Design of an Adaptive Hearing Aid Algorithm using Booth-Wallace Tree Multiplier

Jitendra Kumar Das jkdas12@gmail.com

Asst. Professor, Dept. of Electronics and Telecommunication Engineering, Synergy Institute of Engineering & Technology, Dhenkanal, 759001, Orissa, India

Dr. K. K. Mahapatra kmaha2@rediffmail.com

Professor, Dept. of Electronics and Communication, NIT Rourkela, 769008, Orissa, India

Abstract

The paper presents FPGA implementation of a spectral sharpening process suitable for speech enhancement and noise reduction algorithms for digital hearing aids. Booth and Booth Wallace multiplier is used for implementing digital signal processing algorithms in hearing aids. VHDL simulation results confirm that Booth Wallace multiplier is hardware efficient and performs faster than Booth's multiplier. Booth Wallace multiplier consumes 40% less power compared to Booth multiplier. A novel digital hearing aid using spectral sharpening filter employing booth Wallace multiplier is proposed. The results reveal that the hardware requirement for implementing hearing aid using Booth Wallace multiplier is less when compared with that of a booth multiplier. Furthermore it is also demonstrated that digital hearing aid using Booth Wallace multiplier consumes less power and performs better in terms of speed.

Keywords: Booth Multiplier, Booth Wallace Multiplier, Adaptive Lattice Filte

1. INTRODUCTION

The decimation filter used in designing of a hearing aid has two major disadvantages which are

a. It requires more area for designing in FPGA.
b. As it is highly serial in nature, it requires more output latency.

Due to above reasons it consumes more power and we are specifically interested on lowering the power consumption of digital hearing aids. In this context we have used our own customized multiplier while maintaining the overall signal quality [6-7] to lower the power consumption.

Hearing impairment is often accompanied with reduced frequency selectivity which leads to a decreased speech intelligibility in noisy environments [2-5]. One possibility to alleviate this deficiency is the spectral sharpening for speech enhancement based on adaptive filtering [8-9] by which the intelligibility of the speech signal is maintained. Due to area constraints, such algorithms are usually implemented in totally time-multiplexed architectures, in which multiple operations are scheduled to run on a few processing units. This work discusses the power consumption in an FPGA implementation of the speech enhancement algorithm. It points out that power consumption can be reduced using Booth Wallace multiplier [11]. Several implementations of the algorithm, differing only in the degree of resource sharing are investigated aiming at power-efficiency maximization. At first an overview of the algorithm is given. Next the realized architectures using booth-Wallace tree multiplier are presented.

2. WALLACE TREE MULTIPLIER

Wallace trees are irregular in the sense that the informal description does not specify a systematic method for the compressor interconnections [10]. However, it is an efficient implementation of adding partial products in parallel [12]. The Wallace tree operates in three steps:

- Multiply: The multiplication is carried out using Booth Algorithm [11-14] which will generate n/2 partial product where n is number of bits of the multiplicand. The partial products are generated using the Booth recoding table given in table 1.

Mr_{i+1}	Mr	Mr_{i-1}	Recoded output
0	0	0	0
0	0	1	Y
0	1	0	Y
0	1	1	+2Y
1	0	0	-2Y
1	0	1	-Y
1	1	0	-Y
1	1	1	0

TABLE 1: Booth recoding table

- Addition: As long as there are more than 3 wires with the same weights add a following layer. Take 3 wires of same weight and input them into a full adder. The result will be an output wire of same weight. If there are two wires of same weight, add them using half-adder and if only one is left, connect it to the next layer.
- Group the wires in two numbers and add in a conventional adder. A typical Wallace tree architecture is shown in figure1 below. In the diagram AB0-AB7 represents the partial products.

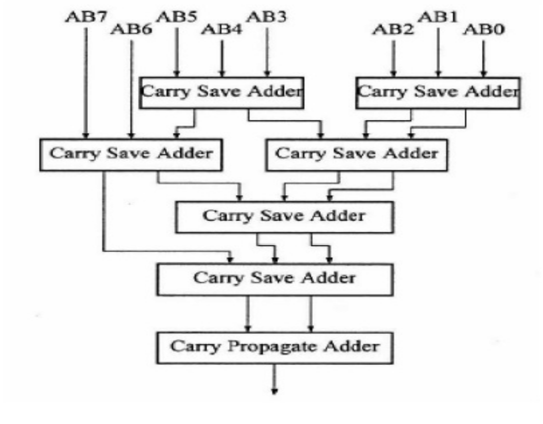

FIGURE 1: Wallace tree multiplier

Wallace multipliers consist of AND-gates, Carry Save Adders and a Carry Propagate Adder or Carry Look-ahead Adder.
The n-bit CSA consists of disjoint full adders (FA's). It consumes three-bit input vectors and produces two outputs, i.e., n-bit sum vector S and n-bit carry vector *C.* Unlike the normal adders [e.g. ripple-carry adder (RCA) and carry-look ahead adder (CLA)], a CSA contains no

carry propagation. Consequently, the CSA has the same propagation delay as only one FA delay and the delay is constant for any value of n. For sufficiently large n, the CSA implementation becomes much faster and also relatively smaller in size than the implementation of normal adders. In Wallace multiplier carry save adders are used, and one carry propagate adder is used as shown in the figure 2. The basic idea in Wallace multiplier is that all the partial products are added at the same time instead of adding one at a time. This speeds up the multiplication process.

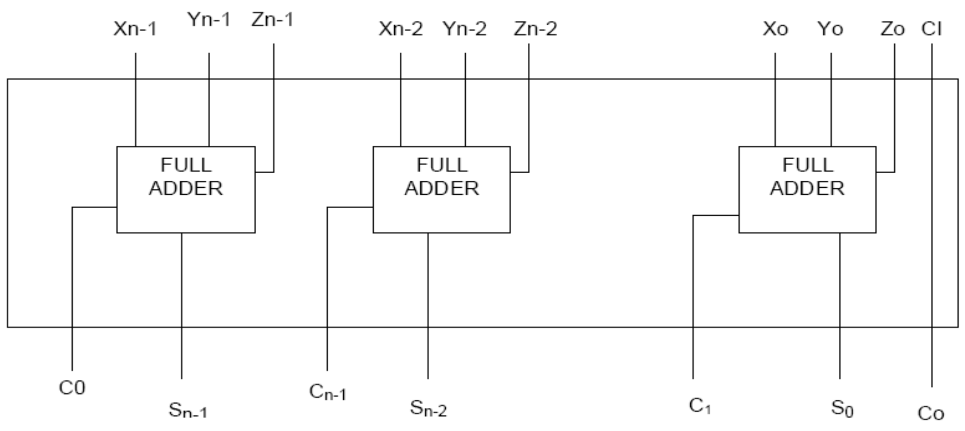

FIGURE 2: Implementation of n bit CSA operation

3. ADAPTIVE LATTICE FILTER

The adaptive lattice filter [6-7, 18-20] consists of three parts and these are

a. Adaptive decorrelator
b. Analysis filter (lattice filter)
c. Synthesis filter(lattice structure)

The Adaptive Decorrelator

An adaptive filter is a filter that adjusts its transfer function according to an optimizing algorithm. Because of the complexity of the optimizing algorithms, most adaptive filters are digital filters that perform digital signal processing and adapt their performance based on the input signal used.

