A DISTRIBUTED ARITHMETIC BASED APPROACH FOR THE REALIZATION OF THE SIGNED-REGRESSOR LMS ADAPTIVE FILTER

Matcha Surya Prakash[1] and Rafi Ahamed Shaik[2]

[1]Department of ECE, NITC, Kozhikode, Kerala, India

[2]Department of EEE, IIT Guwahati, Guwahati, Assam, India

ABSTRACT

This paper presents a distributed arithmetic (DA) based approach for the implementation of signed-regressor LMS adaptive filter. DA, although is an efficient technique for the implementation of fixed coefficient filters, the adaptive filter implementation using DA is not a straight-forward task as the partial-products of the filter weights have to be updated in every iteration. This is achieved by storing the partial-products of the signum values of the input samples in a look-up-table (LUT). It has been shown that this LUT can be updated to accommodate the partial-products of newest set of samples in an efficient way using the circular- shifting of its address bits. Results indicate that the proposed filter can give better throughputs compared to multiply-and-accumulate (MAC) based implementation and can be effective when implementing large filters. With proper choice of system parameters, the proposed architecture for a 32-tap filter consumes around 87% less number of adder units while providing similar throughput performance compared to most recent existing DA based architecture.

KEYWORDS

Adaptive filter, Signed Regressor-LMS, look- up-table (LUT), offset-binary coding (OBC), multiply-and-accumulate (MAC).

1. INTRODUCTION

Many signal processing applications such as system identification, channel equalization, noise cancellation etc use adaptive filters as the basic processing units. These filters contain finite-impulse response (FIR) filters whose tap-weights are updated using an adaptation algorithm such as the Least Mean Square (LMS) algorithm. Each sample of the output of FIR filter is the weighted-sum of present and past input samples and hence such filters can be realized using multiply-and-accumulate (MAC) units. The weight-update algorithm also demands for the multiplication and addition operations (along with the shift operations) often and hence the same MAC units serve the purpose of weight-adaptation. The number of MAC units used for the realization depends on the speed and complexity requirements. Complexity reduction and high sampling rate adaptive filters can be a good topic of research due to the growing needs for higher order and fast processing filters.

2. LITERATURE SURVEY

Early research on adaptive filter implementations are based on pipelining of the algorithms. A look-ahead based technique is used in [1] and the resultant filter is efficient in terms of hardware while preserving the convergence performance. A high speed filter useful for many signal processing application has been presented in [2]. In this work, it was shown that the ratio

David C. Wyld et al. (Eds): SIGV, AI & FL, SESBC, MLAEDU, DSCC, NLPTT, SCOM, SCM, CEEE - 2022
pp. 181-190, 2022. CS & IT - CSCP 2022 DOI: 10.5121/csit.2022.122214

of the maximum throughput rate of the realized structure and the conventional DSP processor based adaptive filter is twice that of the square of the filter size. Few other pipelined architectures also exist in the literature [3]–[7] with their own merits and demerits.

One of the better ways of implementing the adaptive filters is by the use of distributed arithmetic (DA) [8]. DA is an efficient structure for the computation of the inner product of two vectors where one of the vectors is known prior to the implementation. The basic idea behind DA is that the pre- computed partial-products are stored in memories and through the shift operations they are added up for the computation of inner product. It is obvious that DA can be effective for the implementation of FIR digital filters as they need coefficients which are constants. While early works based on DA mainly targeted filters and transforms, few works on adaptive filter implementations also exist in the literature [9]–[11]. The idea in [9] may not be practical as there were approximations made in the standard algorithms. In [10], a multi-memory based structure is presented and it was shown that the structure is more feasible for noise cancellation applications due to its simplicity. For some architectures, the focus was upon increasing the throughput [12]–[15]. In the recent past, several architectures have been presented by researchers for the implementation of adaptive filter that uses LMS and other algorithms [16]-[35]. In this paper, based on the idea used in [15], we modify the filtering and weight-update equations of the signed-regressor LMS based filter so that the filter structure can be implemented using DA. The paper is organized as follows. In Section III, the background of distributed arithmetic along with offset-binary coding (OBC) technique is presented. A brief description of the signed-regressor LMS adaptive filter is given in Section IV. The proposed DA based structure is described in detail in Section V and the performance analysis is presented in Section VI.

3. DISTRIBUTED ARITHMETIC BACKGROUND

Consider the inner product of two vectors '$\boldsymbol{p}$' and '$\boldsymbol{q}$' ($i = 0, 1, \ldots, N-1$) given as

$$y = \sum_{i=0}^{N-1} p_i q_i \tag{1}$$

where p_i and q_i are the elements of the vectors.

