A Novel Adaptive Volterra Filter to Compensate for Speaker Non-linearity

Tonmoy Roy*, Parvez Ahmmed, Md. Shafiur Rahman, Abdullah Ash-Saki and A. B. M. Harun-Ur-Rashid
Department of Electrical and Electronic Engineering
Bangladesh University of Engineering and Technology, Dhaka–1205, Bangladesh
Email: *tonmoy_roy@live.com

***Abstract*—The implementation of an adaptive Volterra filter to compensate for speaker non-linearity using a pipelined recurrent neural network based architecture is demonstrated. The proposed architecture consists of two stages: nonlinear stage performing a nonlinear second order Volterra (SOV) mapping from the input space to an intermediate space and a linear combiner performing a linear mapping from the intermediate space to the output space. The filter design is tested by implementing it on a dataplane processing unit configured for audio processing. The implemented algorithm is adjusted for Xtensa Processor and uses HiFi-2 DSP standard. The collected data confirms smaller error of the output with the speaker and also a very low settling time.**

***Index Terms*—Volterra, JPPSOV, Xtensa, adaptive filter, non-linearity.**

I. Introduction

The Volterra Series filter is one of the most widely used non-linear systems. The ease with which existing linear adaptive filters can be extended to fit this model makes it a popular non-linear filter. The Volterra Series expansion can be interpreted as a Taylor Series expansion with memory [1]. A major disadvantage of the filter is the computational complexity arising when the complete series has to be implemented. One key reason is that the number of the Volterra coefficients increases geometrically as the delays (or memory) and orders increases.

There are a number of methods applied to reduce the computational complexity. These include iterative factorization technique [2], the fast Kalman algorithm [3], multi memory decomposition [4]. With these strategies, the computational complexity can be reduced significantly compared with the SOV filter, but the stable performance is not guaranteed. Again, in Volterra filter using the normalized least mean square (NLMS) the output of the system becomes a nonlinear function of the filter coefficients [5].

However, a novel architecture based on Pipelined Recurrent Neural Network (PRNN) [6], [7] provides a more computationally efficient way to approximate non-linear systems in a modular approach. In this Xtensa processor [8] based implementation, the novel adaptive Joint Process filter using Pipelined Second Order Volterra (JPPSOV) architecture is followed. The attractive attribute of such pipelined architecture is that the capability of such realizations to efficiently approximate nonlinear systems using less computational burdens. Moreover, the combination of the two stages, nonlinear and then linear, makes the output dependent on the filter coefficients linearly.

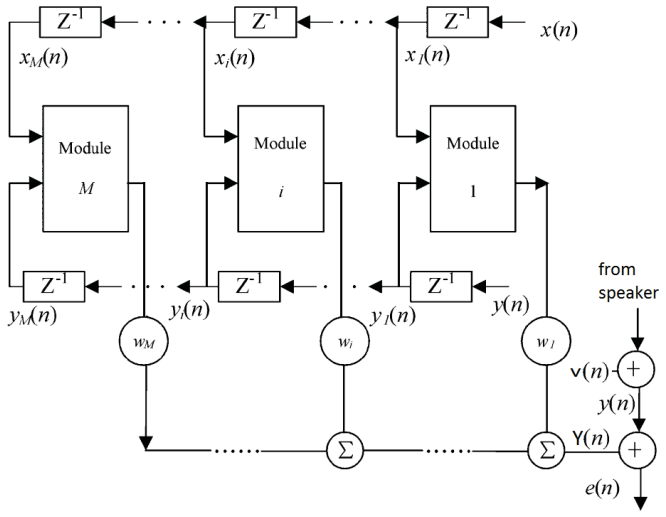


Fig. 1. Block diagram of modified adaptive Joint Process filter using Pipelined Second Order Volterra architecture.

II. Theory

The algorithm effectively splits the system into two subsystems, one is nonlinear and the other is linear. The nonlinear subsystem contains M number of SOV cores (Fig. 1). It performs nonlinear mapping from a input space to an intermediate space which linearizes the nonlinear signals. Each module takes one of the external inputs and the noisy output from the speaker as inputs. The benefit of this arrangement is in the uniformity of numbers of inputs and outputs and the synaptic weight matrix.

For the i-th module, the input can be written as:

$$X_i(n) = [X_{i,1}(n), X_{i,2}(n)] \tag{1}$$

where, the column $X_{i,1}(n)$ is vector of previous external input at the n-th time point and is described by:

$$X_{i,1}(n) = x(n-i)$$

and $X_{i,2}(n)$ contains the noisy output from the speaker,

$$X_{i,2}(n) = y(n-i)$$

At the n-th time point, input signal vector $X_i(n)$ is expanded to a row vector $XX_i(n)$ by the SOV series (2) where the number of columns of the matrix is defined by $L = 6$ which is much less compared to $(N+1)(N+2)/2$ of the ordinary Volterra theory.