The adaptive process involves the use of a cost function, which is a criterion for optimum performance so that the filter coefficients adjusted to minimize the cost on the next iteration [1]. The block diagram for such filter is presented in figure 3 that serves as a foundation for particular adaptive filter realizations, such as Least Mean Squares (LMS), Recursive Least Squares (RLS) or steepest descent algorithm etc. The idea behind the block diagram is that a variable filter extracts an estimate of the desired signal.

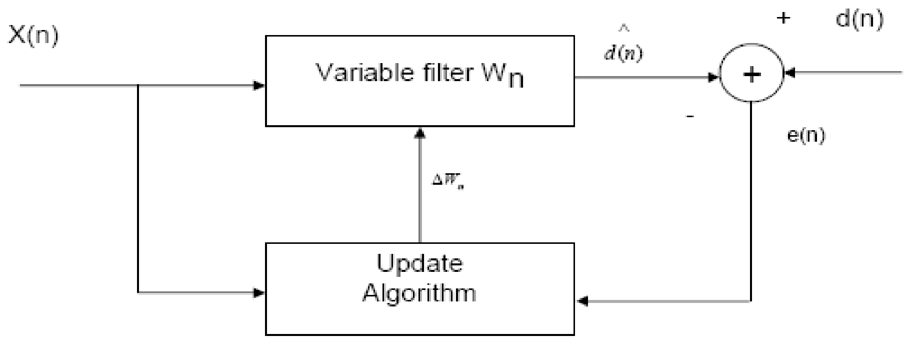

FIGURE 3: Block diagram of an Adaptive filter

The structure presented in figure 3 is described now. We take the following assumptions:

1. The input signal is the sum of a desired signal d(n) and interfering noise v(n)

$$x(n) = d(n) + v(n) \tag{1}$$

2. The variable filter has a Finite Impulse Response (FIR) structure. For such structures the impulse response is equal to the filter coefficients. The coefficients for a filter of order p are defined as

$$W_n(0)=[w_n(0),w_n(1),\ldots w_n(p)]^T \tag{2}$$

3. The error signal or cost function is the difference between the desired and the estimated signal.

$$e(n) = d(n) - \hat{d}(n) \tag{4}$$

The variable filter estimates the desired signal by convolving the input signal with the impulse response. In vector notation this is expressed as

$$d(n) = W^T(n)X(n) \tag{5}$$

where

$$x(n)=[x(n), x(n-1),\ldots\ldots\ldots,x(n-p)]^T \tag{6}$$

is an input signal vector. Moreover, the variable filter updates the filter coefficients at every time instant

$$W_{n+1}=W_n+ \Delta W_n \tag{7}$$

where Δw_n is a correction factor for the filter coefficients. The adaptive algorithm generates this correction factor based on the input and error signals. LMS and RLS define two different coefficient update algorithms. The speech signal to be transmitted is spectrally masked by noise. By using an adaptive filter, we can attempt to minimize the error by finding the correlation between the noise at the signal microphone and the (correlated) noise at the reference microphone. In this particular case the error does not tend to zero as we note the signal d(k) = x(k) + n(k) whereas the input signal to the filter is x(k) and n(k) does not contain any speech [2]. Therefore it is not possible to "subtract" any speech when forming e(k)=d(k)-d(n) . Hence in minimising the power of the error signal e(k) we note that only the noise is removed and e(k) =~ x(k).

Figure 4 depicts the structure of the adaptive gradient lattice decorrelator. The illustration shows three stages only with indices 1, i and m. good results typically require a filter order m where m can vary from 8 to 10 for speech sampled at 8 kHz. The output signal with vanishing autocorrelation is computed on the upper signal path by subtracting from the input sample suitable fractions of the signal values on the lower path. The multipliers $K_1,K_2\ldots\ldots\ldots K_m$ are iteratively computed as in equation 8.

$$K_i[n]:=K_i[n-1]+\Delta K_i[n] \tag{8}$$

At every sampling interval n. the details of this process are illustrated in figure 4 for the i-th stage. Input and output values on upper and lower signal path to and from the ith stage contribute to the computation of the update value ΔK_i.

Both output values are multiplied with the input values on the opposite path and then summed up to form the numerator in the subsequent computation of the update value. The denominator σ^2 is iteratively computed.

$$\sigma^2 := e.\ \sigma^2[n-1] + \Delta\ \sigma^2[n] \qquad (9)$$

The incremental value equals the sum of the two squared input values. The iterative computation of the denominator defines an exponentially decaying window which progressively decreases the influence of past contributions.

