FPGA Implementation of Efficient VLSI Architecture of DLMS Adaptive Filter Algorithm

[1]Gangadharaiah.S. L, [2]Narayanappa C K, [3]Nuthan Prasad V, [4]Divya M. N,[5]Anirudha Rao

[1] Research Scholar, VTU Research Centre, Dept. of E&C, M.S. Ramaiah Institute of Technology, VTU, Belagavi, India.
[2] Department of Medical Electronics, M.S. Ramaiah Institute of Technology, Bangalore, India.
[3] Department of Electronics and Communication, M. S. Ramaiah Institute of Technology, Bengaluru, India.
[4] Research Scholar, VTU Research Centre, Dept of Medical Electronics, M.S. Ramaiah Institute of Technology, VTU,Belagavi, India.
[5] PG Student,Department of Electronics and Communication, M. S. Ramaiah Institute of Technology, Bengaluru, India.

ABSTRACT
In this Paper, an efficient implementation of the direct form of the least mean square (LMS) adaptive filter algorithm is proposed. The conventional multiplier hinders the speed of the delayed LMS (DLMS) algorithm; hence latest high speed Vedic Multiplier is used for its high convergence rate, further, the Vedic Multiplier is explored for reducing the number of logic levels and timing levels, and possible reduction in logic delay. The efficient adders are used in digital signal processing applications to reduce the power requirement, area and delay. Vedic Multiplier and different adders (such as carry-increment, carry-look ahead, carry-select, ripple-carry and carry-skip adder) used to implement the DLMS algorithm are compared (a) with ripple-carry adder and vedic multiplier, (b) with vedic multiplier and carry-look ahead adder, (c) with Vedic multiplier and carry-skip adder, (d) with Vedic multiplier and carry-increment adder, with (e) Vedic multiplier and carry-select adder based on their delay. Xilinx ISE is used for simulation and synthesis of the design with Spartan-6 FPGA, XC6SLX4-2TQG144.

Key Words: Vedic Multiplier, FPGA, DLMS Algorithm, Adaptive Filter

I. INTRODUCTION

Adaptive filters have a wide range of applications, mostly used for digital signal processing; the least mean square algorithm provides better stability and faster convergence. Efficient architectures for higher-speed and lower-complexity in implementation [1], [2], another, the (DLMS-AF) direct Least-Mean-Square adaptive filter, has low register complexity and has critical path equivalent to that of transpose form but convergence is faster. The efficient architecture of the LMS is capable to achieve lower adaptation-delay and area-delay-power [2]. DLMS adaptive filter with and an efficient pipelined architecture and low-adaptation delay can be realized to have better convergence performance, which can support high input sampling rates is discussed [3], [4]. Using Vedic Math, an 8-bit multiplier can generate the partial products [5]. Adders are necessary part of Digital Signal Processing (DSP) [6]. The (DLMS) adaptive filter can reduce the register complexity, it can support the faster convergence, very small adaptation delay and no pipelining approach is necessary. Zero adaptation delay DLMS algorithm provides better performance as compared to other algorithms, Zero adaptation delay DLMS algorithm requires minimum (EPS) energy per given sample and also in comparison to other designs minimum area. The DLMS algorithm can also be used for cancellation of interference in electroencephalogram (EEG) applications [10]. There are various proposed methods to implement the LMS adaptive algorithm and its variants, also these methods can be used widely for applications in digital signal processing, a few of them have been mentioned here. The multiplier block used for the LMS algorithm can also be replaced by the Distributed Arithmetic (DA) as the multiplier causes bottleneck, the Distributed Arithmetic makes use of the memory that store Partial Products (PPs) to emulate multiplication [14]. The approximate distributed arithmetic (DA) circuits can be used to create a fixed-point finite impulse response adaptive filter. Reduction in the number of partial products in the DA architecture is achieved with the radix-8 Booth algorithm [16]. The distributed arithmetic (DA) can be used to create VLSI architecture of LMS adaptive filtering algorithm, The DA is used for the serial bit computations. In DA 'two' separate LUT's are used for storing the partial products of the input sample and filter coefficients and then follows access and summation of the entries, LUTs are multiplexed to improve performance [30]. In the architecture of the DA complexity of structures can be reduced by implementing combinations of the offset-binary coding (OBC) of input samples on hardware, some non-OBC outputs which are produced during the initial clock cycles, will be subsequently eliminated in the error computation [31]. DA based low complexity pipelined least-mean-square filter is realized by combining 2 adaptive-filters the step-sizes of combined Adaptive filters is used to adjust the convergence performance, here the 2 ADF's are replaced by one single DA based ADF [32]. Odd multiples have symmetricity between them and hence using an adder tree, offset terms are added. For higher word length to generate the odd multiples, only a few adders are required, and a single offset adder tree is adequate. Super-latch is used to pipeline adaptation delays of the LUT-less multiplier and the pipelined LMS ADF [33]. Block least-mean-square adaptive filter (BLMS-ADF) is one of the variants of the LMS ADF, partial add-store (PAS) and partial store-add (PSA) method is used to create two partial LUT designs for BLMS-ADF, the PAS method provides shorter critical path as compared to PSA [34]. A novel approximate, low-power implementation of the coefficients update block can be used for updating the coefficients of the weight update block of the LMS adaptive filter algorithm [15]. The LMS algorithm has a third-order statistical feature known as skewness [22]. High frequency noise and low frequency noise can be removed from ECG signal using ANC-ADF (active noise cancellation adaptive filter) [23],[24],[26],[27]. The (LMS) adaptive algorithm has the capability to reduce the effect of acoustic feedback of the public address system [12],[17],[19],[24],[26],[27].Least mean square algorithm is popular due to its robust nature [7], there are many variants of least mean square algorithm, whose framework is based on logarithmic number system such as least mean logarithmic square (LMLS) and least logarithmic absolute difference (LLAD), and are used to stabilize or improve convergence performance of adaptive algorithms through relative cost functions [7],[13]. Another variant of the LMS algorithm is the Pt-NLMS which is used for the sparse system identification, has application in the echo cancellation [8], [19],[24],[26],[27],[28]. Even though Pt-NLMS is very efficient, to simplify the original Pt-NLMS algorithm, it requires several reformulations, making them amenable to real-time VLSI implementations [8],[9]. There are various variants of the proportionate type adaptive filtering algorithms and their VLSI architectures that can be used for sparse system identification under impulsive noise, these variants can also be used for noise cancellation, [9], [19], [24], [26], [27]. Though when there is Non Gaussian noise, Least Mean Fourth (LMF) adaptive filter is superior to LMS algorithm, it has stability issues of low signal to noise ratio, which can be overcome by combining LMF algorithm with LMS algorithm and can be used for a sparse system identification, [7],[11],[19]. Apart from LMS algorithm there is one more very widely used algorithm for the identification of the unknown systems known as the recursive least square algorithm RLS [20],[24],[26],[27]. To improve the performance of the algorithm and solve the contradiction between the convergence speed and the steady-state error, a novel variable step length can be used for the LMS adaptive filtering algorithm [21]. There are many other variants of the LMS algorithm such as the multisplit time varying LMS algorithm (MS TVLMS) and multisplit LMS (MS LMS) algorithm [28] and affine combination of two LMS adaptive filters also there is the algorithm of reduced complexity parallel LMS (RC-pLMS) [29].