978-1-4799-4166-7/14/$31.00 ©2014 IEEE

$$\begin{aligned} XX_i(n) &= [XX_{i,1}(n), XX_{i,2}(n), \ldots, XX_{i,L}(n)] \\ &= [1, X_{i,1}(n), X_{i,2}(n), X^2_{i,1}(n), \\ &\quad X_{i,1}(n)X_{i,2}(n), X^2_{i,2}(n)] \end{aligned} \tag{2}$$

Since the weight vectors are same for all the modules, the synaptic weight vectors of the SOV series of each module is denoted by the column vector $H(n)$, whose length is also L.

$$H(n) = [h_1(n), h_2(n), ..., h_L(n)]^T \tag{3}$$

Thus, the output $y_i(n)$ of the i-th modules is defined by:

$$y_i(n) = XX_i(n)H(n) \tag{4}$$

The linear subsystem simply performs a mapping from the intermediate space to the output. The outputs of the nonlinear subsystems are combined in a ratio provided by a weight vector to produce an estimation of the original signal. The weight vector is given by:

$$W(n) = [w_1(n), w_2(n), ..., w_M(n)] \tag{5}$$

If the outputs of the previous non-linear modules are

$$YY(n) = [y_1(n), y_2(n), ..., y_M(n)]^T \tag{6}$$

the outputs of the linear subsystem would simply be the inner products of these two vectors.

$$Y(n) = W(n)YY(n) \tag{7}$$

The error between the desired signal and estimated signal is given by

$$e(n) = y(n) - Y(n) \tag{8}$$

Under the assumption that the coefficient vector $H(n)$ of the nonlinear subsection is time variant, the adaptation of $H(n)$ is similar to the classical case. Therefore, the normalized least mean square (NLMS) equation driving the update of the nonlinear subsection finally becomes

$$H(n+1) = H(n) + \eta_2 e(n) \frac{U(n)}{||U(n)||_2^2} \tag{9}$$

where $U(n)$ is defined by,

$$U(n) = W(n)XX_i(n)$$

Similarly, the NLMS equation driving the update of the linear filter is

$$W(n+1) = W(n) + \eta_1 e(n) \frac{YY(n)}{||YY(n)||_2^2} \tag{10}$$

where $||YY(n)||_2^2$ is the L_2 norm.

Here, η_1 and η_2 are the two learning rates. A necessary condition for the mean convergence is [9],

$$0 < \eta_1 < 2 \qquad \text{and} \qquad 0 < \eta_2 < 2$$

The values of the learning rates are very crucial for steady response of the filter. In this work, $\eta_1 = \eta_1 = 1/8$ is used.

TABLE I
THE VALUE OF THE NON-LINEARITY COEFFICIENTS FOR SIMULATING DIFFERENT SPEAKERS

Co-efficient	b	c
Speaker-1	-0.1	0.1
Speaker-2	0.05	0.05

III. Optimization Techniques

As illustrated in [10], the filter with only 4-5 modules ensures lowest mean square error (MSE). As the Xtensa processor gives an opportunity of two parallel multiplication operation, 4 modules are used in this implementation. Moreover, the inputs of each module is also modified in order to decrease the no. of memory operations.

According to (9), $U(n)$ has to be divided by $||U(n)||_2^2$ and then η_1 is multiplied to obtain the change of the vector H. The filter converges as long as $0 < \eta_1 < 2$. The performance of the filter does not change much within this range of η_1. If η_1 is kept constant on the other hand, the division result can be allowed to vary to mimic the change of η_1. Since doing complete division using any DSP processor like Xtensa is inefficient, the division is replaced with an efficient bit shift operation. This bit shift can be modeled by,

$$q = \left\lfloor \frac{2^{\lfloor \log_2 a \rfloor}}{b} \right\rfloor \tag{11}$$

where a is the dividend, b is the divisor and q is the quotient. It can be observed that the quotient can be off by only one fourth of its intended value. Since $\eta_1 = 1/8$, the division can be replaced with the bit shift operation, effectively making $\eta_1 = 1/32$ in the worst case scenario.

It can be seen from (10) that, the computation of the change of the vector W can be done in a similar manner to improve efficiency.

IV. Test Strategy

The Xtensa instruction set simulator (ISS) has been used to test the performance of the algorithm. From the tests run in the simulator, various performance characteristics like the time required to process each sample and the residual error are obtained.