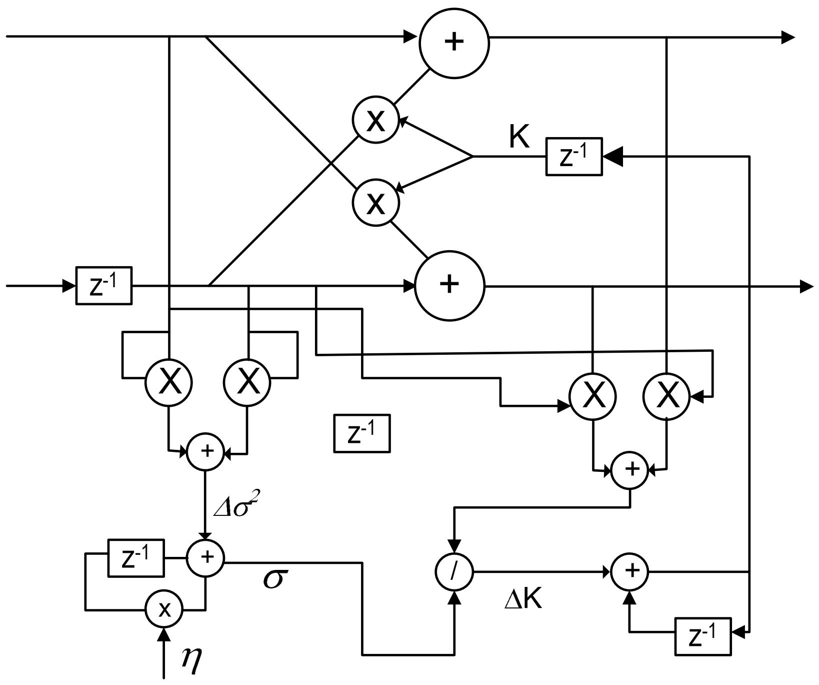

FIGURE 4: Adaptive gradient lattice filter

The computationally expensive division yields fast converging filter coefficients k_i independent of the varying input signal power level. This remarkable property is indispensable for good enhancement results. It is also clear contrast to simpler algorithms replacing the division by a multiplication with a small convergence constant $0<\mu<<1$. The longest delay through a string of lattice filters extends from the output of the storage element in the first lattice filter, through a multiplication with the first reflection coefficient, and then through an addition for each stage of the lattice filter until the output is produced in the final stage. For a large number of lattice filter stages, this longest delay can be reduced by a lattice filter optimization for speed which defers the final carry propagating addition until after the final lattice filter stage. This requires the transmission of an additional value between lattice filter stages. The multiplication process is speeded up using booth multiplier and the accumulation process is done faster using the Wallace multiplier.

The Analysis Filter

The analysis filter $H(z)=[1-A(z/\beta)]$ is illustrated in figure 5. Its structure [1, 8-9] is similar to that of the adaptive decorrelator shown in figure 4. The only difference is the multiplication with the filter parameter β following every shift element z^{-1} on the lower signal path. Furthermore, the analysis filter does not need a separate circuitry for coefficient update. It instead requires and therefore copies the filter coefficients K_1, K_2.....K_m computed by the unmodified (β =1) filter structure, i.e., the adaptive decorrelator.

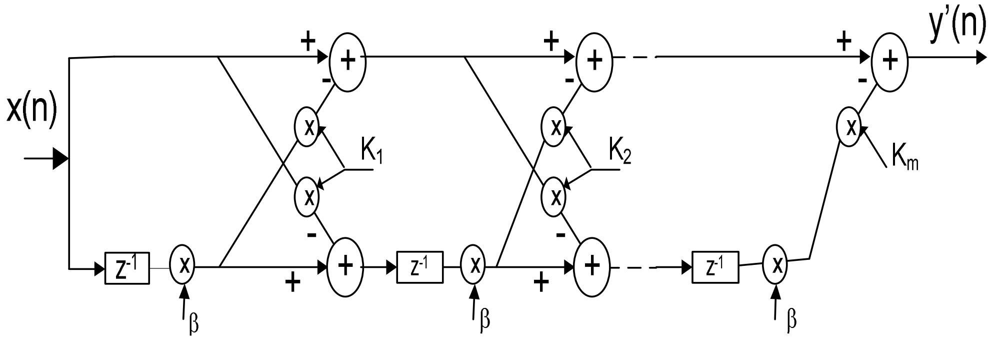

FIGURE 5: Analysis filter (1-A(z/□), x(n) is input and y'(n) is output

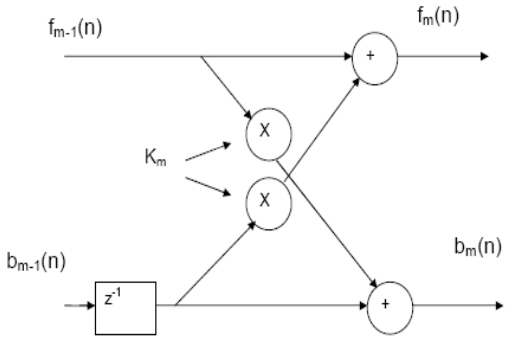

FIGURE 6: Single stage of analysis filter

The two mathematical equations for the single stage analysis filter as shown in figure 6 are

$$f_m(n) = f_{m-1}(n)-k_m b_{m-1}(n-1) \quad (10)$$

$$b_m(n) = b_{m-1}(n-1)-k_m f_{m-1}(n) \quad (11)$$

Some characteristics of the Lattice predictor are

- It is the most efficient structure for generating simultaneously the forward and backward prediction errors.
- The lattice structure is modular: increasing the order of the filter requires adding only one extra module, leaving all other modules the same.
- The various stages of a lattice are decoupled from each other in the following sense: The memory of the lattice (storing *b*0(*n* ¡ 1)*; : : : ; bm¡*1(*n* ¡ 1)) contains orthogonal variables, thus the information contained in *x*(*n*) is splitted in m pieces, which reduces gradually the redundancy of the signal.
- The similar structure of the lattice filter stages makes the filter suitable for VLSI implementation.

Lattice filters typically find use in such applications as predictive filtering, adaptive filtering, and speech processing. One desirable feature of lattice filters are their use of reflection coefficients as the filter parameter. Algorithms exist to compute reflection coefficients to obtain the optimal linear filter for a given filter order. Reflection coefficients have the additional property that for some applications, the optimal reflection coefficients remain unchanged when going from a lower order filter to a higher order filter. Thus, when adding additional filter stages, only the reflection coefficients for the added stages need to be computed. The modification with filter parameter □ causes the analysis filter to produce an output signal with reduced formants instead of a signal with completely flat spectral envelope as produced by the adaptive decorrelator.

The Synthesis Filter

When considering IIR filters, the direct form filter is the common structure of choice. This is true, in general, because when designing an algorithm which adapts the parameters a_k and b_k, the coefficients of the difference equation, described below, are manipulated directly.

$$Y_k+a_1Y_{k-1}+- - - - - - - - - - - - +a_mY_{k-m}=b_0U_k+b_1U_{k-1}+- - - - - - +b_mU_{n-m} \qquad (12)$$

Some problems exist in using the direct form filter for adaptive applications. First of all, ensuring stability of a time-varying direct form filter can be a major difficulty. It is often computationally a burden because the polynomial, A(z), made up of the a_k parameters, must be checked to see if it is minimum phase at each iteration. Even if the stability was assured during adaptation, round off error causing limit cycles can plague the filter. Parallel and cascade forms are often used as alternatives for direct form filters. These consist of an interconnection of first and second order filter sections, whose sensitivity to round off errors tends to be less drastic than for the direct form filter. Since the filter is broken down into a factored form, the round off error associated with each factorization only affects that term. In the direct form filter, the factors are lumped together so that round off error in each term affects all of the factors in turn.

A larger problem exists for both parallel and cascade forms: the mapping from transfer function space to parameter space is not unique. Whenever the mapping from the transfer function space to the parameter space is not unique, additional saddle points in the error surface appear that would not be present if the mapping had been unique. The addition of these saddle points can slow down the convergence speed if the parameter trajectories wander close to these saddle points. For this reason, these filter forms are considered unsuitable for adaptive filtering.

A tapped-state lattice form has many of the desirable properties associated with common digital filters and avoids the problems discussed above. Due to the computational structure, the round off error in this filter is inherently low.