II. PROPOSED METHOD

The Least Mean Square algorithm does not support the pipelined architecture neither to improve convergence nor reduce the delay, hence the Delayed Least Mean Square adaptive algorithm is used. The adaptive filter coefficients need continuous updating to reduce difference of filter output to desired response. To implement DLMS algorithm, the error is estimated, it is next used to update filter weights for each sampling period. The error is estimated using difference of desired response and the filter output. Weight update of the DLMS adaptive filter takes place according to the equations given below:

Weight Update, $(W_{n+1}) = W_n + X_{in} * \mu * E$ (1)

The Weight update Equation is used to reduce the difference of desired response to filter output.
where, $W_{n+1}$is new weight and W is previous Weight
X_{in} is the input andμ the step size($0< \mu < 1$) and E is the Error Signal.

Filter Output, $Y = X_{in} * W_n+R$ (2)

where, Y is Filter Output, X_{in} is the input,
W_n is previous Weight andR is the Register value used for correction of error,
R is used for the
Note: Initially R value is 0

Error Calculation, $E=D-Y$ (3)
Where, E is Error Signal, D is Desired Response and Yis the Filter Output.
The Least mean square algorithm conventional block diagram is as shown below in Figure 1.

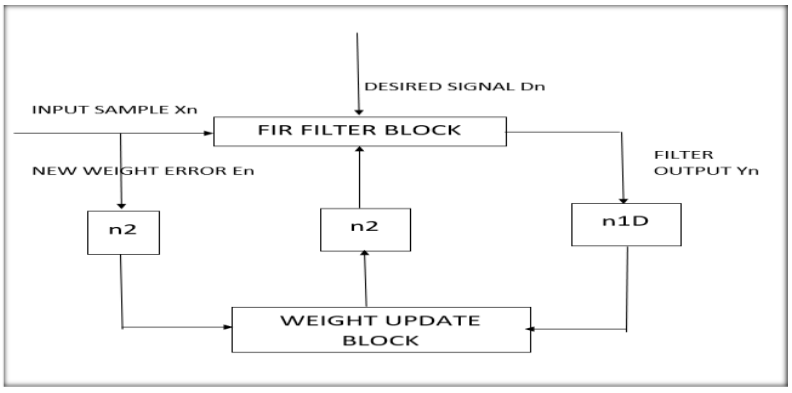


Figure 1. General block diagram of Least mean square algorithm.

In case of pipelined architectures if there are m stages of pipelined architecture, then after the completion of the m cycles of the pipeline, the error E_n becomes available, where 'm' represents the delay required for the adaptation. The Delayed error is used by the DLMS algorithm in such cases will be E_{n-m}, i.e., for the weight update operation the error being used is not the recent error but (n-m)th iteration error.
The formula for which follows is as follows:
Weight update for Pipelined architecture, $W_{n+1} = W_n + X_{n-m} * \mu * E_{n-m}$ (4)
In the conventional Least mean square (LMS) filter, the adaptation delay comprises of two parts:
1st, is pipelined stages of the Finite impulse Response (FIR), which introduces the delay and 2nd is Weight update process delay caused by pipelining.
The Direct form of the Least mean Square adaptive filter algorithm (DLMS) can be implemented based on such delay decomposition; the Direct form of the LMS adaptive filter algorithm (DLMS) is implemented as shown in Figure. 2. The Error Computation Latency is assumed to be n1 number of cycles, after the completion of the nth cycle the error is computed, it is represented by E_{n-n1}. The input samples are also delayed by n1 number of clock cycles and they are used along with the n1 clock cycles delayed error and the new weight generation takes place, the n1 clock cycles delayed input samples is represented by $X_{n-n1.}$ The weight-update, Error Calculation, Filter Output equation for the DLMS algorithm, is as shown below in equations (5),equations (6) and equations (7):

Weight update for DLMS, $W_{n+1} = W_n + \mu * E_{n-n1} * X_{n-n1}$ (5)

Error calculation for DLMS, $E_{n-n1} = D_{n-n1} - Y_{n-n1}$ (6)

Filter Output, $Y_n = W_{n-n2} * X_n + R$ (7)

The implementation of Direct Form of the FIR filter realization is used, since all weights gets updated simultaneously to generate the output.
The direct form of LMS adaptive filter performs computation of inner product, done to generate the output of the filter; hence it is assumed that it has a longer critical path as the arithmetic operation can begin only if all the input operand values are available.

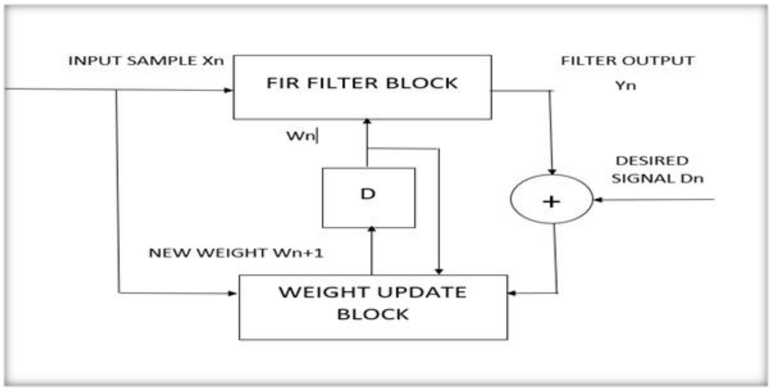


Figure 2. Block diagram of DLM

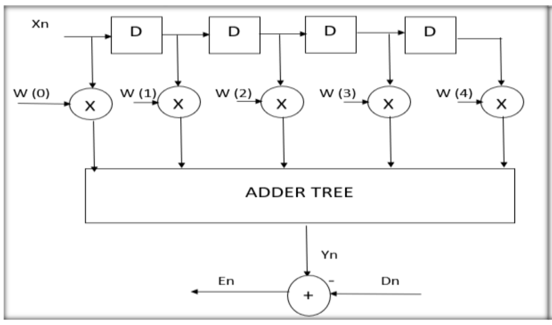
III. RESEARCH METHOD

In this section, we discuss efficient implementation of zero-adaptation delay filter using high speed Vedic multiplier and VLSI architectureof efficient adders.