The audio for the speaker input and the audio output obtained from the microphone both are required during runtime for the filter. Both the audios are assumed to be mono channel with 16 kHz sampling rate and each sample was assumed to consist of 16 bits. The errors in output are obtained as 16-bit values as well.

For the actual testing of the adaptive filter algorithm, the microphone input signal is modeled from the speaker input signal. Furthermore the speaker input is considered to be nonlinearly distorted by,

$$z = \frac{x + bx^2 + cx^3}{1 + |b| + |c|} \tag{12}$$

where the non-linear terms are assumed to be less than or equal 10% of the linear term. A random noise has also been added to the microphone input. The noise is within 1% of the feedback signal at most.

Using the speaker input and the modeled microphone input, the filter is tested for various different scenarios. Three different audio samples each 1 second in length are used to test the

TABLE II
THE REQUIRED CYCLES & MINIMUM PROCESSOR CLOCK SPEED FOR THE DIFFERENT SCENARIOS

Audio type	Speaker	Total cycles	Cycles per sample	Minimum clock speed (MHz)
Human voice	Speaker-1	17238241	1077	17.24
	Speaker-2	17238871	1077	17.24
Instrumental music	Speaker-1	17564397	1098	17.56
	Speaker-2	17642333	1103	17.64
Periodic signal	Speaker-1	17995837	1125	18.00
	Speaker-2	18049309	1128	18.05

TABLE III
THE SETTLING TIME AND THE RESIDUAL NOISE AT DIFFERENT SCENARIOS

Audio type	Speaker	Residual Noise (dB)	Settling Time (ms)
Human voice	Speaker-1	-41.0048	8.75
	Speaker-2	-39.0622	7.50
Instrumental music	Speaker-1	-28.7217	12.50
	Speaker-2	-27.9533	9.38
Periodic signal	Speaker-1	-45.1022	32.19
	Speaker-2	-44.0620	25.00

filter. These audios are: human voice, instrumental music and simple triple frequency audio. All of the audios are of 1 second in duration. Different values of the non-linearity coefficients in the microphone input model are tested to simulate different speaker types. The used coefficients, b and c, are summarized in Table I.

The processor has been configured in the built-in 'HiFi-2 standard' mode [11] for the particular implementation. In this configuration, the processor can handle audio signal processing like multiplication and addition of 24-bit numbers very efficiently. Hence for this implementation, the 16-bit integers from the input are shifted left in order to convert to 24-bit. Processor's parallel multiplication instruction is utilized whenever possible to reduce the number of cycles required per audio sample.

V. RESULTS AND DISCUSSION

The number of instructions required by the processor to operate on the audio samples of 1 second and minimum clock frequency required to do the processing in real-time is given in Table II. Since the processor can operate at frequencies well above the obtained minimum required clock frequency, it will be possible to use the processor for real-time operation with ease. As illustrated in Fig. 2, use of only four modules shows the lowest Mean Square Error (MSE).

The average error in decibels was calculated with respect to the input signal. As the settling time is negligible, the residual noise was calculated as the average of all the error values in decibel (dB). The time after which the error becomes less or equal to the mean error was considered to be the settling time. The mean local error vs. sample plots for the human voice sample, music sample and the simple periodic audio signals through different speakers are shown in Fig. 3 and Fig. 4 respectively. Again, the settling time and the residual noise is presented for the six scenarios in Table III.

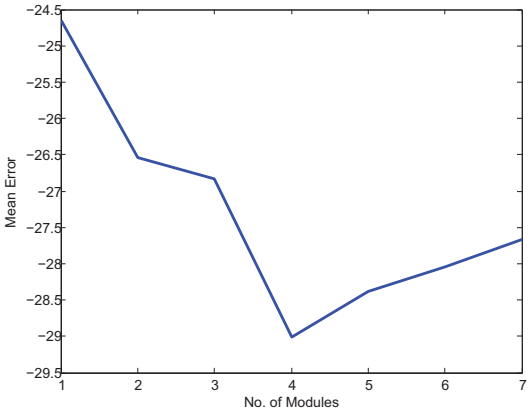


Fig. 2. relation between the MSE and the number of modules for speaker-1.