Direct implementation of the IIR filter can lead to instabilities if it is quantized. The filter is stable using the following structure [1, 7-9, 14-16]. The structure of the synthesis filter $H(z)=[1-A(z/\gamma)]^{-1}$ is shown in figure 7. The synthesis filter also requires and copies the filter coefficients K_1, K_2,……K_m from the adaptive decorrelator at every sampling interval. The structure in figure 4.7 also shows a synthesis filter modified by the multiplication with the filter parameter γ succeeding every shift element z^{-1} on the lower signal path. The unmodified synthesis filter (γ=1) restores the original formants in the output when a signal with flat spectral envelope is fed to its input. The modification with a parameter value less than unity causes the synthesis filter to produce an output signal with partially restored formants only. The spectral sharpening effect results from a suitable choice of both filter parameters $0<\beta<\gamma<1$. Experiments with one adaptive filter only failed in producing satisfactory speech enhancement results.

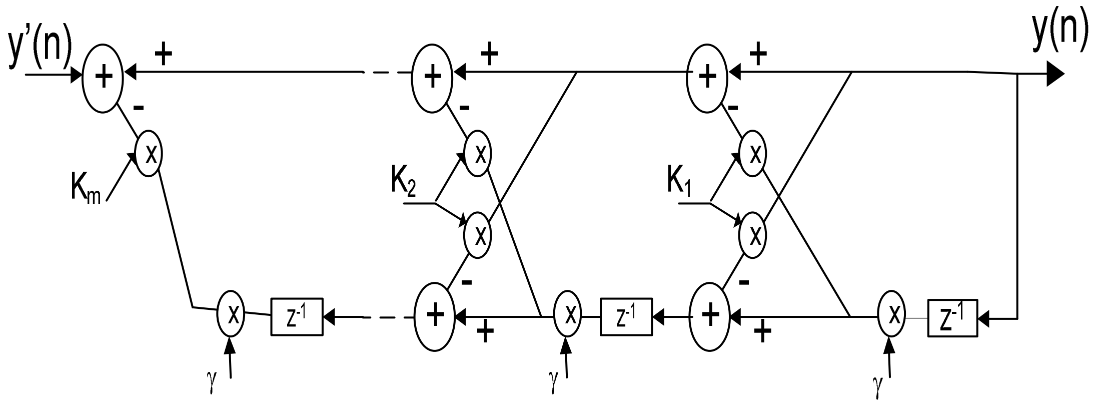

FIGURE 7: Synthesis filter, y'(n) input and y(n) is output

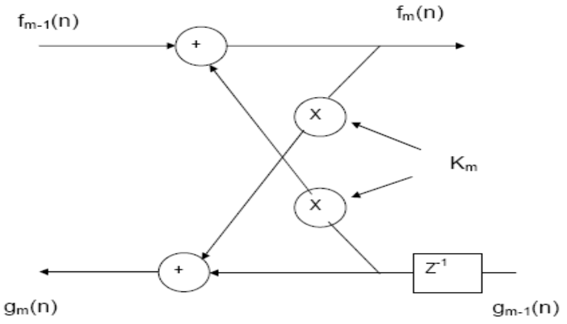

FIGURE 8: Single stage of sysnthesi filter.

The two mathematical equations for the single stage synthesis filter are shown below.

$$f_m(n) = f_{m-1}(n)-k_m b_{m-1}(n-1) \quad (13)$$

$$g_m(n) = g_{m-1}(n-1)-k_m f_{m-1}(n) \quad (14)$$

The computational complexity of a digital filter structure is given by the total number of multipliers and the total number of two input adders required for its implementation which roughly provides an indication of its cost of implementation. The synthesis filter is stable if the magnitudes of all multiplier coefficients in the realization are less than unity i.e. $-1<K_m<1$ for m=M, M-1, M-2, …1, 0.

4. HEARING AID DESIGN

4.1 Spectral Sharpening For Speech Enhancement

Speech enhancement usually results from adaptively filtering the noise reference signals and subsequently subtracting them from the primary input. However, a procedure for speech enhancement based on a single audio path is presented here. It is therefore applicable for real world situations. An example of such a situation is using hearing aid equipment. The hearing impaired person could place additional microphones close to noise sources only rarely. Current hearing aid equipment are used for filtering and amplifying the speech signal, this suggests that hearing impairment is just a more or less reduced sensitivity to sound pressure in various frequency intervals. This view however neglects the loss of frequency discrimination which can be efficiently compensated by the spectral sharpening technique presented. The idea of spectral sharpening originates from the adaptive post filtering method in modern speech coding schemes at bit rates around 8 kb/s and lower [1]. With these algorithms speech is encoded segment by segment. The linear prediction filter is any way computed in every speech segment for the encoding process as

$$A(z)=a_1 z^{-1} + a_2 z^{-2}+\dots+ a_m z^{-m} \quad (15)$$

and post- filtering with the transfer function

$$H(z) = \frac{1 - A(z/\beta)}{1 - A(z/\gamma)} \tag{16}$$

and constant filter parameters $0<\beta<\gamma<1$ is subsequently performed with a moderate computational increase.

Figure 9 shows the block diagram of spectral sharpening of speech sharpening [8-9] for speech enhancement. The speech signal x[n] from the microphone splits into three distinct paths. The signal on the lowest path passes through the analysis filter [1-A(z/β)] and subsequently through the synthesis filter $[1\text{-}A(z/\gamma)]^{-1}$. Both filters are implemented as lattice filters with the analysis and synthesis structures respectively. They both require the identical set of reflection coefficients K_1, K_2,.....K_m. where m represents the number of stages which is updated in every sampling interval by the adaptive decorrelator shown on the middle path of figure 4. The filter parameters β and γ do not vary with time.

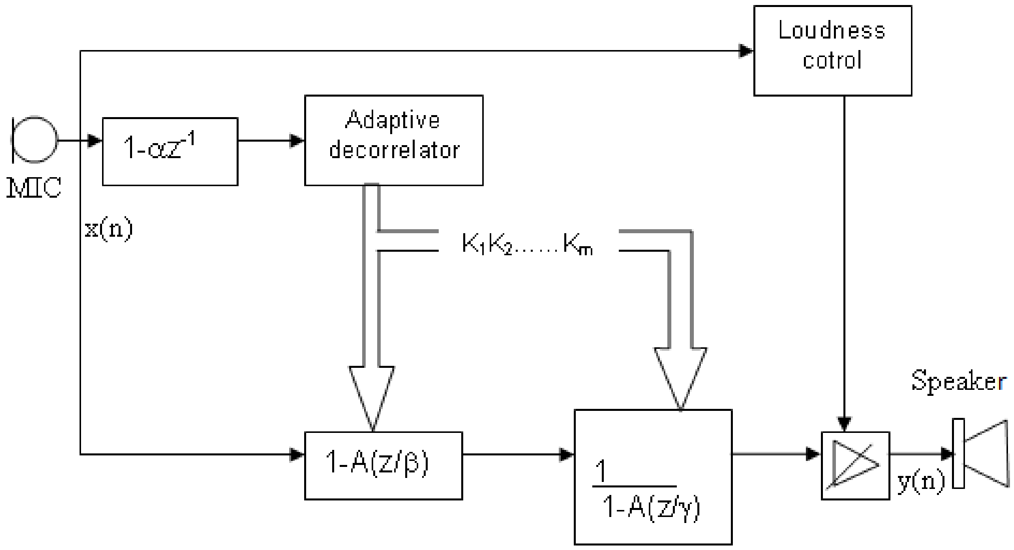

FIGURE 9: Block diagram of Spectral Sharpening for Speech Enhancement.

