packet-based feature extraction technique that signifies both the periodic and aperiodic information. This method is based on the parallel distributed processing technique inspired by the human speech perception process. This front-end feature processing technique employs equivalent rectangular bandwidth (ERB) filter, like the wavelet speech feature extraction method called WERB-SPADE (Wavelet ERB Sub-band based Periodicity and Aperiodicity Decomposition). Wiener filter is used at the front end to minimize the noise for further processing. The speech signal is filtered by 24-band ERB, like wavelet filter banks, and then the output of each sub-band is processed through a comb filter. Each comb filter is designed individually for each sub-band to decompose the signal into periodic and aperiodic features. Thus, it carries the periodic information without losing certain important information like formant

transition incorporated in aperiodic features. Hindi phoneme classification experiments with a standard hidden Markov model recognizer under both clean-training and multi-training conditions are conducted. This technique shows significant improvement in voiced phoneme class without affecting the performance of unvoiced phoneme class.

A Wiener filter approach to microphone leakage reduction in close-microphone applications was implemented by Wang et al. [21]. The applicability of two widely used signal enhancement methods to this problem was discussed, namely blind source separation and noise suppression. They showed that the noise suppression framework was a valid choice and can effectively address the problem of microphone leakage. An extended form of the single-channel Wiener filter was used, which took into account the individual audio sources to derive a multichannel noise term. A novel power spectral density (PSD) estimation method was also implemented based on the identification of dominant frequency bins by examining the microphone and output signal PSDs. The performance of the method was examined for simulated environments with various source-microphone setups, and it was shown that the proposed approach efficiently suppressed leakage.

A method for noise estimation using mean square cross-prediction error (MSCPE) for speech enhancement was designed by Wu et al. [22]. They experimented the feasibility of noise extraction from noisy speech, and presented a two-stage approach for speech enhancement. The MSCPE-based blind source extraction algorithm was utilized to extract the additive noise from the noisy speech signal in the first stage. After that, a modified spectral subtraction and a modified Wiener filter approach were designed to extract the speech signal in the second stage, where all the frequency spectra of the extracted noise were utilized. Theoretical justification showed that the MSCPE-based algorithm can extract the desired signal from mixed sources. Experimental results showed that the averaged correlation coefficient between the extracted noise and the original additive noise are beyond 85% for Gaussian noise and beyond 75% for real-world noise at signal-to-noise ratio (SNR) $= 0$ dB, and the designed speech enhancement approaches performed better than conventional methods, such as spectral subtraction and Wiener filter.

Yang et al. [23] proposed the first VLSI design enabling high-throughput data detection in single-carrier frequency-division multiple access-based large-scale multiple-input multiple-output (MIMO) systems. Large-scale (or massive) MIMO is expected to be one of the key technologies in next-generation multi-user cellular systems based on the upcoming 3GPP LTE Release 12 standard, for example. They proposed a new approximate matrix inversion algorithm relying on a Neumann series expansion, which substantially reduces the complexity of linear data detection. They analyzed the associated error, and compared its performance and complexity to those of an exact linear detector. They presented corresponding VLSI architectures, which perform exact and approximate soft-output detection for large-scale MIMO systems with various antenna/user configurations. The reference implementation results for a Xilinx Virtex-7 XC7VX980T FPGA show that their designs are able to achieve $>$600 Mb/s for a 128-antenna, 8-user 3GPP LTE-based large-scale MIMO system. They finally provided a performance/complexity trade-off comparison using the presented FPGA designs, which revealed that the detector circuit of choice is determined by the ratio between BS antennas and users, as well as the desired error-rate performance.

Carvajal et al. [4] proposed a selected inversion technique to reduce the computation cost of matrix inversion for vectorless verification. Vectorless power grid verification is a practical approach for early-stage safety check without input current patterns. The power grid is usually formulated as a linear system and requires intensive matrix inversion and numerous linear programming, which is extremely time consuming for large-scale power grid verification. In their paper, the power grid is represented in the manner of a domain-decomposition approach. The locality existence among power grids is exploited to decide which blocks of matrix inversion should be computed while the remaining blocks are not necessary. The vectorless verification could be purposefully performed by this manner of selected inversion, while previous direct approaches are required to perform full matrix inversion and then discard small entries to reduce the complexity of linear programming. Meanwhile, constraint locality is proposed by them to improve the verification accuracy. In addition, a concept of quasi-Poisson block is introduced to exploit grid locality among realistic power grids, and a scheme of pad-aware partitioning is proposed to enable the selected inversion approach available for practical use. Experimental results show that their proposed approach could achieve significant speedups compared with previous approaches while still guaranteeing the quality of solution accuracy.

3 Research Methodology

For suppressing the noise signal that is combined with the speech signal, Wiener filter is adapted [5, 24]. Weiner filter plays an important role in noise suppression and enhancement by estimating the relation between the autocorrelation of the noise-affected speech signal and the noise signal. The architecture of Wiener filter includes five main processing units, which are PSD, matrix inverter, matrix subtractor, matrix multiplier, and matrix-vector multiplier. Autocorrelation is a special case of cross-correlation that computes a signal with itself [6]. This provides the relationship between the noisy and noise-free signal. Matrix inversion is an important challenge for researchers while designing any signal processing system, as it consumes a vast area and power [8].

Considering the matrix inversion problem in Wiener filter design, we are going to implement a noise degradation system for noisy speech signals in digital hearing aids incorporating a power- and hardware-efficient Wiener filter. In this work, the pre-processing and post-processing of Wiener filter for converting the input speech and noise signal in time domain to frequency domain and vice versa is done by adopting our pre-designed real-valued FFT processor by contributing it to modify as an RFFT/real-valued IFFT (RIFFT) processor [2]. Our main contribution with this work is a low-power and hardware-efficient matrix inversion module design adopting the advantage of QR [10] decomposition with the given rotation. As the RFFT/RIFFT process is done using a single processor and also by the use of a modified analytic method, we can design an efficient speech signal enhancement system.

4 Proposed Method

The proposed noise degradation system includes two main blocks: (i) efficient pipelined architecture for RFFT/RIFFT processor and (ii) a modified Wiener filter. The overall block diagram of the proposed system is shown in Figure 1.

4.1 Efficient Pipelined Architecture for RFFT/RIFFT Processor

The initial process in the proposed technique for noise degradation is the transformation of input signals in the time domain to the frequency domain. As speech and noise signals are real-valued signals, the conventional FFT architecture for domain conversion can be replaced with a 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 here to design the FFT/IFFT processor.