2.1 Zero Adaptation Delay (ZAD) filter

The ZAD Filter has 2 main blocks which are used for the computations, 1st block shown in Figure. 3 is for computation of error and 2nd block shown in Figure. 4 is for updating the weights. When we consider these blocks separately, we will discover many such components which require large amount of area, these are the common components in both blocks, and these components include registers, adders, multipliers, delay flip-flops and subtractor.

The subtractor and adders which are being used in these two blocks are different, the computations of these two blocks needs to be performed in the same cycle to implement the zero-adaptation delay. The error computation and weight update cannot occur concurrently since the zero-adaptation delay is not having the pipelined architecture. To decrease the delay, multiplexers are used to multiplex the multiplications of both these blocks. We use the high speed vedic multipliers here to decrease the delay. For both these phases same registers are used. In the 1st half cycle error computation is performed and in the 2nd weight update is performed.



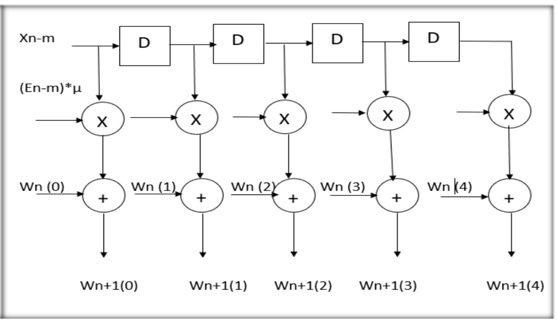


Figure 3. Error update Block of the DLMS filter **Figure 4. Weight update Block of the DLMS filter**

In the zero-adaptation delay, using the tapped delay line, the input (X_n) samples are driven into the multipliers. While in "N" number of registers the Weight values stored. Multiplication operation takes place between the weight values and the input samples and they are given to the adder tree block where they get added, output of adder tree in turn then gets subtracted from the desired response, the error generated is pushed into the "N" 2:1 mux as input. The output of the multiplier needs to be moved either towards the weight update block or towards the adder tree block. To move the product value to either of the two blocks we require "N" 2:1 demux. The "N" number of multiplexers and demultiplexers are controlled by a clock signal. Registers are clocked at rising edge and will stay clocked for entire period. Multiplexers are used to feed coefficients present in the registers into the multipliers, this is done during each clock-period in the 1st half. The adder tree receives the product from the demultiplexers, using the adder tree the output of the filter is generated. The output of adder tree i.e., filter output is then subtracted from the desired response which generates the error value. The Step size gets multiplied to the error value, this product is then multiplied to the input samples and the product value gets generated, this product is transmitted to each adder in weight update block where they get added with the previous weight values. A delay is inserted after the calculation of the product of step size and error value as it reduces register width. The Weights get updated after which the output of the filter is regenerated, it is used to recalculate the error. The time required for computation of error is more as compared to the weight updating.

2.2Multiplier and adders

The important components in the above-mentioned structure is the adders and multiplier block. To optimize the above-mentioned architecture, we focus of adder and multiplier block. The implementations of the adder and multiplier requires a large chip area, the delay is increased and even the power consumption gets increased.

So, to reduce the delay and power consumption, advanced multipliers and adders are required which can satisfy the above-mentioned requirements. There are many High-speed multipliers and adders which can replace the conventional adder and multiplier blocks. Some of the examples of multipliers are Vedic multiplier, array multiplier, Wallace tree multiplier etc. similarly some examples for the adders are ripple-carry, carry-look-ahead, carry-skip, carry-increment and carry-select adders. From the above-mentioned multipliers, *Vedic* multiplier is the most effective and efficient multiplier in terms of delay and consumption of area.The ancient Indian scriptures also known as the Vedas was used to create the Vedic multiplier by Swami Bharati Krishna Tirthaji Maharaja. The Vedic multiplier is based on sutras which basically are 16-word formulae. Vedic math is integrated with the digital signal processing to increase the speed of multiplication process.The architecture of the Vedic multiplier is based on the algorithm, the Urdhva Tiryagbhyam sutra.The 4-bit multiplications operation is done by the Urdhva Tiryagbhyam sutra in one single logic level or one single line, whereas the conventional method needs to add four different partial products to get the result.The number of steps needed to calculate the output of a multiplier is reduced in Vedic multiplier as compared to conventional multiplier, this also improves and increases the speed of the product generation.8-bit Vedic multiplier is implemented using the equations 8-15 given below.

$p\_0 = b\_0 * a\_0$ (8)
$p\_1 = b\_1 * a\_0 + b\_0 * a\_1$ (9)
$p\_2 = b\_1 *a\_1 + b\_0 * a\_2 + b\_2 *a\_0 + b\_0 * a\_0 * b\_1 * a\_1$ (10)
$p\_3 = P2 \text{ carry bit} + b\_1 * a\_2 + b\_0 * a\_3 + b\_2 \& a\_1 + b\_3 * a\_0$ (11)
$p\_4 = P3 \text{ carry bit} + P2 \text{ carry bit} + b\_1 * a\_3 + b\_2 * a\_2 + b\_3 * a\_1$ (12)
$p\_5 = P4 \text{ carry bit} + P3 \text{ carry bit} + b\_2 * a\_3 + b\_3 * a\_2$ (13)
$p\_6 = P4 \text{ carry bit} +b\_3 * a\_3 + b\_1 * b\_2 * a\_1 * a\_2$ (14)
$p\_7 = P6 \text{ carry bit}$ (15)

The 8-bit Vedic multiplier shown in Figure. 5is constructed from four, 4-bit Vedic multipliers and three, 8-bit adder blocks, adder-1, adder-2, adder-3. Figure 5 below shows 8-bit Vedic multiplier block diagram.