Let every sample of $\boldsymbol{q}$ i.e., q_i is represented in signed two's complement representation, given by

$$q_i = -b_{i,B-1} + \sum_{j=1}^{B-1} b_{i,B-1-j} 2^{-j} \tag{2}$$

where $b_i \in \{0,1\}$, $j = 0,1, \ldots, B-1$ are the binary bits of q_i.

The offset-binary coding (OBC) technique [8], can be used by using the relation $q_i = \frac{(q_i - (-q_i))}{2}$ with $-q_i = -\bar{b}_{i,B-1} + \sum_{j=1}^{B-1} \bar{b}_{i,B-1-j} 2^{-j} + 2^{-(B-1)}$. This gives:

$$q_i = -\frac{1}{2}\left(b_{i,B-1} - \bar{b}_{i,B-1}\right) + \frac{1}{2}\sum_{j=1}^{B-1}\left(b_{i,B-1-j} - \bar{b}_{i,B-1-j}\right) 2^{-j} - 2^{-(B-1)} \tag{3}$$

By using $d_{i,B-1} = (1/2)\left(b_{i,B-1} - \bar{b}_{i,B-1}\right)$ and $d_{i,B-1-j} = (1/2)\left(b_{i,B-1-j} - \bar{b}_{i,B-1-j}\right)$,

$$q_i = -\frac{1}{2} d_{i,B-1} + \frac{1}{2}\sum_{j=1}^{B-1} d_{i,B-1-j}\, 2^{-j} - 2^{-(B-1)} \tag{4}$$

Substitution of (4) in (1) and by re-arrangement, we get

$$y = -\sum_{i=0}^{N-1} p_i d_{i,B-1} + \sum_{j=1}^{B-1}\left\{\sum_{i=0}^{N-1} p_i d_{i,B-1-j}\right\} 2^{-j} - \frac{1}{2}\left(\sum_{i=0}^{N-1} p_i\right) 2^{-(B-1)} \tag{5}$$

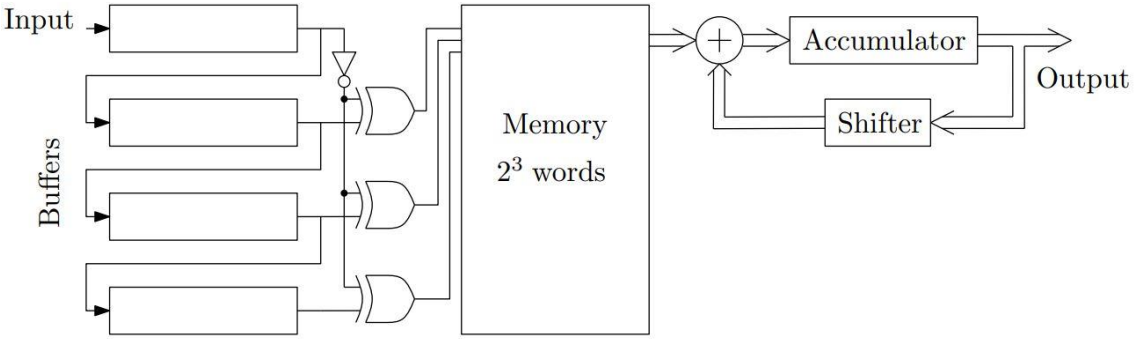

Figure 1. General Block Diagram of OBC-DA processing Unit.

Let

$$F_j = \sum_{i=0}^{N-1} p_i \, d_{i,B-1-j} \tag{6}$$

$$C_{B-1-j} = \begin{cases} -F_0, & j = 0 \\ F_j, & j \neq 0 \end{cases} \tag{7}$$

Also let

$$C_{extra} = \frac{1}{2}\sum_{i=0}^{N-1} p_i \tag{8}$$

Hence, (5) becomes

$$y = \sum_{j=0}^{B-1} C_{B-1-j} \, 2^{-j} - C_{extra} 2^{-(B-1)} \tag{9}$$

From (6) and (7), it can be observed that, taking j-th bit from each of x_i, the term P_{B-1-j} can take only one out of 2^N possible combinations which are nothing but the partial products of elements of $\boldsymbol{p}$. Hence, if the vector $\boldsymbol{p}$ is known prior to the implementation, these partial-products can be stored in a look-up-table (LUT), which is typically a memory whose address bits are formed by j-th bit of every q_i. In this way, the output y can be computed by shift-and-accumulate operations on these partial-products as given by (9).