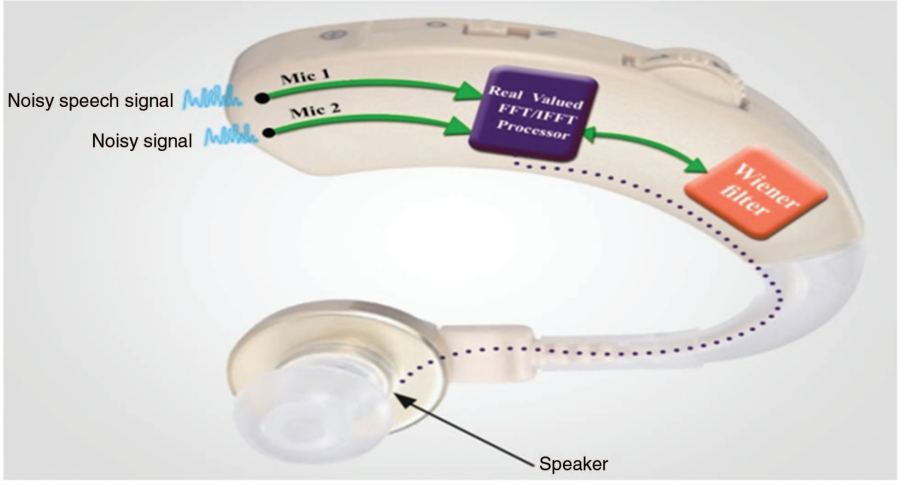

Figure 1: Overall Block Diagram of the Proposed System.

The basic FFT equation is given as

$$X(\omega) = \sum_{\phi=0}^{M-1} x(\phi)\, W_M^{\phi\omega}, \quad \omega = 0,\ 1,\ \ldots,\ M-1, \tag{1}$$

where

$$W_M^{\phi\omega} = e^{-j(2\pi/M)} \tag{2}$$

is the twiddle factor.

The IFFT equation is

$$\boldsymbol{x[\phi]} = \frac{1}{\boldsymbol{M}} \sum_{\boldsymbol{\omega}=0}^{\boldsymbol{M}-1} \boldsymbol{X[\omega]} \boldsymbol{W_M^{-\phi\omega}}, \tag{3}$$

where

$$\boldsymbol{\phi} = 0, 1, \ldots, \boldsymbol{M}-1.$$

Then, Eq. (3) can be rewritten as

$$\boldsymbol{x[\phi]} = \frac{1}{\boldsymbol{M}} \sum_{\boldsymbol{\omega}=0}^{\boldsymbol{M}-1} [\boldsymbol{X[\omega]} * \boldsymbol{W_M^{\phi\omega}}] * . \tag{4}$$

By using the relation between Eqs. (3) and (4), we can design the basic structure of our FFT/IFFT processor, as shown in Figure 2.

4.1.1 RFFT

The RFFT architecture includes four stages of pipelining, as shown in Figure 3. The working process of each stage can be discussed as below.

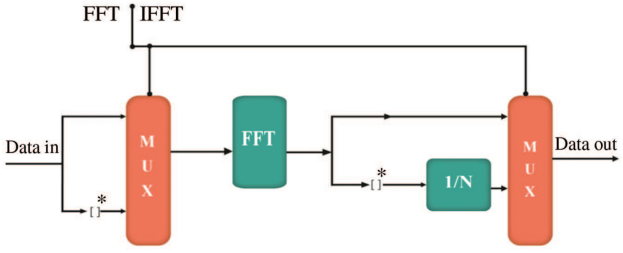

Figure 2: Basic Structure of Our FFT/IFFT Processor.

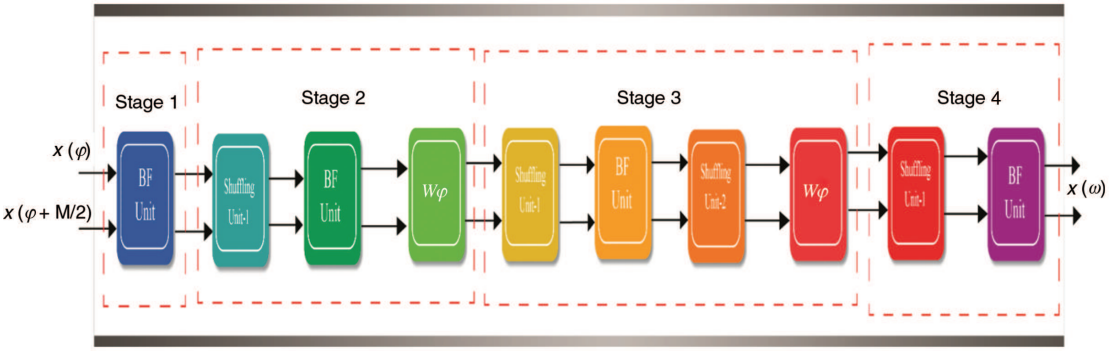

Figure 3: Block Schematic for Two-Parallel Pipelined Architecture for 16-Point Radix-2 RFFT.

Stage 1

At stage 1, the butterfly unit will process the pair of real samples $x(\phi)$ and $x(\phi + M/2)$. The butterfly unit consists of a 2:1 multiplexer with one selector line S. When the inputs are real, then the selector line S is set to 1 and the butterfly starts to compute the input values. When the inputs are complex S set to 0, then the multiplexer just passes the input without computation.

Stage 2

At stage 2, the architecture consists of shuffling unit, butterfly unit, and twiddle factor block A. The shuffling unit is used to transform the order of the data that are required from stage 1 to stage 2, which also contains a 2:1 multiplexer and two delay elements at input and output of the multiplexer. Figure 4 shows the architecture of the $\boldsymbol{W}^{\boldsymbol{\phi}}$ module. The stage includes four twiddle factors, as $\boldsymbol{W}^0$, $\boldsymbol{W}^1$, $\boldsymbol{W}^2$, and $\boldsymbol{W}^3$, with real and imaginary values as tabulated in Table 1.

From Table 1, the value of $\boldsymbol{W}^0$ is 1, so the selector line S is set to 0 and the input passes to the output without any complex multiplication. For twiddle factors $\boldsymbol{W}^1$ and $\boldsymbol{W}^3$, the selector line is set to 1 and allow the multiplexer for complex multiplication. To reduce the number of additions and shifts, canonical signed digit (CSD) is introduced. In CSD calculation, we have to convert the twiddle factor coefficients from binary to CSD, as shown in Table 2. The steps for conversion of binary to CSD is given as follows:

Step 1: Check consecutive number of 1's in the binary sequence.

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 "−".

For $\boldsymbol{W}^2$, real and imaginary values are similar; hence, we can use our modified shift and add/subtract module only.

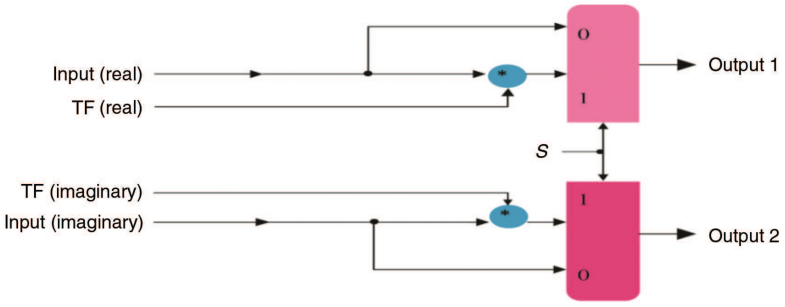

Figure 4: Architecture of $\boldsymbol{W}^{\boldsymbol{\phi}}$ Module.