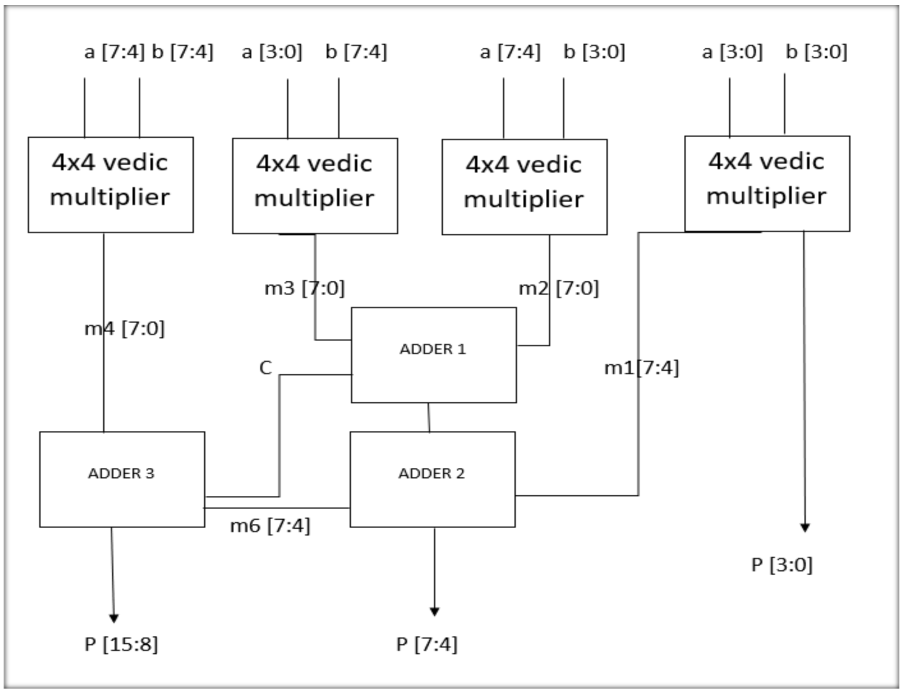


Figure 5. Block diagram of 8x8 Vedic Multiplier

Addition is the most important operation used in digital signal processing. There are several efficient adder blocks which can replace the conventional adder to perform the addition operation in digital signal processing, some examples of these adders are ripple-carry-ripple, carry-skip, carry-look ahead, carry-select, carry-increment adder etc., every adder has its own advantages and some drawbacks. The first adder which can replace the adder blocks mentioned in the architecture of the DLMS adaptive filter is the ripple carry adder shown in Figure 6.

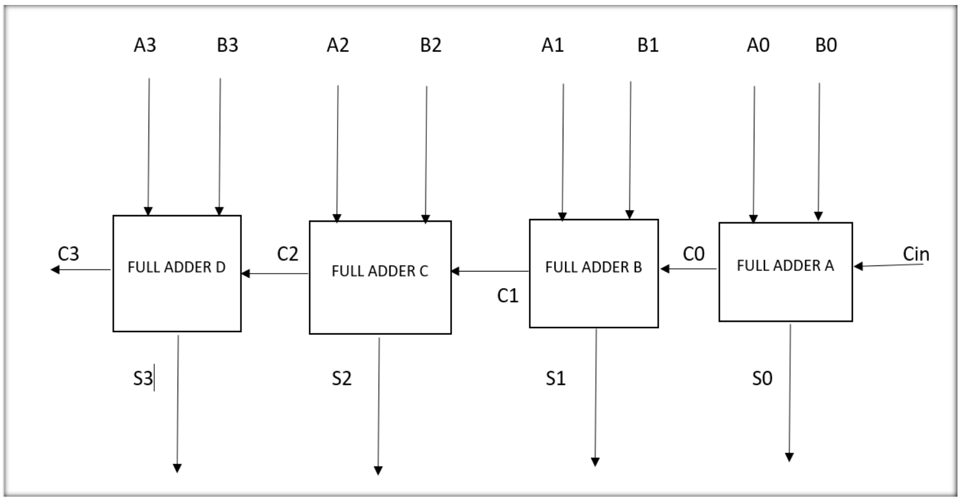


Figure 6. Block diagram of Ripple carry adder

The carry-look-ahead adder is used to generate the carry and propagate the carry bit given in equation 16 and equation 17 , there are 2 functions in the carry-look-ahead adder, one is to propagate the bit and other to generate the bit. The generate bit's purpose is to generate the carry no matter what the value of Ci and propagate bit's function is to propagate the carry bit.Carry bit Equation is shown in equation 18.

Generate bit Equation: $G\_i=X\_i*Y\_i$ (16)

Propagate bit Equation: $P\_i = Y\_i + X\_i$ (17)

Carry bit Equation: $C\_{i+1} = (C\_i*P\_i) + G\_i$ (18)

Due to independent calculation of carry bit carry look ahead adder is faster as compared to the ripple carry adder, but the hardware required increase. The Figure 7 shown below is the architecture of carry-look-ahead adder.

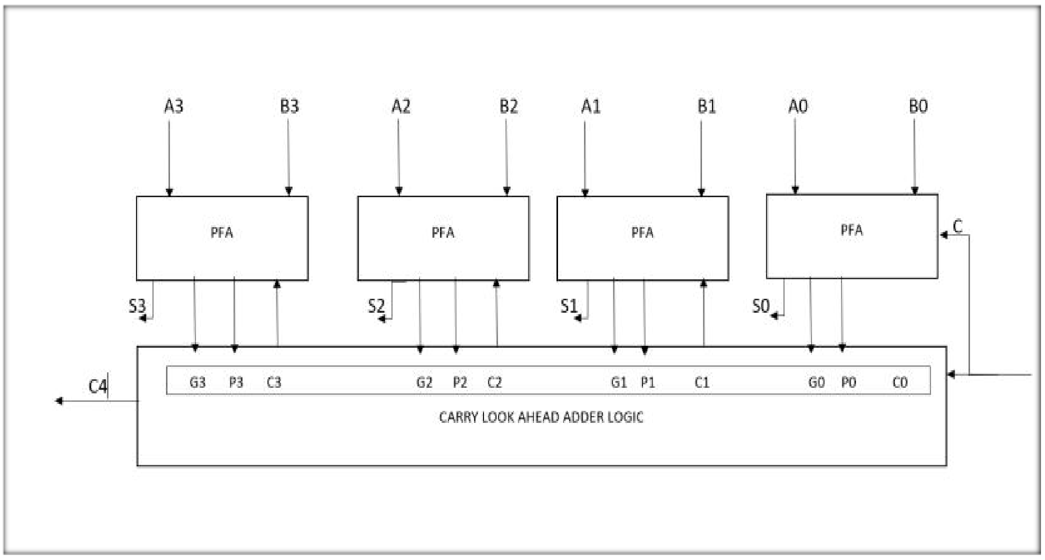


Figure 7. Block diagram of carry look ahead adder.

Carry-skip-adder shown in Figure. 8 uses skip logic to propagate carry bit, this helps in increasing the speed of addition operation. The propagated carry bit is added to the entire adder.