A high pass filter $1\text{-}\alpha z^{-1}$ is shown in front of the adaptive decorrelator, where x=1 may be chosen for simplicity. The high pass filter is used in order to compensate the spectral tilt of natural speech: the average power of the speech signal decreases above 1 KHz at a rate of ~ 10 db per octave. The adaptive transfer function in equation (16) enhances this spectral tilt even more when the filter coefficients K_1,K_2,........K_m are computed from the speech signal x[n] directly. Efficient speech enhancement requires however that the various formants are more or less uniformly emphasized, regardless of their relative power level. This is possible with the use of the high pass filter. It compensates at least partially the original spectral tilt.

The decorrelator on the middle signal path of the figure is an adaptive gradient lattice filter. It produces an output signal with vanishing autocorrelation by updating its filter coefficients in every sampling interval to the continuously changing input signal characteristics. The output signal is not required in this application, however. The updated filter coefficients K_1, K_2……K_m are of interest only for the use in the analysis and synthesis filter.

4.2 Spectral Sharpening For Noise Reduction

The block diagram of the spectral sharpening process for noise reduction is illustrated in figure 10. The arrangement of adaptive decorrelator, analysis and synthesis filters agrees with the previous block diagram in figure 9, however there various differences like

1. no loudness control,.
2. the input signal x[n] goes directly to the adaptive decorrelator, and

3. a high pass filter precedes the analysis and synthesis filters.

$$H_{hp}(z) = \frac{b(1-z^{-1})}{1-az^{-1}} \tag{17}$$

The reasons for these differences are as follows.
As mentioned in the previous section the spectral sharpening process

$$H(z) = \frac{1-a(z/\beta)}{1-a(z/\gamma)} \tag{18}$$

introduces a signal dependent amplification, signal segments with strong formant structure are amplified more than segments with a rather flat spectral envelop. In the sequel it is assumed that back ground noise is the major source for signal degradation and that its spectrum reveals relatively flat resonances only. Speech segments with strong resonances clearly profit in this situation. They experience a remarkable amplification compared to noisy segments. The loudness compensation of the previous block diagram is consequently omitted in order to preserve this effect.

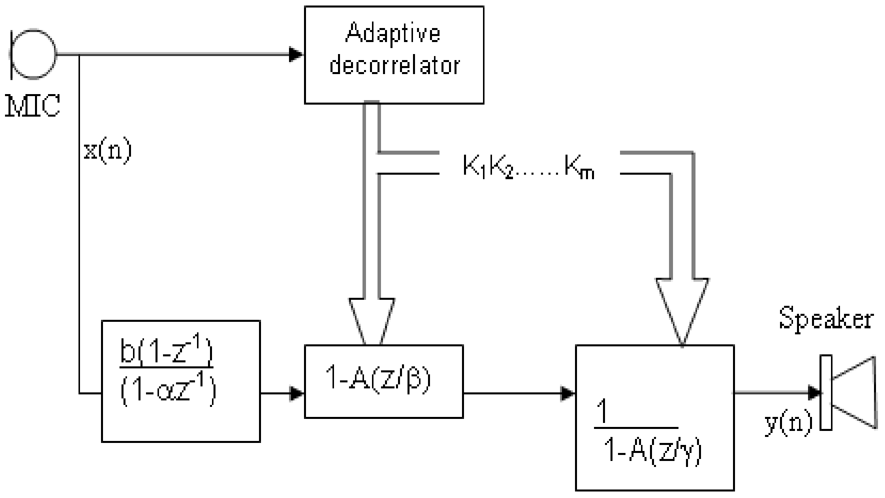
MIC
x(n)
Adaptive decorrelator
$K_1 K_2 \ldots\ldots K_m$
$\frac{b(1-z^{-1})}{(1-\alpha z^{-1})}$
$1-A(z/\beta)$
$\frac{1}{1-A(z/\gamma)}$
Speaker
y(n)

FIGURE 10: Block diagram of Spectral Sharpening by Noise Reduction

Best results require that the input signal is directly fed to the adaptive decorrelator. Only negligible amplification is then applied to noisy signal segments as a consequence of their assumed approximately flat spectrum. The spectral sharpening process further enhances the spectral tilt of speech when the filter parameters are estimated from the speech signal without prior compensation.
The high pass filter which preceded the adaptive decorrelator in the figure 9 has been shifted to the bottom signal path in figure 10 in order to avoid the scheme from producing a dull sound.

4.3 High Pass Filter

In signal processing, there are many instances in which an input signal to a system contains extra unnecessary content or additional noise which can degrade the quality of the desired signal. In such cases we may remove or filter out the useless samples. For example, in the case of the telephone system, there is no reason to transmit very high frequencies since most speech falls within the band of 700 to 3,400 Hz. Therefore, in this case, all frequencies above and below that band are filtered out. The frequency band between 700 and 3,400 Hz, which isn't filtered out, is known as the pass band, and the frequency band that is blocked out is known as the stop band. FIR, Finite Impulse Response, filters are one of the primary types of filters used in Digital Signal Processing [1, 14-16]. FIR filters are said to be finite because they do not have any feedback. Therefore, if an impulse is sent through the system (a single spike) then the output would invariably become zero as soon as the impulse runs through the filter.

There are a few terms that are used to describe the behaviour and performance of FIR filter. These are

- Filter Coefficients - The set of constants, also called tap weights, used to multiply against delayed sample values. For an FIR filter, the filter coefficients are, by definition, the impulse response of the filter.

- Impulse Response – A filter's time domain output sequence when the input is an impulse. An impulse is a single unity-valued sample followed and preceded by zero valued samples. For an FIR filter the impulse response of a FIR filter is the set of filter coefficients.