Table 1: Twiddle Factor Real and Imaginary Coefficients for $M = 16$.

Twiddle Factor (W^k)	Real	Imaginary
W^0	1	1
W^1	0.9239	0.3827
W^2	0.7071	0.7071
W^3	0.3827	0.9239

Table 2: Twiddle Factor Coefficients for $M = 16$.

Decimal	Binary	CSD
0.9239	0011111101101100	0100000-10-10-00
0.3827	0011111011000011	010000-10-00010-
0.7071	0011111100110101	0100000-010-0101

Stage 3
At stage 3, shuffling unit 1 transforms the order of the data that are required from stage 2 to stage 3, then shuffling unit 2 also shuffles the computed samples from the butterfly unit. In case of shuffling unit 2, the initial selector signal "0" last for 21 clock cycles and rest for the clock cycles it operates similar to the shuffling unit 1. In the twiddle factor block B, we use only the twiddle factor $\boldsymbol{W}^2$; hence, we can adopt the same as before in stage 2.

Stage 4
At stage 4, the shuffling unit transforms the samples from stage 3 to stage 4, then the butterfly unit computes the samples and we get the output sample $x(\omega)$.

4.2 Modified Wiener Filter Design

Some assumption should be made for the efficient design of the Wiener filter. First, the noise-affected input speech signal is single-channel (from one source), and the noise and speech signals are uncorrelated. If the noise affecting the speech signal is additive, then

$$\boldsymbol{x}(\boldsymbol{t}) = \boldsymbol{s}(\boldsymbol{t}) + \boldsymbol{a}(\boldsymbol{t}), \tag{5}$$

where
$\boldsymbol{x}(\boldsymbol{t})$ – noisy speech signal in time domain;
$\boldsymbol{s}(\boldsymbol{t})$ – original speech signal in time domain;
$\boldsymbol{a}(\boldsymbol{t})$ – additive noise in time domain.

By merely observing the time domain samples, detecting noise or interference in the input signal is very difficult. The analysis and detection of such signals become easy by mapping the signals in frequency domain. Thus, we need to convert the time domain noise-affected signal to the frequency domain. By taking RFFT, we get

$$\boldsymbol{X}(\boldsymbol{f}) = \boldsymbol{S}(\boldsymbol{f}) + \boldsymbol{A}(\boldsymbol{f}). \tag{6}$$

The original speech signal $\boldsymbol{S}(\boldsymbol{f})$ can be extracted from the noisy signal by multiplying the noisy speech signal $\boldsymbol{X}(\boldsymbol{f})$ with the Wiener filter function $W(f)$.

$$\boldsymbol{S}(\boldsymbol{f}) = \boldsymbol{W}(\boldsymbol{f})\boldsymbol{X}(\boldsymbol{f}). \tag{7}$$

In Eq. (7), ***W(f)*** represent the Wiener filter in frequency domain and can be estimated as

$$\boldsymbol{W}(\boldsymbol{f}) = \frac{|\boldsymbol{S}(\boldsymbol{f})|^2}{|\boldsymbol{S}(\boldsymbol{f})|^2 + |\boldsymbol{A}(\boldsymbol{f})|^2}, \tag{8}$$

where
$|\boldsymbol{S}(\boldsymbol{f})|^2$ is the PSD of the original speech signal;
$|\boldsymbol{A}(\boldsymbol{f})|^2$ is the PSD of the noise speech signal.

In reality, we have no idea about the original speech and noise spectra, as the input to the device is noisy speech signal and noisy signal. The Wiener filter is approximated from the noisy speech signal and the estimated noisy signal as

$$\boldsymbol{W}(\boldsymbol{f}) \approx \frac{|\boldsymbol{X}(\boldsymbol{f})|^2 - |\boldsymbol{N}(\boldsymbol{f})|^2}{|\boldsymbol{X}(\boldsymbol{f})|^2}, \tag{9}$$

where
$|\boldsymbol{X}(\boldsymbol{f})|^2$ is the PSD of input noisy speech signal;
$|\boldsymbol{N}(\boldsymbol{f})|^2$ is the PSD of input noisy signal.

Taking IFFT, we get the original signal $s(t)$

$$s(t) = RIFFT\{S(f)\}. \tag{10}$$

To implement an efficient denoising system for speech signal enhancement, we have to realize Eqs. (5) to (10) with optimal computation units that can adapt to various criteria like low power, high throughput, and less area consumption. Figure 5 shows the complete architecture of our proposed noise degradation system.

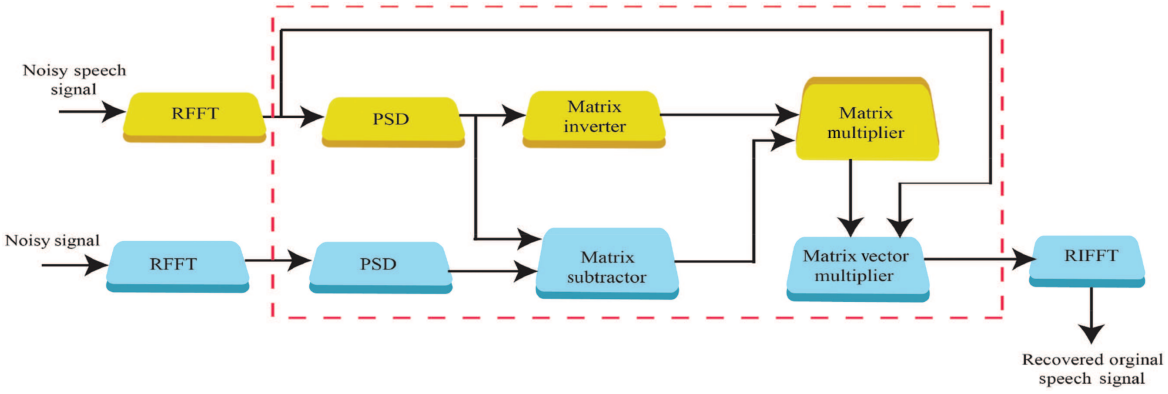

Figure 5: Complete Architecture of Our Proposed Noise Degradation System.