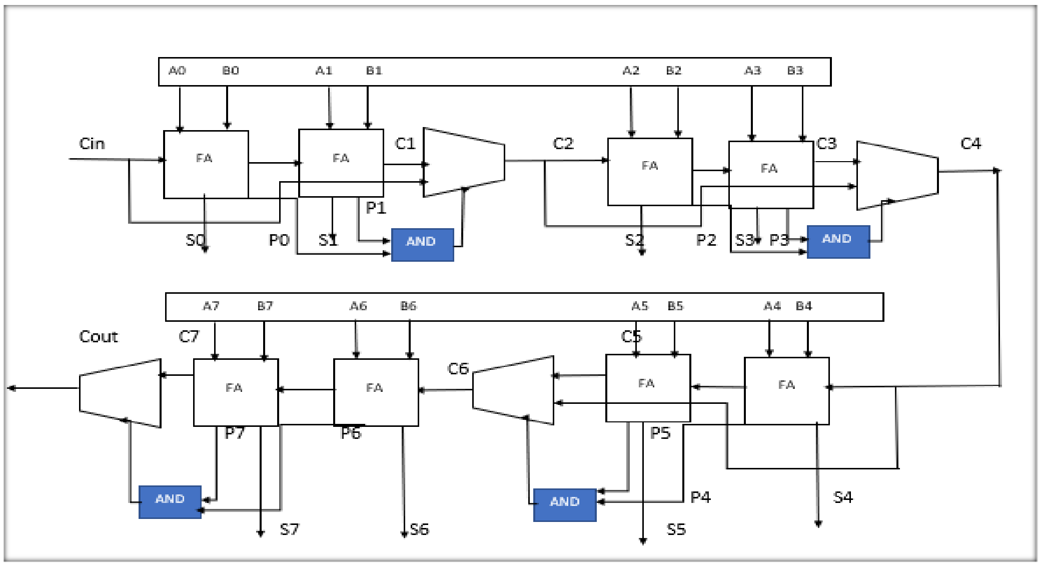


Figure 8. Block diagram of carry skip adder.

The carry-select adder shown in Figure. 9 is made up of ripple-carry adders and multiplexer, the 2 ripple carry adders get cin as input one of them gets cin as 0 and one receives cin as 1, the mux is used to decide the correct carry out and the correct sum and then provide them as output as soon as the correct carry is known. The multiplexers increase the complexity, but the delay gets reduced.

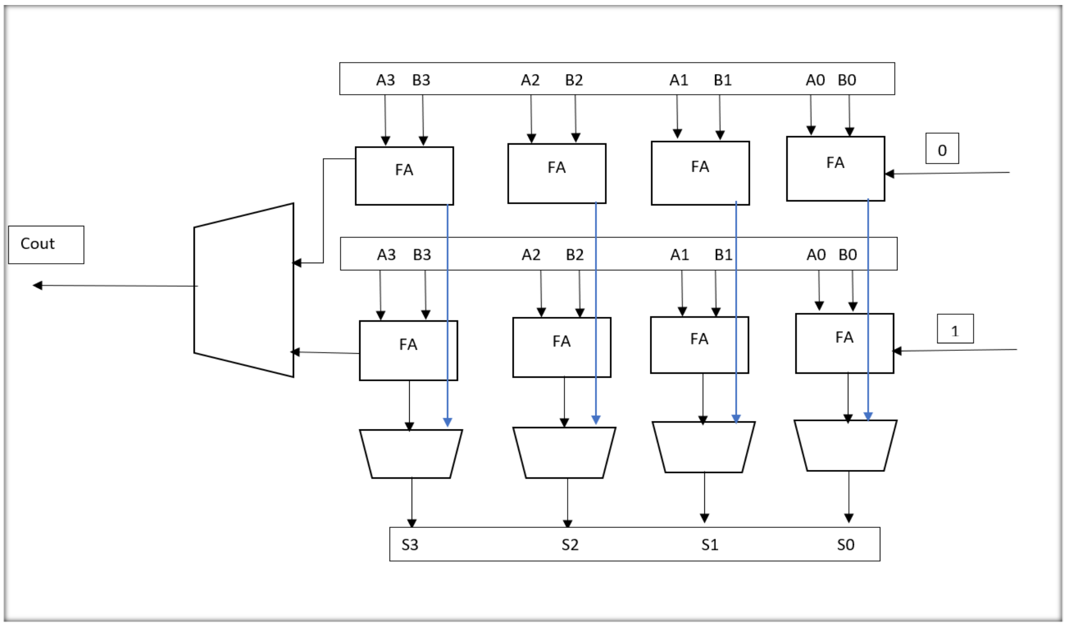


Figure 9. Block diagram of carry select adder

The carry-increment adder shown in Figure10 is created using ripple-carry adder along with the increment logic block. The increment adder logic is created using the half adders in a sequential chain as in the ripple carry adder, the total number of bits are divided in a group of 2 of 4 bits each and the addition operation is done using the 4-bit Ripple carry adders.

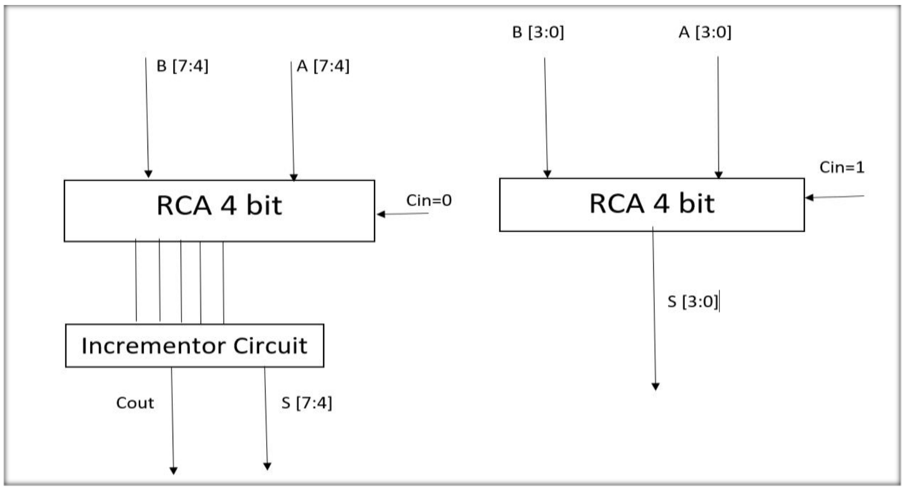


Figure 10. Block diagram of carry Increment adder

III. RESULTS AND DISCUSSION

For Proof of concept, POC we have run the DLMS algorithm equations in python version 3. "65025" Iterations have been run to check the functionality of the filter and also to check its convergence rate and the bit error rate (BER). We are able to achieve an ideal result wherein output of a filter exactly matches desired response. The below Figure 11 is the graph which shows the corrected filter output and desired response. The functionality is validated using the Test bench written in Verilog HDL.