- Tap – The number of FIR taps, typically N, tells us a couple things about the filter. Most importantly it tells us the amount of memory needed, the number of calculations required, and the amount of "filtering" that it can do. Basically, the more taps in a filter results in better stop band attenuation (less of the part we want filtered out), less ripple (less variations in the pass band), and steeper roll off (a shorter transition between the pass band and the stop band).

- Multiply-Accumulate (MAC) – In the context of FIR Filters, a "MAC" is the operation of multiplying a coefficient by the corresponding delayed data sample and accumulating the result. There is usually one MAC per tap.

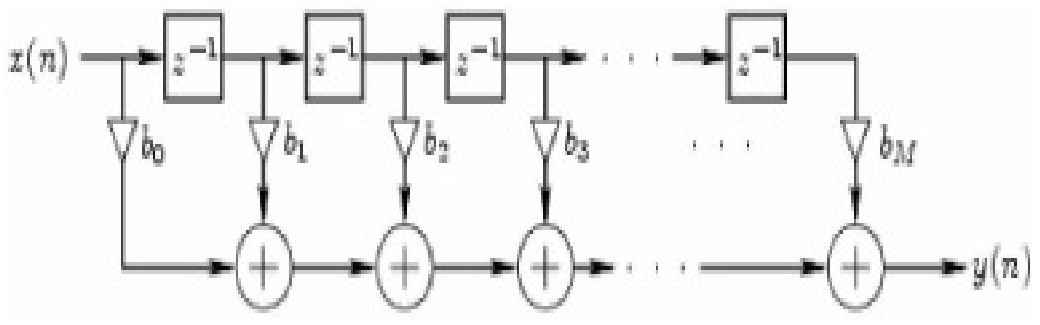

FIGURE 11: General causal FIR filter structure

Figure 11 gives the signal flow graph for a general finite-impulse-response filter (FIR). Such a filter is also called a transversal filter, or a tapped delay line. The implementation is one example of a direct-form implementation of a digital filter. The impulse response h(n) is obtained at the output when the input signal is the impulse signal □=[1 0 0 0…]. If the kth tap is denoted b_k, then it is obvious from figure 11 above that the impulse response signal is given by

$$h(n) = \begin{cases} 0, & n < 0 \\ b_n, & 0 \le n \le M \\ 0, & n > M \end{cases} \tag{19}$$

In other words, the impulse response simply consists of the tap coefficients, pretended and appended by zeros.

5. EXPERIMENTAL RESULTS AND CONCLUSION

5.1 Spectral Sharpening For Speech Enhancement In Matlab

Validation of the proposed filter was conducted using simulation tool. MATLAB v7.01 was used as platform. Speech signal constituting of 26000 samples at 8k samples/ sec was generated using wave recorder of windows and captured as a Matlab data file. This constitutes 3.25s of voice data. The waveform generated is presented in figure 12. The peak value of the signal generated is 275 mV. This signal formed the input to the hearing aid appliance. The output of a single stage is presented in figure 13 for β=0.04, γ=0.6 and μ=0.98. From the observation of the filter output it is seen that the output amplitude is nearly 600 mV. The single stage output with the filter parameters, β=0.4, γ=0.6 and μ=0.98 is presented in figure 14. In this case, peak amplitude is 390 mV which constitute a gain less than 2.

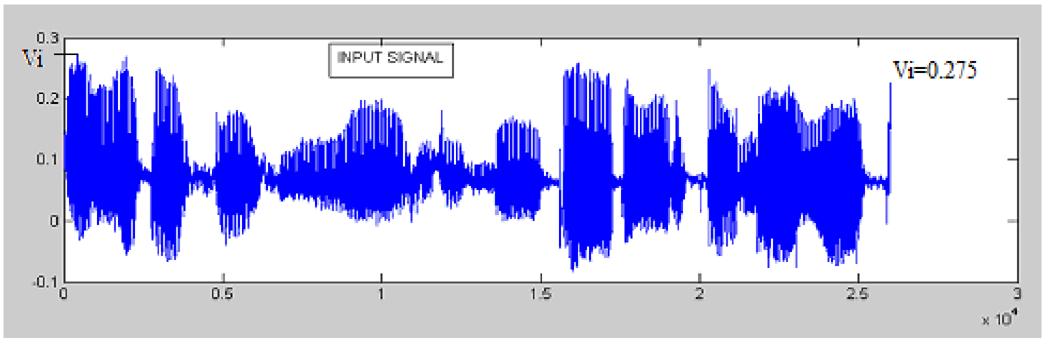

FIGURE 12: Waveform of the 3.25s speech input

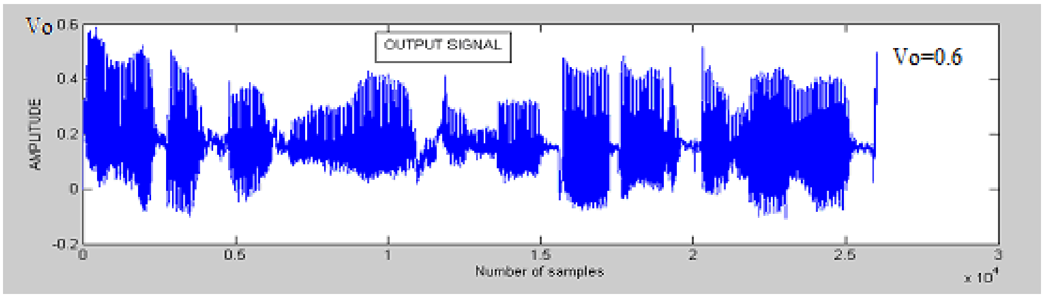

FIGURE 12: Waveform of the 3.25 second hearing aid output using parameters β=0.04,γ=0.6,μ=0.98

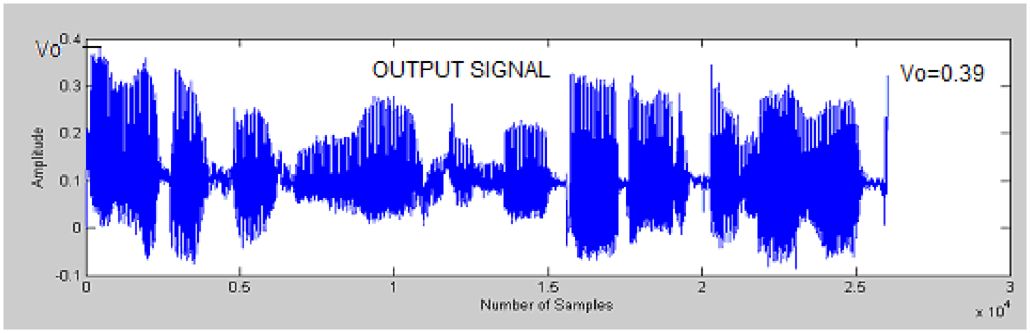

FIGURE 13: Waveform of the 3.25 second hearing aid output using parameters □=0.4,□=0.6,□=0.98

Following this; the performance of a 8 stage filter is observed. The filter output for β=0.04, γ=0.6, μ=0.98 and β=0.4, γ=0.6 and μ=0.98 are presented in figure 15 and figure 16 respectively. From figure 12 and figure 15, it is seen that output is more than double. Considering the superior performance of the 8 stage filter output over single stage filter, a 8 stage is used for hardware implementation.