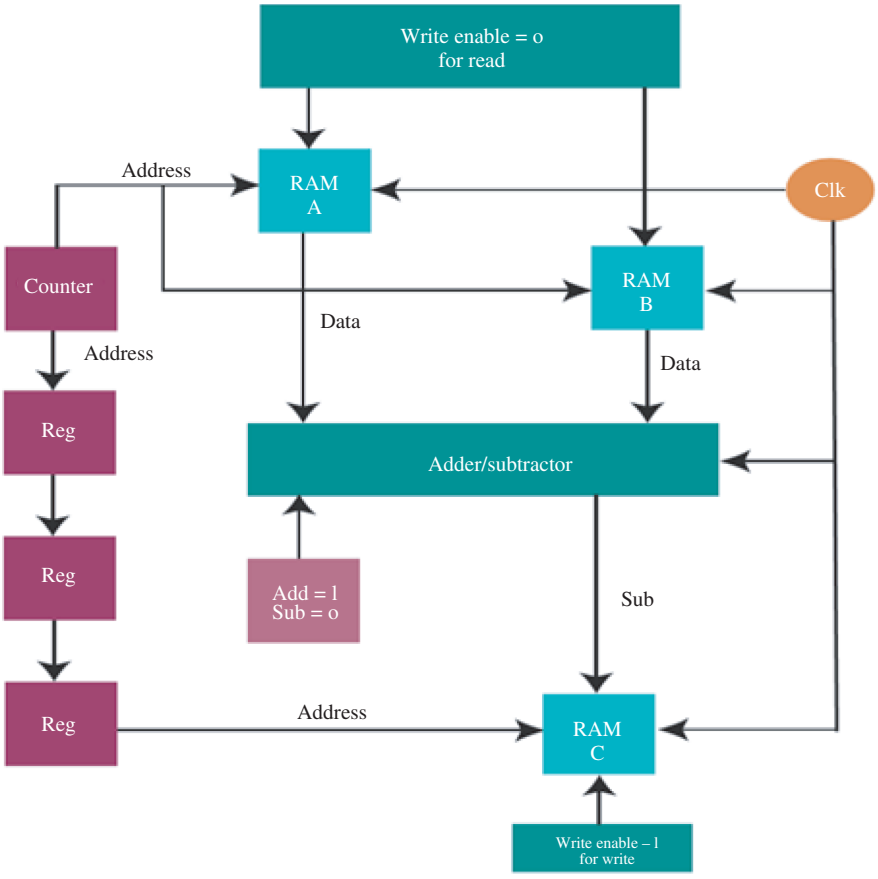

Figure 6: Block Architecture of Matrix Subtractor.

4.2.1 Matrix Subtractor

The algorithms for adding and subtracting matrices are the same. The main idea is that similarly addressed cells of the two input matrices are added together or subtracted from each other. A detailed Block Architecture of Matrix Subtractor is shown in Figure 6. The result is then placed at the same address (cell) in the output matrix. Given three 4×4 matrices, A, B, and C, the following pseudocode describes matrix subtraction. The same pseudocode can be used to describe addition by replacing the "$-$" sign with a "$+$" sign.

for $i = 1 : 4$

for $j = 1 : 4$

$C[i, j] = A[i, j] - B[i, j]$;

end for;

end for.

From the hardware implementation, values from two random-access memory (RAM) modules will be passed to the adder/subtractor module. The result of the adder/subtractor will then be written to the third RAM module. This will be used to address the three RAM modules while the registers will delay the address signal to the output RAM. These registers will compensate for the delay caused by the adder/subtractor logic. This is important to ensure that the result is written to the proper address.

4.2.2 Matrix Multiplier

Basically, every row of the first matrix must be multiplied by every column of the second matrix. For a row-column pair, every element of the row is multiplied by the similarly addressed element of the column. The sum of the four products forms one element of the resulting matrix. A detailed Block Architecture of Matrix Multiplier is shown in Figure 7. The following pseudocode describes this process:

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, counters A and B 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. Each of the output bits from the decoder is multiplexed with a logic low and sent to the write enable of one of the temporary memories. When write is enabled, the decoder determines which of the four temporary memories is written to. The truth table in Table 3 describes the behavior of the decoder. The decoder input is the two-bit counter. Counter D has two addressing modes depending on the read/write state of the temporary memory. When in write mode, the counter increments every four clock cycles to ensure that each temporary memory module is written to before an address change. When in read mode, the counter increments every clock cycle. This addressing scheme is used to read from the temporary memories and write to the final result memory. The address bus controlling the output RAM must be pipelined to compensate for the combinational logic delay of the adders. Two stages of adders are used to add all the products to form an element of the result

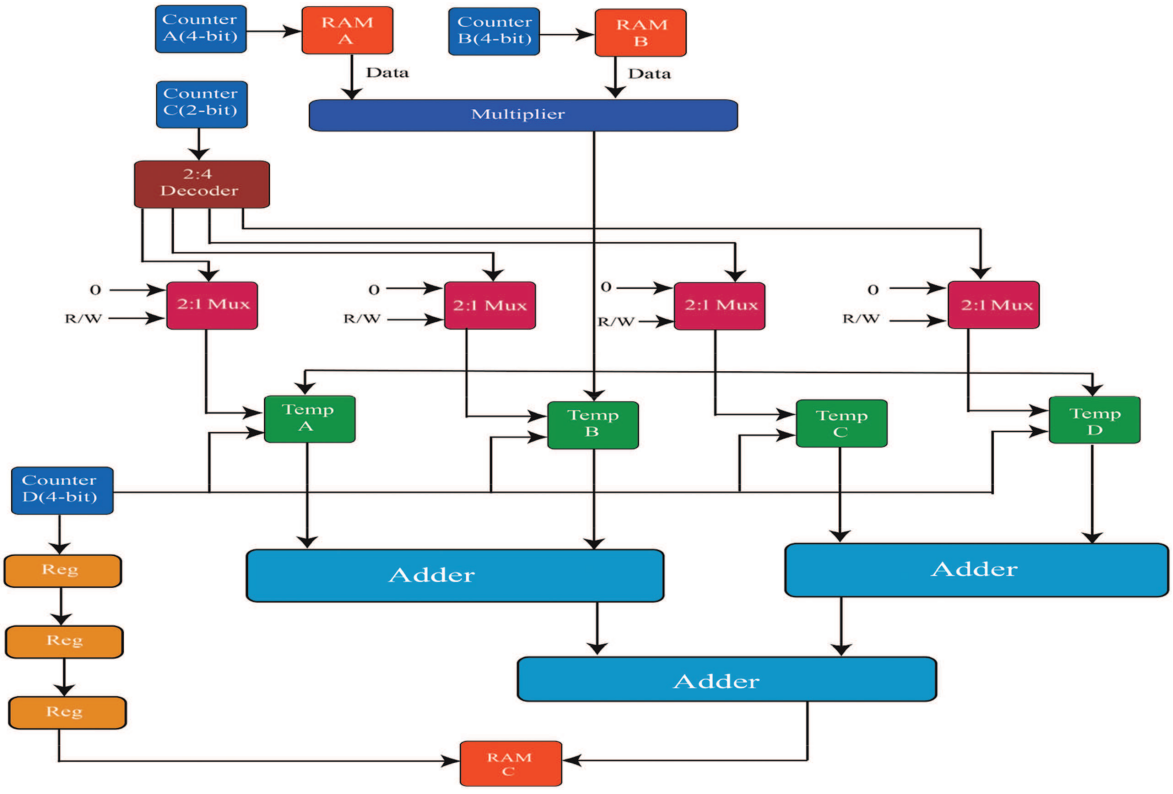

Figure 7: Block Architecture of Matrix Multiplier.