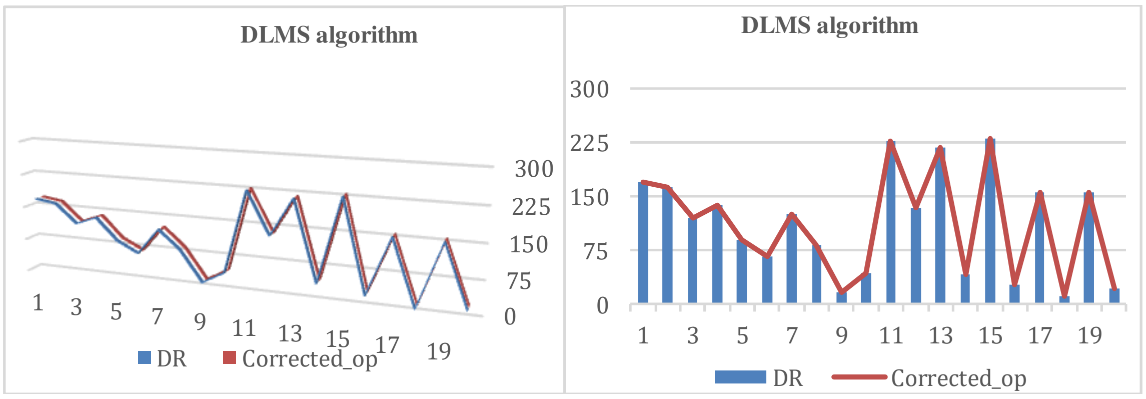


Figure 11.Graph representing the desired response DR and Corrected output

The graphs compare the Desired Response (DR) and Corrected filter output (Corrected_op). Both graphs are represent demonstrate the filter's accuracy is increased, the BER is reduced and it gives the filter output i.e., Corrected output (Corrected_op) which is equivalent to the Desired response (DR), here we have considered the step size or μ as $1/X^2$, here the X in the denominator is nothing but our input signal, the error considered for the estimation of weight update is recent most calculated error, when we update the weight using the equation (5), it gets reduced to equation (19) and due to this we get the corrected output within one iteration.

New update weight equation, $(W_{n+1}) = W_n + E_{n-n1}/X_{n-n1}$ (19)

The DLMS adaptive filter module is simulated and synthesized using Xilinx Project Navigator.

3.1 Simulation Result of 8 Bit Vedic Multiplier

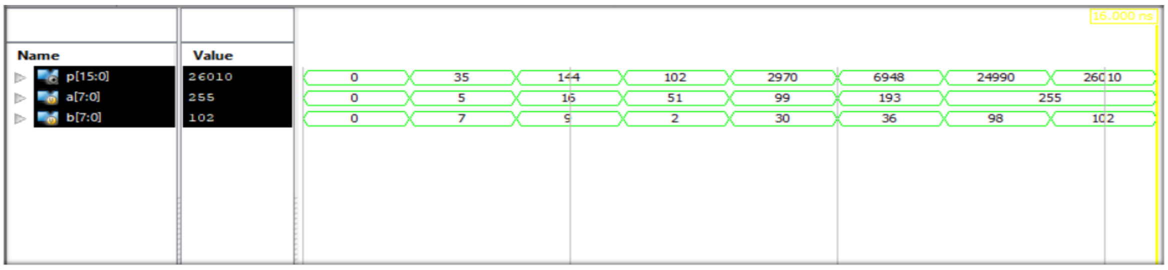


Figure 12.Simulation results of 8-bit Vedic multiplier

The Input signal is multiplied with the weight to obtain the filter output and to perform that multiplication operation in the DLMS algorithm 8-bit high speed Vedic multiplier has been used, in the aboveFigure 12, 'a and b' are the inputs to the multiplier and 'p' is the product output.

3.2 Simulation Result of 16 Bit Adders

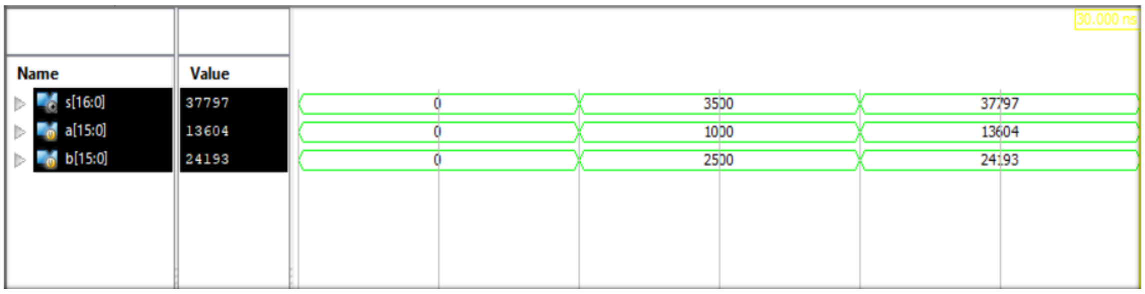


Figure 13.Simulation results of 16-bit Adders – (RCA, CLA, CSA, CSpA & CIA)

For the weight update we need to add the previous weight to the product of input signal, error and the step size and for the addition purpose we used carry-look-ahead, ripple-carry, carry-skip, carry-select, and, carry-increment efficient adders. Figure 13 is the simulation result for all the adders of 16 bit, in the above image figure 13, 'a' and 'b' are the inputs and's' are the sum output.