VI. CONCLUSION

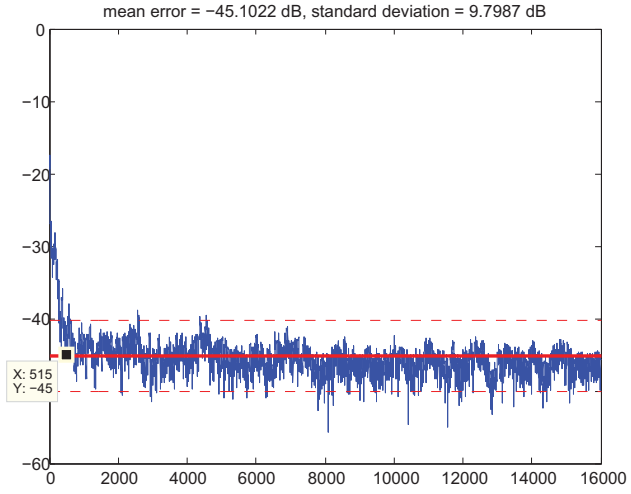
An implementation of a novel adaptive joint process filter employing the pipelined architecture is presented in this paper. It is seen that the algorithm can easily be implemented using the Tensilica Xtensa dataplane processing unit. The obtained residual error was very low and the settling time was very small. Furthermore, size of the code and the required processor clock speed was within desired levels for the implementation indicating the efficiency of the algorithm.

ACKNOWLEDGMENT

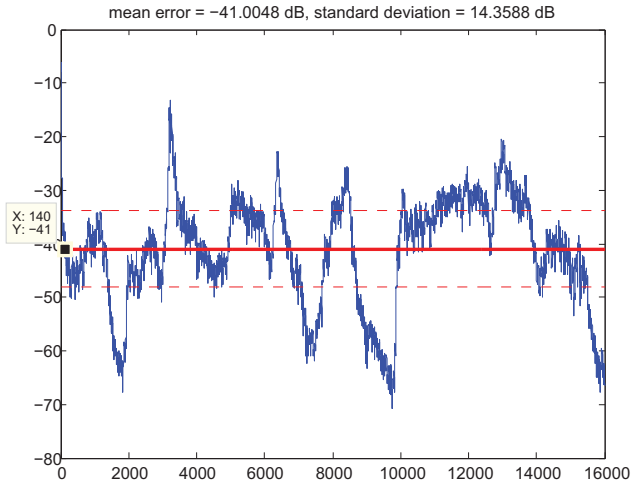
The authors would like to thank Cadence Design Systems, India for providing the software, license and samples required for the work.

REFERENCES

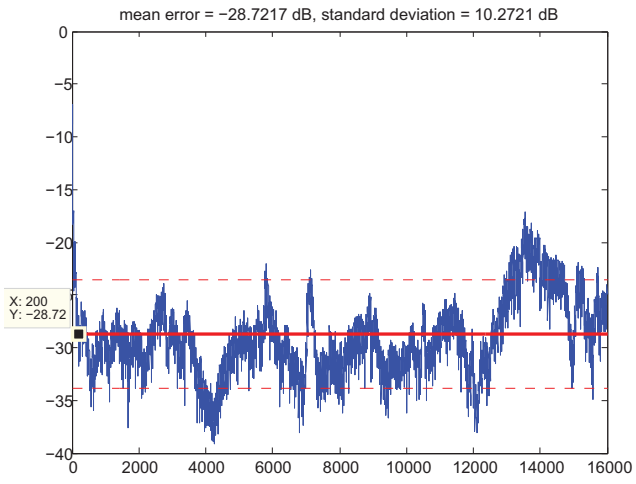
[1] V. Mathews, "Adaptive polynomial filters," *IEEE Signal Processing Magazine*, vol. 8, no. 3, pp. 10–26, July 1991.
[2] T. Koh and E. Powers, "Second-order volterra filtering and its application to nonlinear system identification," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 33, no. 6, pp. 1445–1455, Dec 1985.
[3] C. Davila, A. Welch, and I. Rylander, H., "A second-order adaptive volterra filter with rapid convergence," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 35, no. 9, pp. 1259–1263, Sep 1987.
[4] Y. Lou, C. Nikias, and A. Venetsanopoulos, "Efficient VLSI array processing structures for adaptive quadratic digital filters," *Circuits, Systems and Signal Processing*, vol. 7, no. 2, pp. 253–273, 1988. [Online]. Available: http://dx.doi.org/10.1007/BF01602100
[5] T. Harada, M. Muneyasu, and T. Hinamoto, "A pipeline architecture of quadratic adaptive volterra filters based on NLMS algorithm," in *The IEEE International Symposium on Circuits and Systems, ISCAS*, vol. 2, May 2001, pp. 785–788.
[6] S. Haykin and L. Li, "Nonlinear adaptive prediction of nonstationary signals," *IEEE Transactions on Signal Processing*, vol. 43, no. 2, pp. 526–535, Feb 1995.
[7] L. Li and S. Haykin, "A cascaded recurrent neural network for real-time nonlinear adaptive filtering," in *IEEE International Conference on Neural Networks*, vol. 2, 1993, pp. 857–862.
[8] (2014) Xtensa customizable processors - Cadence IP. [Online]. Available: http://ip.cadence.com/ipportfolio/tensilica-ip/xtensa-customizable
[9] S. Haykin, *Adaptive Filter Theory*. Englewood Cliffs, NJ: Prentice Hall, 2002.
[10] H. Zhao and J. Zhang, "A novel adaptive nonlinear filter-based pipelined feedforward second-order volterra architecture," *Signal Processing, IEEE Transactions on*, vol. 57, no. 1, pp. 237–246, Jan 2009.
[11] (2014) Tensilica HiFi audio voice DSP IP. [Online]. Available: http://ip.cadence.com/ipportfolio/tensilica-ip/audio