Table 3: Truth Table of Decoder.

Counter	Decoder
00	0001
01	0010
10	0100
11	1000

matrix. The first stage is made up of two adders, each of which adds two of four products. The second stage consists of one adder that adds the outputs of the previous adders together. The output of this third adder is an element of the final matrix product and is written to the output RAM.

4.2.3 PSD

From the block-level architecture of PSD computation as shown in Figure 8, we can calculate the PSD of the input sequence using an N-point FFT estimator, window filter, and absolute square multiple accumulator circuits (ASMACs). From the block diagram, the input signal is divided into $N/2$ samples. Then, FFT is applied to each $N/2$ sample. Windowing block is used to calculate the subset of the dataset to avoid complexity of the whole dataset calculation. The function of the ASMAC circuit is to compute the periodograms and average them over L segments. The control blocks control the address decoder and multiplexer to correctly accumulate the periodograms outputs over different segments. Finally, we get PSD of the input signal as

$$PSD = (1/K)\sum |X_k|^2. \tag{11}$$

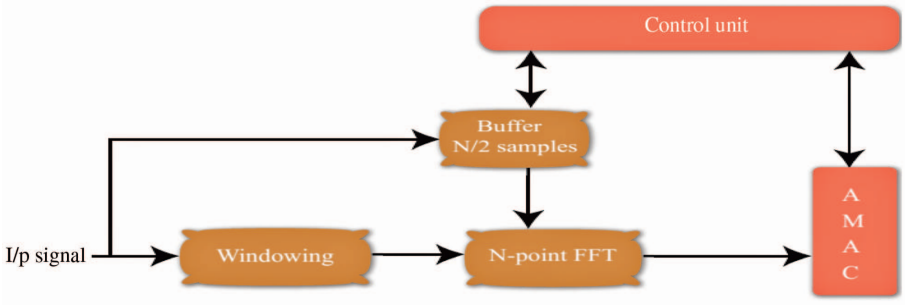

Figure 8: Block Architecture of PSD.

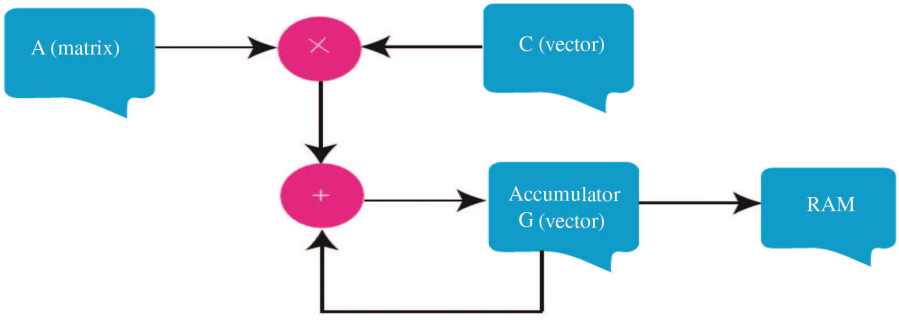

Figure 9: Block Architecture of Matrix Vector Multiplier.

4.2.4 Matrix Vector Multiplier

From the block architecture of the matrix vector multiplier as shown in Figure 9, A is a matrix and C is a vector. These matrix and vector are multiplied, and we get the multiplier output as vector $G = AC$. Then, the output vector is stored in the accumulator. The output of the multiplier is given as one input of the adder. The previous output of the adder is feedback as the second input to the adder. When the process is completed, the output of the adder will be stored in the output RAM.

4.2.5 Matrix Inversion Using Given Rotation

In this paper for matrix inversion, we used the method called given rotation. Matrix inversion is done by QR decomposition using given rotation. Q is an orthogonal matrix and R is an upper triangular matrix. From the block diagram of given rotation, as shown in Figure 10 the input signal gets rotated until we obtain the upper triangular matrix R. The rotation is done based on the given algorithm.

When a given rotation matrix $G(i, j, \theta)$ multiplies another matrix A, from the left GA, only rows i and j of A are affected. Given a and b, find $c = \cos\theta$ and $s = \sin\theta$ such that

$$\begin{bmatrix} c & -s \\ s & c \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} r \\ 0 \end{bmatrix}, \tag{12}$$

where $r = \sqrt{a^2 + b^2}$ is the length of the vector (a, b), $c = a/r$, $s = -b/r$. Using this algorithm, we can get the upper triangular matrix R. Then, the Q value is calculated by multiplying the transpose of G values, i.e.

$$Q = G_1^T G_2^T, \cdots, G_k^T. \tag{13}$$

To get the inverse of A, we have to multiply the transpose of Q and inverse of R. The inverse of R will be calculated by simple matrix inversion using a back-substitution method. The identical matrix I is given by input of the simple matrix inversion. The pseudocode for the back substitution is given below:

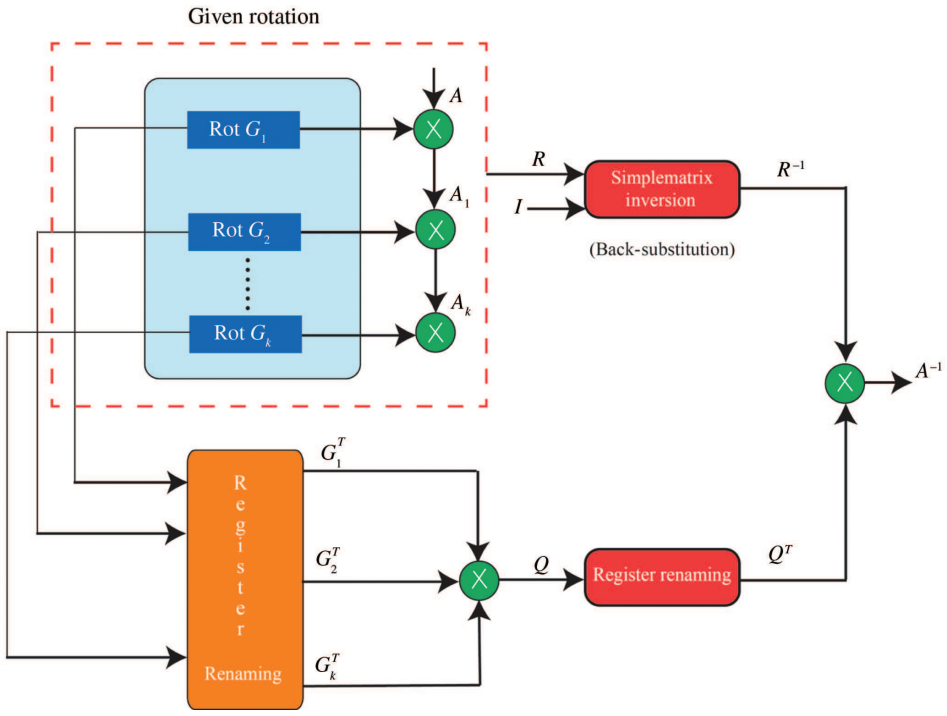

Figure 10: Block Diagram of Matrix Inversion Using the Given Rotation Method.

function BACKSOLVE (U, b)
% Find the solution to $Ux = b$, *where* U *is an* $n \times n$ *upper triangular matrix*
$X_n = b_n / u_{nn}$
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.

