International Journal of Electrical and Computer Engineering (IJECE)
Vol. 15, No. 2, April 2025, pp. 2513~2522
ISSN: 2088-8708, DOI: 10.11591/ijece.v15i2.pp2513-2522

Efficient power optimized very-large-scale integration architecture of proportionate least mean square adaptive filter

Gangadharaiah Soralamavu Lakshmaiah[1], Narayanappa Chikkajala Krishnappa[2], Poornima Golluchinnappanahalli Ramappa[3], Divya Muddenahally Narasimhaiah[4], Umesharaddy Radder[5], Chakali Chandrasekhar[6]
[1]Department of Electronics and Communication Engineering, Cambridge Institute of Technology, Bangalore, India
[2]Department of Medical Electronics Engineering, M.S. Ramaiah Institute of Technology, Bangalore, India
[3]Department of Electronics and Communication, Sri Venkateshwara College of Engineering, Bangalore, India
[4]School of Electronics and Communication Engineering, REVA University, Bangalore, India
[5]Department of Electronics and Communication Engineering, East West Institute of Technology, Bangalore, India
[6]Department of Electronics and Communication Engineering, Sri Venkateswara College of Engineering, Tirupati, India

Article Info

Article history:

Received Jul 7, 2024
Revised Nov 25, 2024
Accepted Dec 2, 2024

Keywords:

Delayed least mean square algorithms
Delayed wavelet μ-law proportionate least mean square
Field-programmable gate array
Filter
Least mean square
Proportionate least mean square
Very-large-scale integration

ABSTRACT

The focus on power optimization in embedded systems is especially important for embedded applications since it has brought in many methods and factors that are necessary for developing systems that are both power- and area-efficient. In contrast to the current delayed wavelet μ-law proportionate least mean square (DWMPLMS) and delayed least mean square (DLMS) algorithms, this work offers the development of adaptive filters based on the least mean square (LMS) method, which improves power and timing performance. In order to improve area and time efficiency, the proportionate least mean square (PLMS) algorithm's architecture has been modified to remove delay, add a proportionate gain block, design for a fixed length, include an approximate multiplier block, and swap out standard blocks for floating-point adder and divider blocks. According to a power and temporal comparison with the DWMPLMS and DLMS algorithms, field-programmable gate array (FPGA) synthesis reduces power usage by 95% for a 32-bit filter length in PLMS when compared to the above methods.

Corresponding Author:

Divya Muddenahally Narasimhaiah
School of Electronics and Communication Engineering, REVA University
Kattigenahally, Yelahanka, Bangalore, India
Email: divya.mn@reva.edu.in

1. INTRODUCTION

Adaptive filters adjust their transfer functions based on optimization algorithms to adapt to changes in the operating environment, making them effective for sparse system identification. Among these, the least mean square (LMS) adaptive filter is widely used due to its simplicity, avoiding correlation functions and matrix inversions while offering good convergence performance. However, LMS has limitations, including a feedback error time lag that hinders pipeline implementation at high sampling rates and sensitivity to input scaling, complicating learning rate selection.

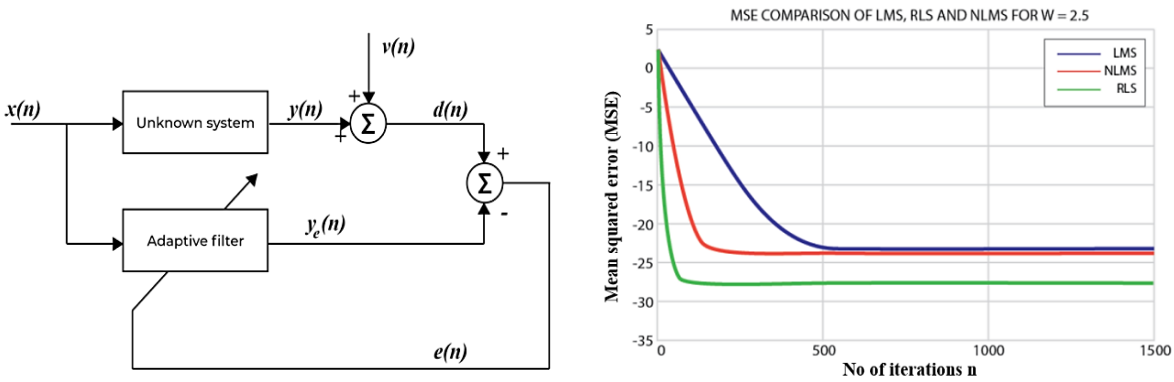
The delayed least mean square (DLMS) architecture in [1] demonstrated effective error convergence through MATLAB® Simulink modeling. The proportionate least mean square (PLMS) architecture in [2] further clarified PLMS fundamentals, enabling enhanced simulations. Although the novel design in [3] achieved power and timing efficiency, it required more area. This inspired our design, which optimizes area at a slight timing complexity cost.