The partial-products with and without OBC scheme of the elements of vector $\boldsymbol{p}$ are shown in Fig. 2. In case of the OBC scheme, the upper and lower half of the contents of the LUT are mirror image to each other and therefore only one half is enough for the computation of the inner product in which case the remaining half can be generated using the MSB of the address bits and exclusive-OR (Ex-OR) gates. The general block diagram of DA processing unit with OBC scheme with $B = 4$ is shown in Fig. 1. It has to be noted that the NOT gate in Fig. 1 may or may not be used based on the choice of upper half or lower half of the partial-products in OBC scheme. In DA, the size requirement of the memory will be high when N is large. In such cases, multiple smaller LUTs may be used and the output from each of the LUTs may be added up for the generation of the required partial-product. This is known as ROM decomposition [8] where $k\ (= N/m), (m, k \in Z)$ decides the size of the LUTs.

Address	DA	OBC-DA
0 0 0 0	0	$-(p_0+p_1+p_2+p_3)/2$
0 0 0 1	p_3	$-(p_0+p_1+p_2-p_3)/2$
0 0 1 0	p_2	$-(p_0+p_1-p_2+p_3)/2$
0 0 1 1	p_2+p_3	$-(p_0+p_1-p_2-p_3)/2$
0 1 0 0	p_1	$-(p_0-p_1+p_2+p_3)/2$
0 1 0 1	p_1+p_3	$-(p_0-p_1+p_2-p_3)/2$
0 1 1 0	p_1+p_2	$-(p_0-p_1-p_2+p_3)/2$
0 1 1 1	$p_1+p_2+p_3$	$-(p_0-p_1-p_2-p_3)/2$
1 0 0 0	p_0	$(p_0-p_1-p_2-p_3)/2$
1 0 0 1	p_0+p_3	$(p_0-p_1-p_2+p_3)/2$
1 0 1 0	p_0+p_2	$(p_0-p_1+p_2-p_3)/2$
1 0 1 1	$p_0+p_2+p_3$	$(p_0-p_1+p_2+p_3)/2$
1 1 0 0	p_0+p_1	$(p_0+p_1-p_2-p_3)/2$
1 1 0 1	$p_0+p_1+p_3$	$(p_0+p_1-p_2+p_3)/2$
1 1 1 0	$p_0+p_1+p_2$	$(p_0+p_1+p_2-p_3)/2$
1 1 1 1	$p_0+p_1+p_2+p_3$	$(p_0+p_1+p_2+p_3)/2$

Figure 2. The partial-products stored in the look-up-tables of DA and OBC-DA.

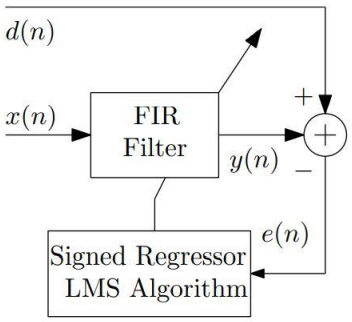

Figure 3. Block diagram of Signed Regressor-LMS based adaptive filter.

4. THE SIGNED-REGRESSOR LMS ALGORITHM

Consider an adaptive filter shown in Fig. 3 which processes an input sequence $x(n)$, $(n \in Z)$ and generates the output sequence $y(n)$ as per the following:

$$y(n) = \boldsymbol{w}^T \boldsymbol{x} \tag{10}$$

where $\boldsymbol{w}^T = [w_0(n), w_1(n), \dots, w_{N-1}(n)]$ is the filter's tap weight vector, $\boldsymbol{w}^T = [x(n), x(n-1), \dots, x(n-N+1)]$ is the vector containing input samples and N is the number of filter coefficients. When signed-regressor LMS algorithm is used, the filter weights are updated using the following recursion.

$$\boldsymbol{w}(n+1) = \boldsymbol{w}(n) + \mu e(n) sign(\boldsymbol{x}) \tag{11}$$

where $sign(.)$ represents the signum function and $e(n) = d(n) - y(n)$ is the error signal. The parameter μ is an appropriate step size which is to be chosen as $0 < \mu < \frac{2}{[\rho tr\mathbf{R}]}$ (where **R** is the auto-correlation matrix of vector containing input samples) for convergence of the algorithm.

5. DISTRIBUTED ARITHMETIC BASED REALIZATION

The filtering equation of the signed-regressor LMS adaptive filter with the filter weights w_i, $i = 0,1,2, \dots, N-1$ is given by:

$$y(n) = \boldsymbol{w}^T \boldsymbol{x} = \sum_{i=0}^{N-1} w_i \, x(n-i)$$

The above equation is similar to (1) and hence may be performed using the DA processing unit as described in Section III. The final set of equations describing the filtering operation may be given by

$$P_{B-1-j} = \begin{cases} -F_0, & j = 0 \\ F_j, & j \neq 0 \end{cases} \tag{12}$$