The transpose matrix Q^T requires simple register renaming due to usage of scheduling and does not require any calculation. The pseudocode for the transpose of 3×3 is given below:

1. Let the input matrix be A;
2. Let the matrix holding the transpose be called transpose;
3. For I in 1...3 loop;
   i. For J in 1...3 loop;

$$\text{Transpose}\,(I,\ J) = A\,(J,\ I).$$

4. Return matrix transpose.
   After getting the transpose of Q and the inverse of R, these values are multiplied by the multiplier, then we get the final output as

$$A^{-1} = R^{-1} Q^T. \tag{14}$$

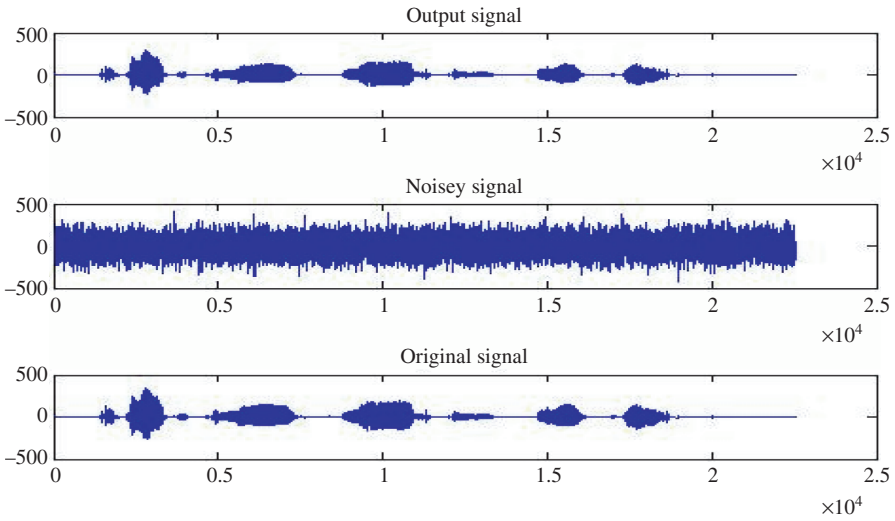

Figure 11: MATLAB Plot for Filtered Output Speech Signal, Noisy Signal, and Original Noise-Free Signal.

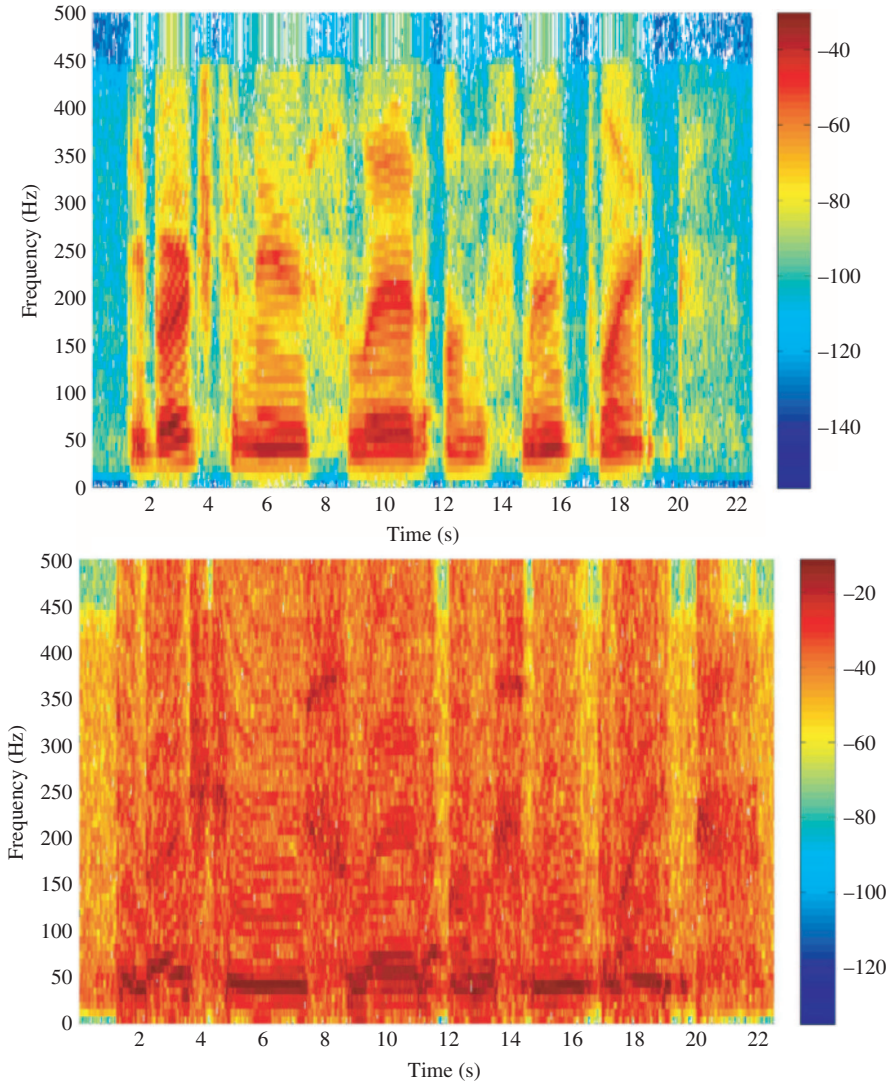

Figure 12: Spectrogram for Original Speech Signal and Filtered Speech Signal.

5 Results and Comparison

The experimental results of our proposed method are presented below. Initially, a sample speech signal and a random noise signal is mixed together and then fed to the MATLAB model (MathWorks, Natick, MA, USA) of our proposed noise degradation system, and the resulting signal is found to be noise-free, as shown in Figure 11.

Then, our proposed architecture is coded in Xilinx ISC 14.5 using Verilog-HDL. The target device is chosen as Vertex-4(xc4vlx160-11ff1148). Moreover, all experiments were performed on 3.10 GHz Intel(R) i5, 4.00 GB RAM and 32-bit operating system with Windows 8 Professional (Microsoft, Redmond, WA, USA). Figure 12 shows the Spectrogram for Original Speech Signal and Filtered Speech Signal. Figure 13 exhibits the register-transfer level (RTL) schematic of our proposed noise degradation system.

5.1 Power

The report generated by the XPower analyzer tool (Xilinx, San Jose, CA, USA) is shown in Figure 14. Our implemented noise degradation system consumes a dynamic power of 0.027 W and a quiescent power of 0.167 W. Hence, our proposed architecture for noise degradation of speech signal needs a total power supply of 0.194 W. The power consumption by the sub-modules is listed in Table 4.