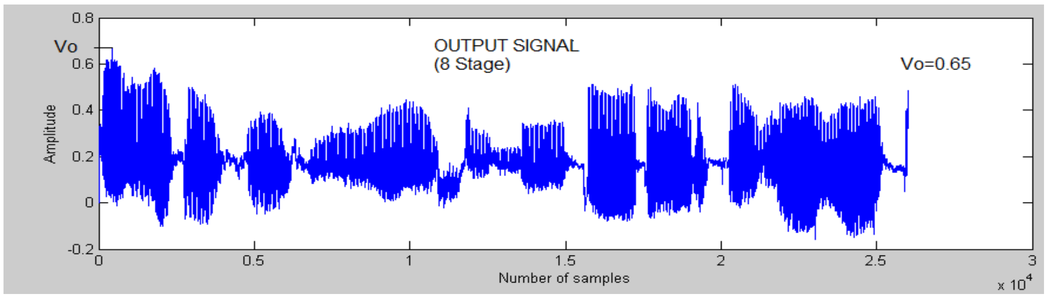

FIGURE 15: Waveform of the 3.25 second hearing aid output using parameters

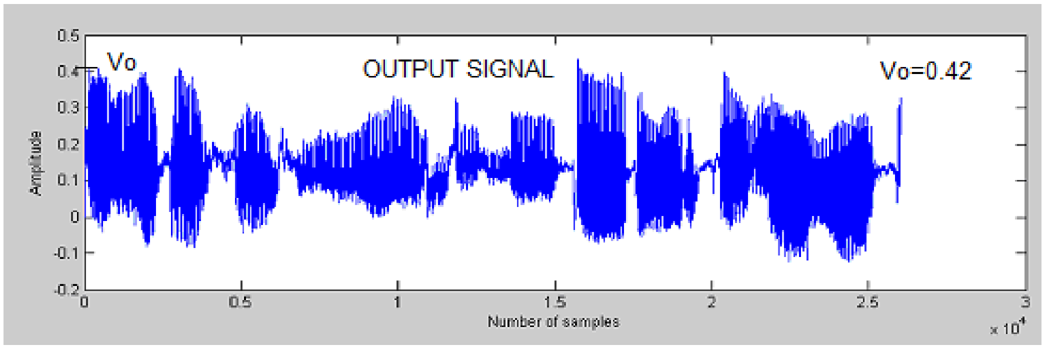

FIGURE 16: Waveform of the 3.25 second hearing aid output using parameters =0.4,=0.6,=0.98

5.2 FPGA Based Simulation Results.

MULTIPLIERS

The table below compares the cell usage of the three multipliers (SHIFT/ADD, BOOTH'S and BOOTH WALLACE multiplier) for 8 bit by 8 bit multiplication and 16 bit by 16 bit multiplication. From the table we can see that the booth Wallace multiplier uses less hardware compared to that of the shift/add multiplier and booth multiplier. The details are given table 2.

Cell Usage	**Shift/add multiplier (8x8)**	**Shift/add multiplier (16x16)**	**Booth multiplier (8x8)**	**Booth multiplier (16x16)**	**Booth Wallace multiplier (8x8)**	**Booth Wallace multiplier (16x16)**
BELS	240	1000	333	975	167	697
LUT-1	1	1	0	0	0	0
LUT-2	14	1	37	36	5	9
LUT-2	34	186	28	66	51	234
LUT-4	74	290	116	399	83	328
MUXCY	56	240	64	228	0	0
MUXF5	11	27	14	2	28	126
XORCY	49	225	61	219	0	0

TABLE 2: Cell used for the three Multipliers in virtex2p

Cells used	Slices	4-LUT	IO	Delay	Power consumption
Booth multiplier	109	192	32	24.41 ns	5 mW
Booth Wallace multiplier	76	139	32	20.62 ns	3 mW

TABLE 3: Power consumption and Delay for two multipliers with 8x8 bits

From the table 3 we can see that the using the booth Wallace multiplier consumes less power compared to the booth multiplier and also that booth Wallace multiplier is faster than booth multiplier. Hence the Booth Wallace multiplier [17-20] is used for hearing aid design in VHDL in this investigation. The table 4 gives area used by the two multipliers. From this it can be seen that the booth Wallace tree multiplier uses less hardware than other.

Cells used	Slices	Slice flip flops	4-LUT	Logic	Shift registers	IO
Booth multiplier	2684	183	5003	4979	24	32
Booth Wallace Multiplier	2583	196	4885	4866	19	32

TABLE 4: Cell usage for hearing aid component in virtex2p

5.3 Spectral Sharpening for Speech Enhancement in VHDL

The amplitude values of the speech signal sampled at 8 kS/s is rounded to 8 bits and stored in a text file for VHDL simulation. The hearing aid is designed in VHDL and is tested using different multipliers. The first 250 samples are taken as input for the hearing aid in VHDL .The output obtained through simulation is stored in a text file. The text file is read in MATLAB and is plotted as shown in the figure 18. The parameters used in VHDL are β=0.04, γ=0.6, μ=0.98.

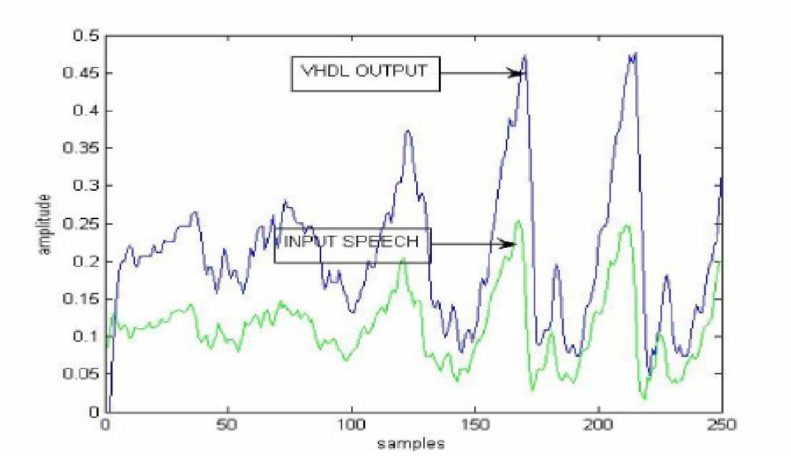

FIGURE 14: Comparision of input speech signal with output using vhdl for 250 samples