$$P_{extra} = \frac{1}{2} \sum_{i=0}^{N-1} w_i \tag{13}$$

Address	$LUT_{weights}$	$LUT_{input-samples}$
0 0 0	$(w_0(n) - w_1(n) - w_2(n) - w_3(n))/2$	$(-x_s(n-1) - x_s(n-2) - x_s(n-3))/2$
0 0 1	$(w_0(n) - w_1(n) - w_2(n) + w_3(n))/2$	$(-x_s(n-1) - x_s(n-2) + x_s(n-3))/2$
0 1 0	$(w_0(n) - w_1(n) + w_2(n) - w_3(n))/2$	$(-x_s(n-1) + x_s(n-2) - x_s(n-3))/2$
0 1 1	$(w_0(n) - w_1(n) + w_2(n) + w_3(n))/2$	$(-x_s(n-1) + x_s(n-2) + x_s(n-3))/2$
1 0 0	$(w_0(n) + w_1(n) - w_2(n) - w_3(n))/2$	$(x_s(n-1) - x_s(n-2) - x_s(n-3))/2$
1 0 1	$(w_0(n) + w_1(n) - w_2(n) + w_3(n))/2$	$(x_s(n-1) - x_s(n-2) + x_s(n-3))/2$
1 1 0	$(w_0(n) + w_1(n) + w_2(n) - w_3(n))/2$	$(x_s(n-1) + x_s(n-2) - x_s(n-3))/2$
1 1 1	$(w_0(n) + w_1(n) + w_2(n) + w_3(n))/2$	$(x_s(n-1) + x_s(n-2) + x_s(n-3))/2$

Figure 4. Contents of LUT_weights and LUT_input-samples in the *n*th iteration before the weight-update operation.

where

$$F_j = \sum_{i=0}^{N-1} w_i \, d_{i,B-1-j} \tag{14}$$

If the upper half of the partial-products are stored in the LUT as described in Section III, then the entry of the LUT at the address location a can be given by

$$A_j^{(a)}(n) = \frac{1}{2} w_0 + \sum_{i=1}^{N-1} w_i (-1)^{c_{N-1-i}^{(a)}+1} \tag{15}$$

where $c_i^{(a)}$ is the ith bit in the binary representation of a. We call this LUT as $LUT_{weights}$.

Although the filtering operation may be performed using DA, the weight-update operation would be difficult as the partial products of the filter weights have to be updated. For this purpose, we use one more LUT where the partial-products of the $signum$ values of the input samples are stored, for the reason that the weight-update operation needs the $signum$ value of the input samples as described by (11). The contents of this LUT can be obtained by replacing the filter weight samples with the $signum$ values of the corresponding input samples in (15). Specifically, if $x_s(n-i) = sign\big(x(n-i)\big)$, then a new equation may be obtained by replacing the the w_i terms with $x_s(n-i)$ terms $(i = 0, 1, \ldots, N-1)$ which can be given as

$$\bar{A}_j^{(a)}(n) = \frac{1}{2} x_s(n) + \sum_{i=1}^{N-1} x_s(n-i)(-1)^{c_{N-1-i}^{(a)}+1} \tag{16}$$

Let,

$$C(n) = \frac{1}{2} x_s(n) \tag{17}$$

and

$$B_j^{(a)}(n) = \sum_{i=1}^{N-1} x_s(n-i)(-1)^{c_{N-1-i}^{(a)}+1} \tag{18}$$

The left hand side of (18) represent the content at address location a of an LUT storing the partial-products of $signum$ values of input samples excluding the term of the $signum$ value of the most recent input sample. We call this LUT as $LUT_{input-samples}$. The weight-update equation for the DA based approach can now be given as

$$A_j^{(a)}(n+1) = A_j^{(a)}(n) + \mu e(n) \left[C(n) + B_j^{(a)}(n) \right] \tag{19}$$

The contents of $LUT_{weights}$ and $LUT_{input-samples}$ in the nth iteration is shown in Fig. 4. The challenge now, is to update $LUT_{input-samples}$ as well as the term $C(n)$ in every iteration. For this purpose, we use the following equation to update $LUT_{input-samples}$ based on the observation that the oldest input sample in the current iteration is not useful in the next iteration.

$$B_j^{(a)}(n+1) = \frac{1}{2}\left[B_j^{\left(2\left\lfloor\frac{a}{2}\right\rfloor\right)}(n) + B_j^{\left(2\left\lfloor\frac{a}{2}\right\rfloor+1\right)}(n)\right] + (-1)^{a+1}C(n) \tag{20}$$

In other words, taking the average of the pair of consecutive location of $LUT_{input-samples}$ would generate a term that is independent of the $signum$ value of the oldest input sample. Then by addition and subtraction of the result with the term $C(n)$ would generate terms that can once again be stored in the same consecutive locations. In this way, all the locations of $LUT_{input-samples}$ can be updated. The term $C(n)$ can be stored in a register which can be updated by storing the shifted version of the new input sample i.e., the term $\frac{1}{2}x(n+1)$ and we call it as C-register. The resultant $LUT_{input-samples}$ and C-register terms at various time instants based on the above LUT-update scheme is depicted in Fig 5. Using the above memory update scheme, although, the LUT has been updated, the new partial-products containing the newest set of samples cannot be in proper order for the weight-update operation as can be observed in Fig. 5. However, the new partial-products are placed in those locations, the addresses of which are actually the circularly right-shifted versions of the addresses of the locations in which they are supposed to be (in order to preserve the pattern of the partial-products). Hence, in every iteration this can be corrected just by circularly-left shifting of the address bits of $LUT_{input-samples}$.