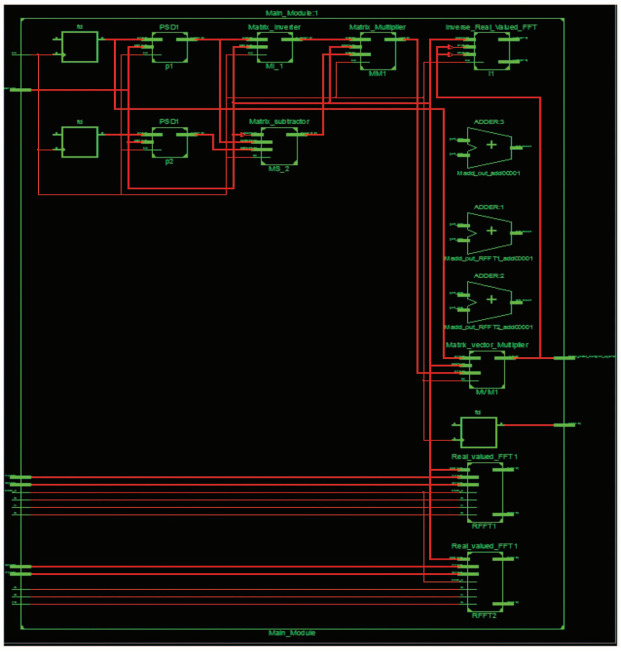

Figure 13: RTL Schematic for the Proposed Noise Degradation System.

Supply Summary		Total	Dynamic	Quiescent
Source	Voltage	Current (A)	Current (A)	Current (A)
Vccint	1.200	0.095	0.023	0.072
Vccaux	2.500	0.031	0.000	0.031
Vcco25	2.500	0.001	0.000	0.001

	Total	Dynamic	Quiescent
Supply Power (W)	0.194	0.027	0.167

Figure 14: XPower Analyzer Report for Our Proposed Noise Degradation System.

Table 4: Power Consumption by Each Module.

Modules	Dynamic power (W)	Quiescent power (W)	Total power (W)
RFFT	0.000	0.027	0.027
Power spectrum	0.006	0.166	0.172
Matrix subtractor	0.009	0.167	0.176
Matrix inversion	0.014	0.167	0.181
Matrix multiplier	0.018	0.167	0.185
Vector multiplier	0.007	0.167	0.174
Proposed noise degradation system	0.027	0.167	0.194

5.2 Area

Figure 15 shows the device utilization table for our proposed noise degradation system. The proposed design occupies 8 among the available 5472 slices utilizing about 1% of the available resources and utilizes 16 lookup tables (LUTs) among the available 10,944, thereby utilizing about 1% of the resources. Table 5 lists the area occupied by various sub-modules used within our proposed system.

Table 6 lists the power and area comparison of our proposed noise degradation system with other existing systems. From the comparison table, we can prove that our proposed noise degradation is much efficient in power and area than the existing systems.

5.3 Performance

The performance of our system can be analyzed efficiently by calculating the SNR values. Table 7 shows the results of the analyses of SNR for input noisy signal and the Wiener-filtered noise-free signal SNR, and also

Device Utilization Summary			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Flip Flops	641	10,944	5%
Number of 4 input LUTs	933	10,944	8%
Number of occupied Slices	722	5472	13%
Number of Slices containing only related logic	722	722	100%
Number of Slices containing unrelated logic	0	722	0%
Total Number of 4 input LUTs	1018	10,944	9%
Number used as logic	887		
Number used as a route-thru	85		
Number used as Shift registers	46		
Number of bonded IOBs	136	240	56%
Number of BUFG/BUFGCTRLs	2	32	6%
Number used as BUFGs	2		
Number of DSP48s	11	32	34%
Average Fanout of Non-Clock Nets	1.99		

Figure 15: Device Utilization Summary for the Proposed Noise Degradation System.

Table 5: Area Occupied by Each Module.

Modules	Slices	LUT	Flipflop
RFFT	69	32	117
Power spectrum	8	16	17
Matrix subtractor	16	32	32
Matrix inversion	218	311	91
Matrix multiplier	582	586	1080
Vector multiplier	62	62	64
Proposed noise degradation system	722	933	641

Table 6: Comparison of Power and Area with Existing Methods.

Module	Power (W)		Area			
	Proposed	Existing	Proposed		Existing	
			LUT	Flipflop	LUT	Flipflop
RFFT	0.027	0.121 [12]	32	117	17,793 [12]	8998 [12]
Matrix inverse	0.181	0.191 [9]	311	91	4993 [17]	791 [17]
Noise degradation system	0.194	0.194 [9]	933	641	8360 [9]	2385 [9]

Table 7: Performance Analysis.

	SNR (dB)	PSNR (dB)	MSE
Filtered output signal	49.0307	52.0059	0.4097
Input noisy signal	24.5277	50.5251	0.5762
Improved SNR	24.503	29.3226	

Table 8: Comparison of the Proposed Filter with Existing Filters.

	Kalman filter	Median filter	Proposed filter
Input noisy signal			
SNR (dB)	24.55	24.54	24.52
MSE	0.5778	0.577	0.5762
PSNR (dB)	50.51	50.518	50.52
From filtered output signal			
SNR (dB)	23.39	24.05	49.03
MSE	0.405	0.406	0.409
PSNR (dB)	51.05	51.04	52.0059

the improvement in SNR between the two signals. Table 7 shows that an improvement of about 50.01% of SNR is obtained from our proposed method.

5.4 Comparison of the Proposed Filter with Existing Filters

Table 8 shows the comparison of our proposed modified Wiener filter with the existing filters such as Kalman filter [15] and median filter [11]. From the table, we inferred that the output peak SNR (PSNR) of our modified Wiener filter is improved by 1.8% compared with that of the existing filters (Kalman filter and median filter). Similarly, the SNR of our proposed filter is improved compared with that of the existing filters. Although the noisy signal with same SNR and PSNR values is given as input to the three filters, the filtered output signal of our proposed filter has the better SNR and PSNR values than the rest of the filters.

6 Conclusion

In this paper, an FPGA-based denoising system for speech signal enhancement in digital hearing aids using Wiener filter and RFFT/RIFFT is proposed. The proposed modified Wiener filter and RFFT/RIFFT processor were used to reduce the power consumption and hardware requirement of the system. Matrix inversion using the given rotation method is proposed here, also used to reduce the computational complexity of the system. Our system, when coded, synthesized, and simulated in Xilinx ISE, exhibited efficient SNR values. The SNR analysis proved that our proposed system exhibits about 50.01% improvement in SNR compared with existing methods.

Bibliography

[1] V. Arunachalam and A. N. J. Raj, Efficient VLSI implementation of FFT for orthogonal frequency division multiplexing applications, *IET Circuits Devices Syst.* **8** (2014), 526–531.

[2] M. Ayinala, Y. Lao and K. K. Parhi, An in-place FFT architecture for real-valued signals, *IEEE Trans. Circuits Syst. II Express Briefs* **60** (2013), 652–656.