Replacing multipliers with logarithmic and anti-logarithmic computations, as discussed in [4], improved time efficiency but increased power consumption. Thus, we adopted an 8-bit Vedic multiplier from [5]–[12] for better latency and power performance. Adder selection, informed by [13]–[20], and insights from systolic architectures in [21]–[29] contributed to our design's faster convergence. Studies [30], [31] highlighted the accuracy and wide range benefits of floating-point arithmetic, guiding the efficient implementation of floating-point operations in our design.

2. RESEARCH METHOD

The block diagram shown in Figures 1 (a) and 1(b) represents the block diagram of the adaptive filter as an unknown system identifier and convergence graph for different algorithm respectively. Both the adaptive filter as well as the unknown system are given the same inputs. The output that occurs across the unknown system will be the desired signal $d(n)$.

The input vector $U(n)$ is the result of further encoding the input provided by the adaptive filter into digital binary data. In the filter, the tap length determines the filter's order. Simple convolution is used to calculate the adaptive filter's output, which will initially have some unknown weights. If there is a discrepancy between this output and the intended output, it is passed back to the weight update block to provide new coefficients.



(a) (b)

Figure 1. Analysis and performance evaluation of the adaptive filter system: (a) block diagram of the adaptive filter illustrating the key components and signal flow and (b) convergence graph comparing the performance of different algorithms in terms of error reduction over iterations

This process continues to happen till the error signal ideally goes down to zero. If the error signal is zero it implies that:

a. The output of the adaptive filter is same as that of the output of the unknown system, *i.e.* $Y(n) = d(n)$ (because $e(n) = Y(n) - d(n)$).
b. If $Y(n) = d(n)$ it suggests that the adaptive filter is producing the same output as the unknown system for a given input and hence the coefficients of both the unknown system and the adaptive filter are the same. So therefore, the adaptive filter is said to have identified the unknown system under test.

The mathematical equations that will be used to find out the filter output, the error signal and the updated weights are as given in Table 1.

Table 1. Adaptive filter equations

Function	Equation
Input vector	$X(k) = [x(k), x(k-1), \ldots, x(k-L+1)]T$
Filter output	$Y(k) = xT(k)W(k)$
Error signal matrix	$E(k) = d(k) - Y(k)$
Updated weights	$W(k+1) = W(k) + \beta g(k)X(k)E(k)$
Identity matrix	$g(k) = I$

Note: $w(k)$ is set of current weights, β is adaptive step size, and $g(k)$ is gain matrix

The novel implementations on our design would be: i) A proposed floating point module approach for the PLMS register transfer level implementation and ii) Implementing an area efficient architecture for PLMS algorithm based adaptive filter implementation on FPGA's. The PLMS update equation is given by (1):

$$w(n+1) = w(n) + G(n)u(n)e(n) \tag{1}$$

The Pt-NLMS family of algorithms iteratively estimate the filter weights

$$w(n) = [w0(n), w1(n), \dots, w(L-1)(n)]^T \tag{2}$$

The Gain matrix $G(n)$ is explained in (3),

$$G(n) = diag\left(g_0(n), g_1(n), \dots., g_L - 1\ (n)\right) \tag{3}$$

and a gain factor $g_i(n)$ is assigned to the ith tap in proportion to $|w_i(n)|$

$$g_i(n) = \frac{w(n)}{\frac{1}{L}\sum_{L-1}^{i=0} wi(n)} \tag{4}$$

For the simplified PLMS algorithm, $\gamma_i(n)$ for each tap is evaluated as

$$\gamma_i(n) = F[|w_i(n)| + \rho] \tag{5}$$

and

$$F[|w_i(n)|] = |w_i(n)| \tag{6}$$

The Pt-LMS algorithm simplifies its predecessors by omitting weighted normalization and simplifying gain factor evaluation, with a small constant p ensuring minimum gain for inactive coefficients and reducing time complexity. These changes improve area and power efficiency, but high time complexity remains due to repeated gain matrix and weight updates. Delayed adaptation addresses this issue, leveraging the unchanged error gradient despite delays.

$$w(n+1) = w(n) + \mu G(n-M)u(n\ -\ M)e(n-M) \tag{7}$$

When we compare the results of Pt-LMS with other LMS algorithm we observe that the convergence performance of Pt-LMS is comparatively better than that of other LMS algorithms and its convergence performance can be improved further. It is also observed that Pt-LMS is real time flexible and robust. Hence, we decided to move forward with PLMS.