The step by step procedure for the DA based approach for one complete iteration of the filter is explained as follows. The incoming bits of input samples are stored in the buffers and one bit is shifted per one clock cycle. In this way the filtering is done and during the same time, $LUT_{input-samples}$ and C-register are updated so that the sign term of the newest input sample is stored while the sign term of the oldest input sample is eliminated. The error $e(n)$ is calculated using the desired signal samples and in order to avoid any multiplication operation, $e(n)$ can be quantized to the nearest powers of 2 so that the multiplication (for gradient estimation) becomes a shifting operation. The partial-products of filter weights are then updated as follows. Each of the entry of $LUT_{input-samples}$ is accessed and multiplied (using shift operation) with the $\mu e(n)$, the result is then added to the same address location entry of $LUT_{weights}$ and the final result is stored back in the same location of $LUT_{weights}$. This completes the weight-update operation and once the term P_{extra} which can be stored in a register is updated (new P_{extra} will be the entry of the last address location of $LUT_{weights}$), the system will be ready for the next iteration.

6. PERFORMANCE ANALYSIS

The proposed DA based scheme has been verified by using a system identification application. A total of 1024 normally distributed samples have been given as input to an unknown system. The same set of input samples have been provided as an input to the adaptive filter as well which is used for the identification of the unknown system. The output of the unknown system is mixed with additive white gaussian noise (AWGN) and the noisy sequence is provided as the desired sequence to the adaptive filter. The length of impulse responses of both the unknown system and the adaptive filter have been taken as 8. In order to compare the DA based scheme with the traditional multiply-and-accumulate (MAC) based scheme, both have been implemented taking the step size as 2^{-2}. The corresponding convergence curves for both the implementations is shown in Fig. 5. It is clear that there is only a negligible effect on the convergence performance due the quantization of the error. A rough estimation of the number of clock cycles the system

would take for one complete iteration can be calculated as follows. The filtering operation is performed in B clock cycles during which $LUT_{input-samples}$ is also updated. When the ROM decomposition technique is used, the adder tree would take $\lceil log_2(m) \rceil$ clock cycles. Hence, the system would take $max\left(B, 2^{(k-1)}\right) + \lceil log_2(m) \rceil$ clock cycles just for the filtering operation and to make $LUT_{input-samples}$ for weight-update operation. The process of updating the C-register and the computation of error would take a single clock cycle. The process of updating $LUT_{weights}$ would take $2^{(k-1)}$ clock cycles. Hence, a total of $max\left(B, 2^{(k-1)}\right) + \lceil log_2(m) \rceil + 2^{(k-1)}$ clock cycles are required for one iteration of the filter. The throughput performance in case of MAC based implementation degrade with respect to the filter order while for DA based implementation, the curves are more horizontal which makes the DA based implementation more suitable for higher order filters. For $N = 32$ and with proper choice of m and k, the proposed architecture consumes around 87% less number of adder units while providing similar throughput performance as that of [35]. Comparison of throughput and hardware resource utilization for the proposed and different architectures is provided in Tab. I.

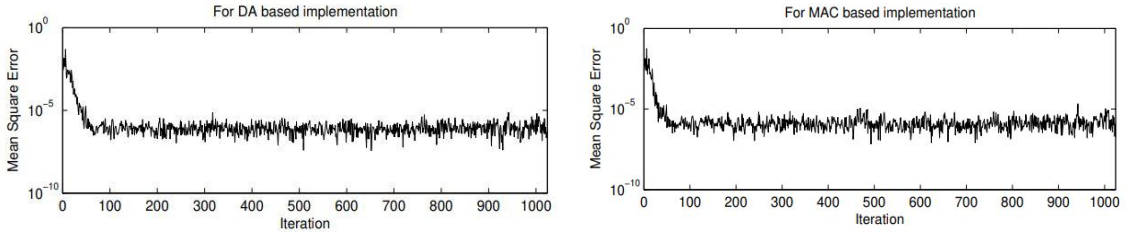

Figure 5. Mean Square Error plots for the DA based and MAC based implementations.