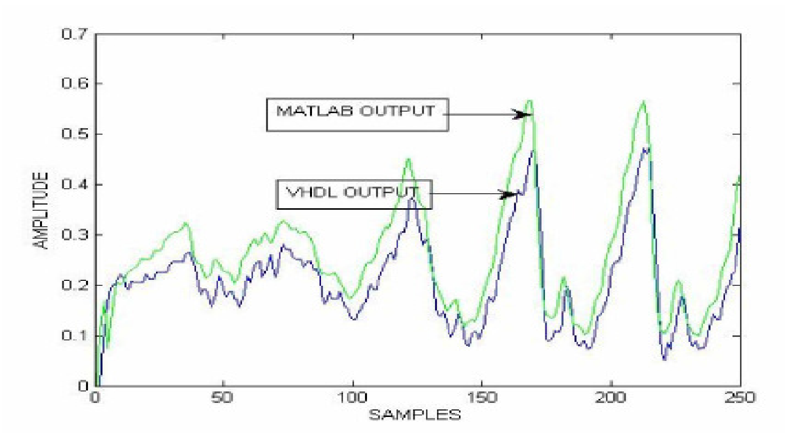

FIGURE 15: Comparison of the MATLAB output speech signal with the output obtained using VHDL for 250 samples

From the figure we can see that the output obtained using VHDL is slightly less in magnitude than MATLAB output. This is due to rounding of the values and due to fixed point multiplication. But from the figure 19 we see that the VHDL output follows the MATLAB output. The table 5 below shows the resource utilization summary and power consumed by the two designs using different multipliers.

Design name	**Slice used out of 3008**	**4-LUTs used out of 6016**	**Slice FFs used out of 6016**	**Shift registers**	**Logics**	**IO used out of 348**	**Power consumption**
High pass filter using Booth multiplier	241 (7%)	446 (7.4%)	41 (<1%)	-	-	25 (7%)	-
Synthesis filter	175 (5.8%)	323 (5.3%)	32 (<1%)	-	-	41 (11%)	-
Decorrelator	164 (5.4%)	292 (4.9%)	104 (1.7%)	-	-	40 (11%)	-
Hearing aid using Booth multiplier	2684 (89%)	5003 (83%)	183 (3%)	24	4979	32 (9.5%)	40 mW
Hearing aid using Booth Wallace multiplier	2583 (85%)	4885 (81%)	196 (3.3%)	19	4866	33 (9.5%)	30 mW

TABLE 5: FPGA resources used and power of hearing aid design

6. Conclusions

All the papers referred so far do not convey any information about the power consumption and only architectural part is discussed. Our emphasis is to design an adaptive algorithm based on Booth-Wallace tree multiplier which consumes less power with respect to the use of Booth multiplier suitable for hearing aid application.. with this effort we the whole system in figure 10 is implemented using Booth-Wallace tree multiplier and power calculation of the whole system is done using Xilinx XPower Analyser. From the figure 19 it can be seen that our design output matches the Matlab output. Also referring to table 5 we can see that the power consumed by the hearing aid with booth Wallace tree multiplier is less than the hearing aid using booth multiplier which is about 25% lesser than the latter. So we can conclude that the hearing aid using booth Wallace tree multiplier consumes less power in this case.

6. REFERENCES

[1] S. K. Mitra. *"DSP A Computer based Approach".* Tata McGraw Hill Publication, 2nd Edition, 2001

[2] S. Ramamoha, S. Dandapat. *"Sinusoidal Model based Analysis and Classification of Stressed Speech".* IEEE Trans. On Speech, Audio and Language Processing, 14(3):737- 746, 2006

[3] R. Dhiman, A. Kumar and R. Bahl. *"Design of low power multi-DSP module for acoustic signal processing on remote platforms".* Proc. Intl. Conf. on Sonar – Sensors and Systems, Kochi, 2002

[4] A. Kumar, S. K. Mullick. *"Nonlinear dynamical analysis of speech".* Journal of Acoustical Society of America, 100(1):615-629, 1996

[5] V. Venugopal, K.H. Abed and S.B. Nerurkar. *"Design and implementation of a Decimation Filter for hearing aid applications".* Proceedings of IEEE Southeast Con: 2005

[6] Edwards, B. D. *"Signal Processing Techniques for a DSP Hearing Aid".* IEEE International Symposium on Circuits and Systems, 6:586 – 589,1998

[7] O. O. Khalifa, M.H Makhtar, M.S. Baharom. *"Hearing Aids System for Impaired People".* International journal of computing and Information science,2(1):23-26, 2004

[8] F. Buergin, F. Carbognani and M. Hediger. *"Low Power Architecture Trade-offs in a VLSI Implementation of an Adaptive Hearing Aid Algorithm".* IEEE, 2006

[9] A. Schaub and P. Straub. *"Spectral sharpening for speech enhancement/noise reduction".* In IEEE Acoustics, Speech and Signal Processing (ICASSP-91)1991

[10] Wallace, C.S. *"A suggestion for a fast multiplier".* IEEE Trans. Electron. Compute, EC-13:14-17, 1964

[11] A. D. Booth. *"A signed binary multiplication technique".* Quart. J. Math., IV(2):1951

[12] L. P. Rubinfield. *"A proof of the modified booth's algorithm for multiplication".* IEEE Trans. Computers, 37: 1988.

[13] M.J.Liao, C.F.Su, Chang and A. Wu. "A carry select adder optimization technique for high-performance Booth-encoded Wallace tree multipliers". IEEE International Symposium on Circuits and Systems, 2002.

[14] J. Wassner, H. Kaeslin, N. Felber, W. Fichtner. *"Waveform coding for low-power digital filtering of speech data".* IEEE Transactions on Signal Processing, 51(6):1656– 1661, June 2003

[15] Regalia, Phillip A. *"Adaptive IIR Filtering in Signal Processing and Control".* Marcel Dekker, Inc., 1995

[16] M. Nayeri, W. Jenkins. *"Alternate realizations of adaptive IIR filters and properties of their performance surfaces".* IEEE Transactions on Circuits and Systems, 36:485-496, 1989

[17] Charles H Roth. "*Digital system design using VHDL*". Jr. Thomson Brooks/Cole

[18] J. A. Maxwell, P. M. Zurek, *“Reducing acoustic feedback in hearing aids”.* IEEE Trans. Speech Audio Processing, 3:304–313, 1993

[19] Sankarayya, N., Roy, K., and Bhattacharya, D. *“Algorithms for Low-Power and High-Speed FIR Filter Realization Using Differential Coefficients”.* IEEE Trans. On Circuits and Systems, 44(6):488-497, 1997

[20] S. Haykin. *“Adaptive Filter”.* PHI Publication