(a) periodic signal

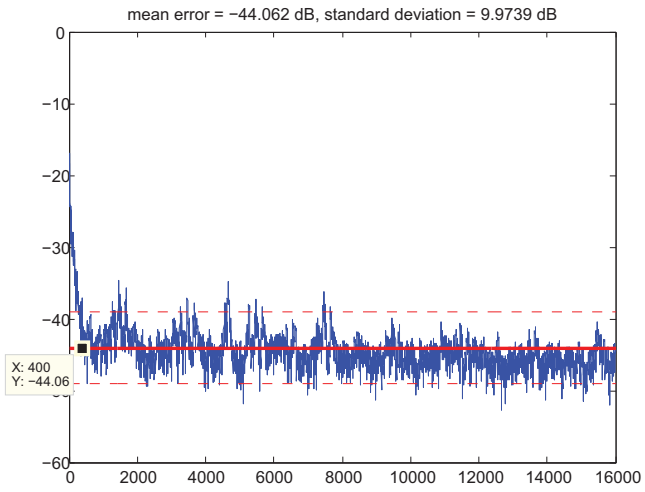


(b) human voice

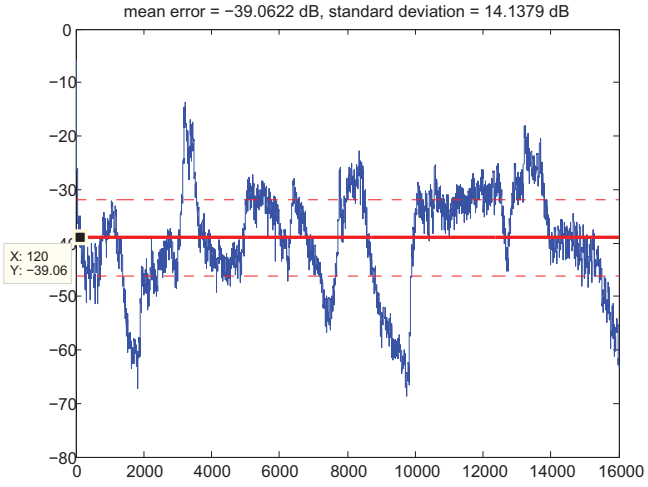


(c) instrumental music

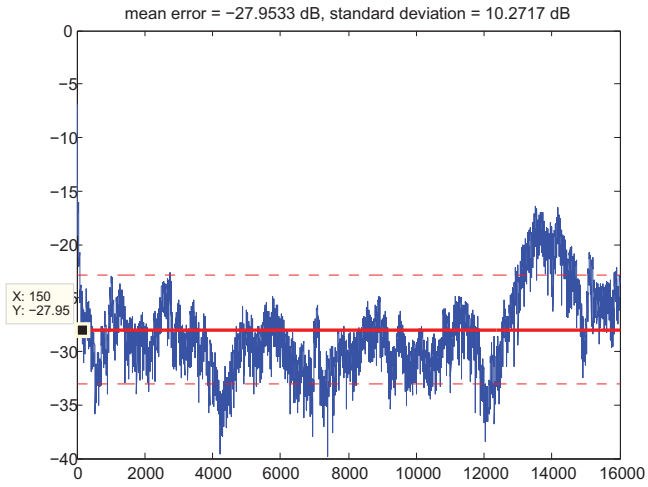
Fig. 3. Error in speaker-1 where $b = -0.1$ and $c = 0.1$



(a) periodic signal



(b) human voice



(c) instrumental music

Fig. 4. Error in speaker-2 where $b = 0.05$ and $c = 0.05$