Scheme	Throughput	Adders	Registers
[32]	$1/[m_2(T_R + kT_{MUX} + T_A)]$	$m\left(2^{k-1} + 2^{k-1}\right) + mB - 1$	$m(1+k)+3$
[33] Scheme 1	$2^{k-1} + B + \lceil log_2 m \rceil + 1$	$m\left(2^{k+1} + 1 + k\right) + mB - 1$	$m(2+2k)$
[33] Scheme 2	$2^{k-1} + B + \lceil log_2 m \rceil + 1$	$m\left(2^{k+1} + 1 + k\right) + mB + 1$	$m(3+2k)$
[34] Scheme 1	$1/k_4[max(a_0, a_1, a_2)]$	$(2L+3)M+1$	$(2L+4)M+7$
[34] Scheme 2	$1/W[max(b_0, b_1, b_2)]$	$(L+3)M+1$	$(2L+4)M+7$
[34] Scheme 3	$1/W[max(c_0, c_1, c_2)]$	$(L+3)M+1$	$(2L+4)M+7$
[35] Scheme 1	$1/[B(2T_{add} + T_{AND} + T'_{FA} + T_{xor} + T_d)]$	$(2+L)M+N$	$(4+L)M+2N+4$
[35] Scheme 2	$1/[B(2T_{add} + T_{MUX} + T'_{FA} + T_{xor} + T_d)]$	$(2+L)M+N+1$	$(4+L)M+2N+4$
Proposed	$max\left(B, 2^{(k-1)}\right) + \lceil log_2(m) \rceil + 2^{(k-1)}$	$16+m$	$NB+3B$

T_R – LUT access time, T_{MUX} – Multiplexer delay, $N = m \times k$, $M = N/L$ – number of DA base units, B-wordlength of filter weights, $m_2 = 2^{k/2} + max\left(W, 2^{(k/2)}\right) + log_2 m + 1$, $a_0 = T_{FA'} + 2T_X + 2T_D + T_A$, $a_1 = T_{FA'} + 2T_X + 2T_D + 3T_A$, $a_2 = T_{FA'} + 2T_X + 2T_D + 2T_A$, $b_0 = T_{FA'} + T_X + T_D + T_M + T_A log_2 M$, $b_1 = T_{FA'} + T_X + T_D + T_M + T_A log_2 M$, $b_2 = T_{FA'} + T_X + T_D + T_M + T_A log_2 M$, $c_0 = 3T_A + (1 + log_2 M)T_m$, $c_1 = 3T_A + (1 + log_2 M)T_m$, $c_2 = 3T_A + (1 + log_2 M)T_m$, T_{ACC}, T_M, T_A, T_{FA}, $T_{FA'}$ and T_D are computational delays due to LUT, multiplexer, adder, binary CSA, MMP-CSFA and D-flipflop (FF) respectively.

Table I. Comparison of throughput and utilization of hardware resources of proposed and existing architectures.

7. CONCLUSION

In this paper, a signed regressor LMS based adaptive filter based on distributed arithmetic is presented. Based on the DA framework, the filtering and weight-update equations of the filter are modified. The proposed implementation consists of two LUTs, one for filtering operation and the other to aid the weight-adaptation process. It is shown that the LUT that stores the partial-products of signum values of the input samples can be smartly updated using the circular shifting of its addressing bits. The proposed architecture can operate with good throughput rates, suitable for implementation of large filters and consumes less number of adder units compared to recently evolved architecture.

ACKNOWLEDGEMENTS

This work is the outcome of the research work carried at Indian Institute of Technology Guwahati, National Institute of Technology Calicut and was supported by Ministry of Education of India.

REFERENCES

[1] K. Matsubara, K. Nishikawa, and H. Kiya, “Pipelined lms adaptive filter using a new look-ahead transformation,” IEEE Trans. Circuits Syst. II, Exp. Briefs., vol. 46, no. 1, pp. 51–55, 1999.

[2] H. K. Kwan and Q. P. Li, “High-speed realisation of adaptive linear phase fir digital filters,” IEE Proc. F Radar and Signal Process., vol. 140, no. 1, pp. 48–54, 1993.

[3] K. Parhi and D. Messerschmitt, “Concurrent cellular vlsi adaptive filter architectures,” IEEE Trans. Circuits Syst., vol. 34, no. 10, pp. 1141– 1151, 1987.

[4] N. R. Shanbhag and K. K. Parhi, “Relaxed look-ahead pipelined lms adaptive filters and their application to adpcm coder,” IEEE Trans. Circuits Syst. II, Exp. Briefs, vol. 40, no. 12, pp. 753–766, 1993.

[5] N. R. Shanbhag and K. K. Parhi, “A pipelined adaptive lattice filter architecture,” IEEE Trans. Signal Process., vol. 41, no. 5, pp. 1925– 1939, 1993.