3.3 Simulation Result of DLMS algorithm using the 8-bit Vedic multiplier and 16-bit adders

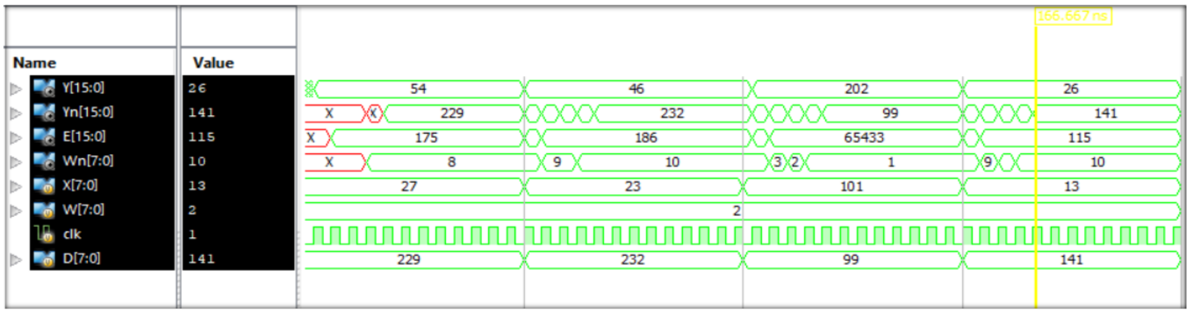


Figure 14.Simulation results of DLMS algorithm – (using Vedic - multiplier, RCA, CLA, CSA, CSpA & CIA – adders)

The above Figure 14, 'X' represents the input signal with noise, 'D' represents the desired response, 'W' represents the initial weights, 'Y' represents the filter output which we obtained after taking product of input 'X' and weight 'W'. 'E', represents the Error signal which we obtain after subtracting the filter output 'Y' from desired response 'D'. The obtained error signal is then divided by input and added to previous weight to generate the new weight 'Wn'. When we perform the division operation if it's not divisible exactly, we will get some decimal values which cannot be represented in Verilog and hence we store the modulus of error signal 'E' and the input signal 'X' in a register R. The new filter output 'Yn' is generated by multiplying the new weight 'Wn' to the input signal 'X' and also adding the register value for error correction purpose. Hence, we get the new filter output 'Yn' equivalent to the desired response (this concept has been tried on 255*255 iterations with different random input values and we got the same result for all the iterations).The Xilinx ISE is used for design and Spartan-6 FPGA for synthesis, with XC6SLX4-2TQG144 and the timing summary for 8-bit Vedic multiplier, 16-bit adders and the DLMS algorithms are mentioned in the tables below.Table 1 summarizes the timing summary of different multipliers used in the DLMS Adaptive Filter algorithm.

3.4 Timing summary of 8-bit Vedic multiplier

Table 1: Timing summary of different multipliers

Multiplier type	Modified 8-bit Vedic multiplier	Normal 8-bit Vedic multiplier	8-bit array multiplier	8-bit Wallace tree multiplier
Total delay	15ns	17.4ns	17.5ns	15.9ns
Logic level	11	13	13	12

3.5 Timing summary of 16-bit adders

Table 2 summarizes the timing summary of different adders used in the DLMS Adaptive Filter algorithm.

Table 2: Timing summary of 16-bit adders.

Name of adder	CIA	CSA	CLA	RCA	CSPA
Time delay in (ns)	4.72	5.88	6.2	7.308	6.72

3.6 Timing summary of DLMS algorithm with 8-bit vedic multiplier and 16-bit adders

Table 3 summarizes the timing summary of DLMS Adaptive Filter algorithm using different adders and multipliers.

Table 3: Timing summary of DLMS algorithm

DLMS adaptive filter using	Vedic & CIA	Vedic & CSA	Vedic & CLA	Vedic & RCA	Vedic & CSPA	Conventional adder and multiplier
Time delay in (ns)	23.711	24.091	23.484	23.817	23.670	27.16

IV. CONCLUSION

Table1 represents the timing summary of the various multipliers, the modified Vedic multiplier provides less delay compared to the other multipliers, Table 2 represents the timing summary of various adders, carry increment adder provides less delay as compared to other adders. The implementation of the DLMS algorithm using the Vedic multiplier and carry-look-ahead adder provides least delay compared to other combinations of adders and multipliers the comparison has been made with the conventional adder and multiplier and is more efficient implementation of the DLMS algorithm which also provides the corrected filter output for 65,025 different combinations of input iterations, it is more efficient as it converges faster as compared to the conventional multiplier and adder architecture.

ACKNOWLEGDEMENTS

The authors with utmost gratitude express special thanks to the Professors, Principal and mentors who were always available and made time to provide insight and guidance in the research. The research and the successful completion of the paperwork has been possible for all the facilities extended to us by our institute.