[3] A. Biswas, P. K. Sahu, A. Bhowmick and M. Chandra, Hindi phoneme classification using Wiener filtered wavelet packet decomposed periodic and aperiodic acoustic feature, *Comput. Electr. Eng.* **42** (2015), 12–22.

[4] R. Carvajal, K. Mahata and J. C. Aguero, Low complexity Wiener filtering in CDMA systems using a class of pseudo-noise spreading codes, *IEEE Commun. Lett.* **16** (2012), 1357–1360.

[5] J. Chen, J. Benesty, Y. Huang and S. Doclo, New insights into the noise reduction Wiener filter, *IEEE Trans. Audio Speech Lang. Process.* **14** (2006), 1218–1234.

[6] J. Cho, J. Kim and W. Cho, VLSI implementation of auto-correlation architecture for synchronization of MIMO-OFDM WLAN systems, *J. Semicond. Technol. Sci.* **10** (2010), 185–192.

[7] A. Ghassemi and T. A. Gulliver, PAPR reduction of OFDM using PTS and error-correcting code subblocking – transactions papers, *IEEE Trans. Wireless Commun.* **9** (2010), 980–989.

[8] A. Irturk, B. Benson, S. Mirzaei and R. Kastner, GUSTO: an automatic generation and optimization tool for matrix inversion architectures, *ACM Trans. Embedd. Comput. Syst.* **9** (2010), 32.

[9] H. Y. Jheng, Y. H. Chen, S. J. Ruan and Z. Qi, FPGA implementation of high sampling rate in-car non-stationary noise cancellation based on adaptive Wiener filter, in: *2011 IEEE/IFIP 19th International Conference on VLSI and System-on-Chip*, October 2011, pp. 114–117, IEEE, 2011.

[10] D. Kim and K. H. Park, Tiled QR decomposition and its optimization on CPU and GPU computing system, in: *2013 42nd International Conference on Parallel Processing*, October 2013, pp. 744–753, IEEE, Lyon, France, 2013.

[11] C. Lu, Reduction of musical residual noise using block-and-directional-median filter adapted by harmonic properties, *Speech Commun.* **58** (2014), 35–48.

[12] A. Malashri and C. Paramasivam, Low power and memory efficient FFT architecture using modified CORDIC algorithm, in: *2013 International Conference on Information Communication and Embedded Systems (ICICES)*, February 2013, pp. 1041–1046, IEEE, Chennai, India, 2013.

[13] Y. Maluenda and J. C. M. Bermudez, Transient mean-square analysis of prediction error method-based adaptive feedback cancellation in hearing aids, *IEEE Trans. Audio Speech Lang. Process.* **20** (2012), 261–275.

[14] A. Martinez-Leira, R. Vicen-Bueno, R. Gil-Pita and M. Rosa-Zurera, Acoustic feedback reduction based on FIR and IIR adaptive filters in ITE digital hearing aids, in: *International Conference on Audio, Language and Image Processing*, *ICALIP 2008*, July 2008, pp. 1442–1448, IEEE, Shanghai, China, 2008.

[15] W. Nabi, N. Aloui and A. Cherif, Speech enhancement in dual-microphone mobile phones using Kalman filter, *Appl. Acoust.* **109** (2016), 1–4.

[16] H. Nikmehr, B. Phillips and C. C. Lim, Fast decimal floating-point division, *IEEE Trans. VLSI Syst.* **14** (2006), 951–961.

[17] D. Sonawane and M. Sutaone, High throughput iterative VLSI architecture for Cholesky factorization based matrix inversion, *Int. J. Comput. Appl.* **35** (2011), 10–15.

[18] R. Talmon, I. Cohen and S. Gannot, Transient noise reduction using nonlocal diffusion filters, *IEEE Trans. Audio Speech Lang. Process.* **19** (2011), 1584–1599.

[19] P. Y. Tsai and C. Y. Lin, A generalized conflict-free memory addressing scheme for continuous-flow parallel-processing FFT processors with rescheduling, *IEEE Trans. VLSI Syst.* **19** (2011), 2290–2302.

[20] R. Vicen-Bueno, A. Martinez-Leira, R. Gil-Pita and M. Rosa-Zurera, Modified LMS-based feedback-reduction subsystems in digital hearing aids based on WOLA filter bank, *IEEE Trans. Instrum. Measure.* **58** (2009), 3177–3190.

[21] G. Wang, C. Li and L. Dong, Noise estimation using mean square cross prediction error for speech enhancement, *IEEE Trans. Circuits Syst. I Reg. Papers* **57** (2010), 1489–1499.

[22] M. Wu, B. Yin, G. Wang, C. Dick, J. R. Cavallaro and C. Studer, Large-scale MIMO detection for 3GPP LTE: algorithms and FPGA implementations, *IEEE J. Select. Topics Signal Process.* **8** (2014), 916–929.

[23] J. Yang, Y. Cai, Q. Zhou and W. Zhao, A selected inversion approach for locality driven vectorless power grid verification, *IEEE Trans. VLSI Syst.* **23** (2015), 2617–2628.

[24] A. Yasodai and A. V. Ramprasad, Noise degradation system using Wiener filter and CORDIC based FFT/IFFT processor, *J. Central South Univ.* **22** (2015), 3849–3859.

[25] C. Yu and M. H. Yen, Area-efficient 128-to 2048/1536-point pipeline FFT processor for LTE and mobile WiMAX systems, *IEEE Trans. VLSI Syst.* **23** (2015), 1793–1800.

Bionotes

Madam Aravind Kumar
Electronics and Communication Engineering, Grandhi Varalakshmi Venkata Rao Institute of Technology Engineering College, Bhimavaram, Andhra Pradesh, India,
aravindkumarm0883@gmail.com

Madam Aravind Kumar received a BTech degree from Acharya Nagarjuna University, Guntur, Andhra Pradesh, India, in 2005; received an MTech degree from JNTU-H, Hyderabad, Telangana, India, in 2010; and is pursuing a PhD degree from Gandhi Institute of Technology and Management (GITAM) University, Hyderabad, India. He is currently an associate professor in the Department of Electronics and Communication Engineering at Grandhi Varalakshmi Venkata Rao Institute of Technology Engineering College, Bhimavaram, A.P, India. His research interests are in very large scale integration design in low-power techniques, speech processing, image processing, and advanced signal processing.

Kamsali Manjunatha Chari
Electronics and Communication Engineering, Gandhi Institute of Technology and Management University, Hyderabad, Telangana, India

Kamsali Manjunatha Chari received a PhD from Jawaharlal Nehru Technological (JNT) University, Kakinada, and an MTech degree from JNT University, Hyderabad, and has an experience of more than 17 years in teaching and 3 years in industry. At present, Dr. Kamsali Manjunatha Chari is working with GITAM University, Hyderabad, Telangana, as a professor and head of the Electronics and Communications Engineering Department. He has published many papers in his field of interest viz. signal and image processing.