[6] Q. Zhu, S. C. Douglas, and K. F. Smith, “A pipelined architecture for lms adaptive fir filters without adaptation delay,” in Proc. IEEE Int. Conf. on Acoust., Speech, Signal Process. ICASSP-97, vol. 3, pp. 1933–1936, 1997.

[7] S. C. Douglas, Q. Zhu, and K. F. Smith, “A pipelined lms adaptive fir filter architecture without adaptation delay,” IEEE Trans. Signal Process., vol. 46, no. 3, pp. 775–779, 1998.

[8] K. Parhi, VLSI Digital Signal Processing Systems: Design And Implementation. Wiley India Pvt. Limited, 2007.

[9] C. F. N. Cowan and J. Mavor, “New digital adaptive-filter implementation using distributed-arithmetic techniques,” Proc. Inst. Elect. Eng., vol. 128, pp. 225–230, Aug. 1981.

[10] C.-H. Wei and J.-J. Lou, “Multimemory block structure for implementing a digital adaptive filter using distributed arithmetic,” Proc. Inst. Elect. Eng., vol. 133, pp. 19 –26, Feb. 1986.

[11] G. Sicuranza and G. Ramponi, “Adaptive nonlinear digital filters using distributed arithmetic,” IEEE Trans. Acoust., Speech, Signal Process., vol. 34, pp. 518–526, Jun. 1986.

[12] D. J. Allred, H. Yoo, V. Krishnan, W. Huang, and D. V. Anderson, “LMS adaptive filters using distributed arithmetic for high throughput,” IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 52, pp. 1327 –1337, Jul. 2005.

[13] R. Guo and L. S. DeBrunner, “Two high-performance adaptive filter implementation schemes using distributed arithmetic,” IEEE Trans. Circuits Syst. II, Exp. Briefs., vol. 58, pp. 600–604, Sept. 2011.

[14] M. Surya Prakash and R. Shaik, “High performance architecture for LMS based adaptive filter using distributed arithmetic,” in 2012 Int. Conf. on Information and Computer Applications (ICICA 2012),, vol. 24, pp. 18–22, Mar. 2012.

[15] M. S. Prakash and R. A. Shaik, “Low-Area and High-Throughput Architecture for an Adaptive Filter Using Distributed Arithmetic,” in *IEEE Trans. on Circ. and Sys. II: Exp. Briefs*, vol. 60, no. 11, pp. 781-785, Nov. 2013, doi: 10.1109/TCSII.2013.2281747.

[16] Reddy, S. Raghunadha, and P. JayaKrishnan, “ASIC implementation of distributed arithmetic in adaptive FIR filter,” In 2017 Int. Conf. Circuit, Power and Computing Technologies (ICCPCT), pp. 1-4. IEEE, 2017.

[17] S. R. B, G. S. L and N. C K, "FPGA based Optimized LMS Adaptive Filter using Distributed Arithmetic," *2018 3rd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT)*, 2018, pp. 1863-1867, doi: 10.1109/RTEICT42901.2018.9012288.

[18] A. A. Chandekar and M. Pawar, "Delay and power optimized adaptive filter using distributed arithmetic," *2017 International conference of Electronics, Communication and Aerospace Technology (ICECA)*, 2017, pp. 256-261, doi: 10.1109/ICECA.2017.8203682.

[19] C. S. Vinitha and R. K. Sharma, "Area and Energy-efficient Approximate Distributive Arithmetic architecture for LMS Adaptive FIR Filter," *2020 International Conference for Emerging Technology (INCET)*, 2020, pp. 1-5, doi: 10.1109/INCET49848.2020.9154125.

[20] A. Changavi and T. Ogunfunmi, "On the performance of the BLMS adaptive filter based on distributed arithmetic," *2017 IEEE AFRICON*, 2017, pp. 377-383, doi: 10.1109/AFRCON.2017.8095512.

[21] M. T. Khan and S. R. Ahamed, "Area and Power Efficient VLSI Architecture of Distributed Arithmetic Based LMS Adaptive Filter," *2018 31st International Conference on VLSI Design and 2018 17th International Conference on Embedded Systems (VLSID)*, 2018, pp. 283-288, doi: 10.1109/VLSID.2018.77.

[22] S. Ahmad, S. G. Khawaja, N. Amjad and M. Usman, "A Novel Multiplier-Less LMS Adaptive Filter Design Based on Offset Binary Coded Distributed Arithmetic," in *IEEE Access*, vol. 9, pp. 78138-78152, 2021, doi: 10.1109/ACCESS.2021.3083282.