REFERENCES

1. Meher, Pramod Kumar, and Sang Yoon Park. "Critical-path analysis and low-complexity implementation of the LMS adaptive algorithm." *IEEE Transactions on Circuits and Systems I: Regular Papers* 61.3 (2013): 778-788.
2. Meher, Pramod Kumar, and Sang Yoon Park. "Area-delay-power efficient fixed-point LMS adaptive filter with low adaptation-delay." *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 22.2 (2013): 362-371.
3. Meher, Pramod Kumar, and Sang Yoon Park. "Low adaptation-delay LMS adaptive filter part-II: An optimized architecture." *2011 IEEE 54th International Midwest Symposium on Circuits and Systems (MWSCAS)*. IEEE, 2011.
4. Meher, Pramod K., and Megha Maheshwari. "A high-speed FIR adaptive filter architecture using a modified delayed LMS algorithm." *2011 IEEE International Symposium of Circuits and Systems (ISCAS)*. IEEE, 2011.
5. Premananda, B. S., et al. "Design and implementation of 8-bit Vedic multiplier." *International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering* 2.12 (2013): 5877-5882.
6. Kaur, Jasbir, and Lalit Sood. "Comparison between various types of adder topologies." *IJCST* 6.1 (2015): 62-66.
7. Mula, Subrahmanyam, Vinay Chakravarthi Gogineni, and Anindya Sundar Dhar. "Algorithm and architecture design of adaptive filters with error nonlinearities." *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 25.9 (2017): 2588-2601.
8. Mula, Subrahmanyam, Vinay Chakravarthi Gogineni, and Anindya Sundar Dhar. "Algorithm and VLSI architecture design of proportionate-type LMS adaptive filters for sparse system identification." *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 26.9 (2018): 1750-1762.
9. Mula, Subrahmanyam, Vinay Chakravarthi Gogineni, and Anindya Sundar Dhar. "Robust proportionate adaptive filter architectures under impulsive noise." *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 27.5 (2019): 1223-1227.
10. Prakash, D. M. *Realization of Delayed Least Mean Square Adaptive algorithm using Verilog HDL for EEG signals*. Diss. 2016.
11. Bashir, Murwan, and Azzedine Zerguine. "Transform domain LMF algorithm for sparse system identification under low SNR." *2015 49th Asilomar Conference on Signals, Systems and Computers*. IEEE, 2015.
12. Reas, Roxcella T. "Performance Evaluation Of A Low Order LMS Based Adaptive Feedback Canceller For Public Address System." *2020 IEEE 12th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment, and Management (HNICEM)*. IEEE, 2020.
13. Sayin, Muhammed O., N. Denizcan Vanli, and Suleyman Serdar Kozat. "A novel family of adaptive filtering algorithms based on the logarithmic cost." *IEEE Transactions on signal processing* 62.17 (2014): 4411-4424.
14. Ahmad, Shawez, et al. "A Novel Multiplier-less LMS Adaptive Filter Design based on Offset Binary Coded Distributed Arithmetic." *IEEE Access* 2021.
15. Di Meo, Gennaro, et al. "Low-power Implementation of LMS Adaptive Filters Using Scalable Rounding." *2020 27th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*. IEEE, 2020.
16. Jiang, Honglan, et al. "A high-performance and energy-efficient FIR adaptive filter using approximate distributed arithmetic circuits." *IEEE Transactions on Circuits and Systems I: Regular Papers* 66.1 (2018): 313-326.
17. Ghmati, Tarek S., and Abeer AS Elhoula. "Adaptive Digital FIR Filters: Case Study: Noise Cancellation using LMS Algorithm." *Albahit journal of applied sciences* 2.1 (2021): 30-36.
18. J. Zeng, Y. Lin, and L. Shi, “A robust quantized kernel least mean square algorithm based on the arctangent cost function in impulsive interference,” *J. Comput. Inf. Syst., vol. 11*, no. 15, pp. 5553–5560, Aug. 2015
19. Paleologu, Constantin, Jacob Benesty, and Silviu Ciochina. "Sparse adaptive filters for echo cancellation." *Synthesis Lectures on Speech and Audio Processing* 6.1 (2010): 1-124.
20. Tajdari, Teimour. "ADAPTIVE METHOD TO PREDICT AND TRACK UNKNOWN SYSTEM BEHAVIORS USING RLS AND LMS ALGORITHMS." *Facta Universitatis, Series: Electronics and Energetics* 34.1 (2021): 133-140.
21. Liu, Duo, Hao Cha, and Binbin Wang. "A Novel Variable step length LMS Algorithm based on Arctangent Compound Function." *2021 IEEE 5th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*. Vol. 5. IEEE, 2021.
22. Silva, Thiago TP, et al. "On the Skewness of the LMS Adaptive Weights." *IEEE Transactions on Circuits and Systems II: Express Briefs* 2021.
23. Prajapati, Priyank H., and Anand D. Darji. "Two Stage Step-size Scaler Adaptive Filter Design for ECG Denoising." *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2021.
24. VATANSEVER, Fahri. "NOISE CANCELLATION WITH LMS VARIANTS." *Uludağ University Journal of The Faculty of Engineering* 26.1 2021: 153-170.
25. Kher, Rahul. "IJERT-Performance Analysis of Multisplit Time Varying LMS Algorithm." 2021
26. Gogineni, Vinay Chakravarthi, and Subrahmanyam Mula. "Improved proportionate-type sparse adaptive filtering under maximum correntropy criterion in impulsive noise environments." *Digital Signal Processing* 79 (2018): 190-198.

27. Ma, Wentao, et al. "Robust proportionate adaptive filter based on maximum correntropy criterion for sparse system identification in impulsive noise environments." *Signal, Image and Video Processing* 12.1 (2018): 117-124.
28. Wagner, Kevin, and Miloš Doroslovački. Proportionate type normalized least mean square algorithms. ISTE, 2013.
29. Akkad, Ghattas, et al. "Stability Analysis of the RC-PLMS Adaptive Beamformer Using a Simple Transfer Function Approximation." *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021.
30. 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
31 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. 630-6 642, Feb. 2019, doi: 10.1109/TCSI.2018.2867291.
32. Khan, Mohd Tasleem, Rafi Ahamed Shaik, and Surya Prakash Matcha, "Improved convergent distributedarithmetic based low complexity pipelined least-mean-square filter." *IET Circuits, Devices & Systems* 12.6 (2018): 792-801.
33. R. K. Sarma, M. T. Khan, R. A. Shaik and J. Hazarika, "A Novel Time-Shared and LUT-Less Pipelined Architecture for LMS Adaptive Filter," in *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 28, no. 1, pp. 188-197, Jan. 2020, doi: 10.1109/TVLSI.2019.2935399
34. Khan, Mohd Tasleem,"Partial-LUT Designs for Low-Complexity Realization of DA-based BLMS Adaptive Filter." *IEEE Transactions on Circuits and Systems II: Express Briefs* (2020).

BIOGRAPHIES OF AUTHORS

Gangadharaiah S. L completed B.E. in Electronics and CommunicationEngineering from Kuvempu University in 1997, M. Tech. in Digital Electronics and Advanced Communication from KREC, Surathkal in 2002. Currently, he is pursuing Ph.D. in the area of VLSI signal processing under VTU, Belgaum. Presently he is working as Assistant Professor in the Department of Electronics & Communication Engineering, M. S Ramaiah Institute of Technology, Bengaluru, since 2002. His areas of interest are Analog VLSI,Digital VLSI,VLSI Signal Processing and Machine Learning.

Narayanappa C. K received Ph.D. from Visvesvaraya Technological University, Belagavi in2014 and M.Tech in Biomedical Instrumentation from Mysore University in 1996. He iscurrently working asAssociate Professor at department of Medical Electronics, M. S. RamaiahInstituteofTechnology,Bengalurusince2000.HisresearchinterestsincludeSignal&Imageprocessing and Control Systems. He is the member of ISTE, IETE and BMESI. He is also afellowatTheInstitutionofEngineers (India).

V.Nuthan Prasad completedhisB.E. in Electronics and CommunicationEngineeringfromVisvesvarayaTechnologicalUniversityin2003,M.Tech.in DigitalElectronicsandCommunication from Visvesvaraya Technological University in 2006. Currently he is pursuinghis Ph.D. in Antenna design. Presently he is working as Assistant Professor in the Department of Electronics & Communication Engineering, M. S. Ramaiah Institute of Technology, Bengaluru His areas of interest are Information Theory and Coding, AntennaDesign, SignalProcessingand Machine Learning.

Divya M.N completed her B.E. in Electronics and Communication Engineering from Bangalore University in 2001, Masters in Electronics from VTU Belgaum in 2007. She is currently pursuing a Ph.D. in Aerospace Electronics under VTU, Belgaum. Presently she is working as Assistant Professor in the school of Electronics & Communication Engineering, Reva University Bengaluru, since 2011. Her areas of interest are Aerospace Electronics, Signal Processing and Machine Learning.

Anirudha Rao Completed B.E. in Electronics and Communication Engineering from Mumbai University. He is presently pursuing M. Tech in Digital Electronics and Communication at M. S. Ramaiah Institute of Technology, Bengaluru.