[23] M. Saritha *et al*., "Pipelined Distributive Arithmetic-based FIR Filter Using Carry Save and Ripple Carry Adder," *2021 2nd International Conference on Communication, Computing and Industry 4.0 (C2I4)*, 2021, pp. 1-6, doi: 10.1109/C2I454156.2021.9689396.

[24] M. T. Khan and S. Rafi Ahamed, "Enhanced Convergence Distributed Arithmetic based LMS Adaptive Filter using Convex Combination," *2018 Twenty Fourth National Conference on Communications (NCC)*, 2018, pp. 1-6, doi: 10.1109/NCC.2018.8600171.

[25] M. T. Khan and S. R. Ahamed, "A New High Performance VLSI Architecture for LMS Adaptive Filter Using Distributed Arithmetic," *2017 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, 2017, pp. 219-224, doi: 10.1109/ISVLSI.2017.46.

[26] M. T. Khan and R. A. Shaik, "Analysis and Implementation of Block Least Mean Square Adaptive Filter using Offset Binary Coding," *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2018, pp. 1-5, doi: 10.1109/ISCAS.2018.8350946.

[27] M. T. Khan and R. A. Shaik, "High-Performance Hardware Design of Block LMS Adaptive Noise Canceller for In-Ear Headphones," in *IEEE Consumer Electronics Magazine*, vol. 9, no. 3, pp. 105-113, 1 May 2020, doi: 10.1109/MCE.2020.2976418.

[28] M. T. Khan, R. A. Shaik and M. A. Alhartomi, "An Efficient Scheme for Acoustic Echo Canceller Implementation Using Offset Binary Coding," in *IEEE Transactions on Instrumentation and Measurement*, vol. 71, pp. 1-14, 2022, Art no. 2001114, doi: 10.1109/TIM.2021.3132087.

[29] H. B. Kundhu Prabakaran and A. Yada, "High Throughput Parallelized Realization Of Adaptive FIR Filter Based On Distributive Arithmetic Using Offset Binary Coding," *2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, 2019, pp. 1-6, doi: 10.1109/ICCCNT45670.2019.8944681.

[30] M. T. Khan and R. A. Shaik, "High-Performance VLSI Architecture of DLMS Adaptive Filter for Fast-Convergence and Low-MSE," in *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 69, no. 4, pp. 2106-2110, April 2022, doi: 10.1109/TCSII.2022.3141687.

[31] M. T. Khan, J. Kumar, S. R. Ahamed and J. Faridi, "Partial-LUT Designs for Low-Complexity Realization of DA-Based BLMS Adaptive Filter," in *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 68, no. 4, pp. 1188-1192, April 2021, doi: 10.1109/TCSII.2020.3035693.

[32] M. T. Khan, S. R. Ahamed and F. Brewer, "Low Complexity and Critical Path Based VLSI Architecture for LMS Adaptive Filter Using Distributed Arithmetic," *2017 30th International Conference on VLSI Design and 2017 16th International Conference on Embedded Systems (VLSID)*, 2017, pp. 127-132, doi: 10.1109/VLSID.2017.16.

[33] M. T. Khan, M. A. Alhartomi, S. Alzahrani, R. A. Shaik and R. Alsulami, "Two Distributed Arithmetic Based High Throughput Architectures of Non-Pipelined LMS Adaptive Filters," in *IEEE Access*, vol. 10, pp. 76693-76706, 2022, doi: 10.1109/ACCESS.2022.3192619.

[34] M. T. Khan and R. A. Shaik, "Optimal Complexity Architectures for Pipelined Distributed Arithmetic-Based LMS Adaptive Filter," in *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 66, no. 2, pp. 630-642, Feb. 2019, doi: 10.1109/TCSI.2018.2867291.

[35] M. T. Khan and S. R. Ahamed, "VLSI realization of low complexity pipelined LMS filter using distributed arithmetic," *TENCON 2017 - 2017 IEEE Region 10 Conference*, 2017, pp. 433-438, doi: 10.1109/TENCON.2017.8227903.

Authors

Dr. Matcha Surya Prakash is working as Assitant Professor in the department of ECE at National Institute of Technology Calicut, Kerala, India. He obtained his B. Tech degree from JNTU Kakinada, Andhra Pradesh, India and Ph.D degree from Indian Institute of Technology Guwahati, Assam, India. His research interests include Signal Processing algorithms and VLSI Architectures, VLSI for communications, VLSI for multimedia, ASIC design, design of consumer electronics products.

Dr. Shaik Rafi Ahamed received the B. Tech and M. Tech degrees in Electronics and Communication Engineering from Sri Venkateswara University, Tirupati, India in 1991 and 1993 respectively and Ph.D degree from IIT Kharagpur, India, in 2008. He is currently a Professor in the Department of EEE, IIT Guwahati, Assam, India. His research interests include digital, adaptive, biomedical and VLSI Signal Processing.