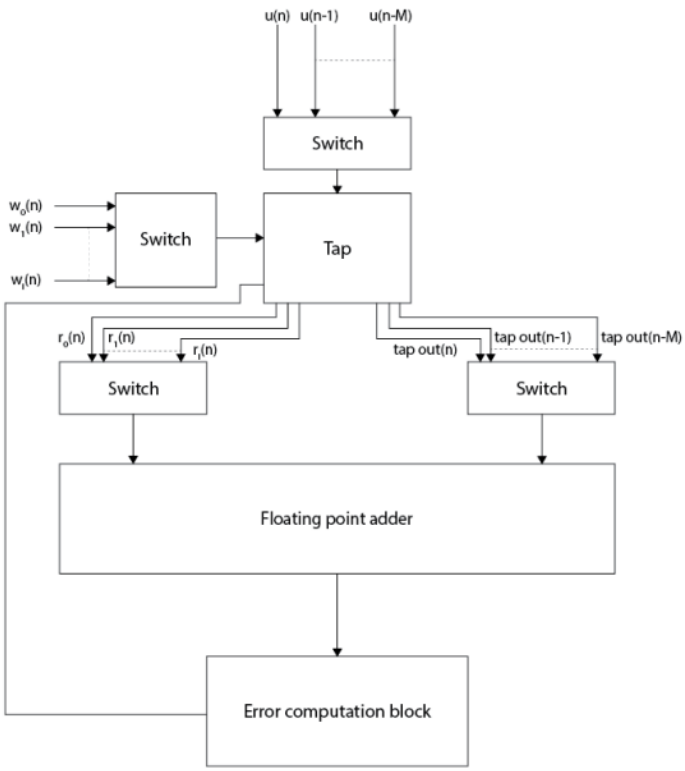
3. ARCHITECTURE

3.1. Proposed PLMS architecture

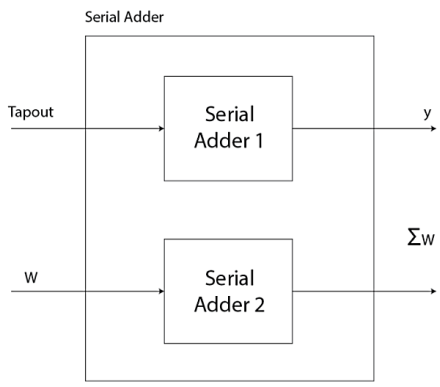
Figure 2 shows the proposed PLMS architecture and floating-point adder block respectively. As illustrated in Figure 2(a); to implement pipelining, the number of taps increases with the order of the filter, which significantly impacts the area. Additionally, switches are placed after every two taps to manage the tap-out and gamma function at the corresponding clock phases. While these switches help reduce timing complexity, their large number contributes to increased area. Instead of connecting the regressor input and initial weights directly to the taps, they are routed through a switch. Depending on the clock phase, each filter coefficient and input pass through a floating-point multiplier, which accelerates the multiplication process and generates the tap-out (n) and gain function. These outputs are directed to switch 2. After all the outputs are generated from a single tap, switch 1 is activated during one clock phase and switch 2 in the next. When switch 1 is active, the adder sums the tap-outs, providing the adaptive filter output, and when switch 2 is active in the following phase, the sum of the weighted functions is obtained.

The inputs to the serial adder block, as shown in Figure 2(b), come from a switch that receives partial filter outputs and gain factors from the corresponding taps over four successive switching cycles. The switch sends these tap-outs to the adder only after it has received the partial outputs from all N input samples. Since the inputs are 32-bit floating-point values, the tap-out, which is the product of the input x and weight W, also results in a 32-bit floating-point value. Therefore, a 32-bit floating-point adder is required to combine the partial filter outputs and generate the complete filter output. This architecture employs a 32-bit floating-

point adder (FPA) due to its ability to handle a wide range of numbers with high precision. Since the IEEE 754 32-bit floating-point format separates the exponent and mantissa, adding two floating-point numbers involves adding their mantissas, with a specific number of shifts applied to the mantissa of the number with the smaller exponent.



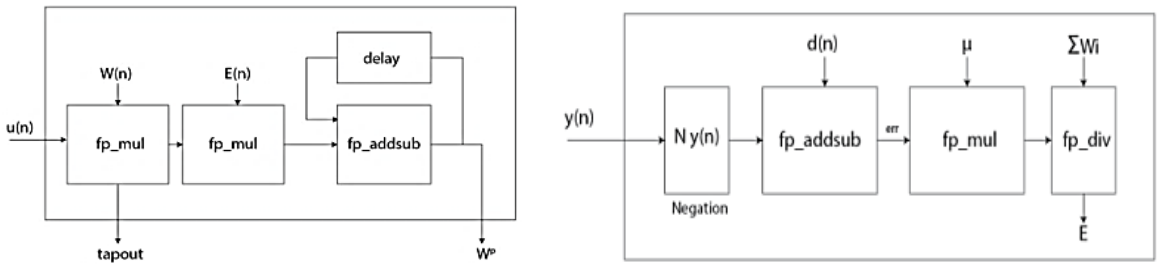
(a)



(b)

Figure 2. Representation of the proposed architecture components: (a) proposed PLMS architecture showcasing the pipeline and control mechanism and (b) floating-point adder block used for accurate computation of weights and updates

Figures 3(a) and 3(b) represent the tap block and error computation block respectively. The tap block architecture computes the gain factor and partial filter output using a 32-bit floating-point multiplier. The tap also receives an input, $E(n-M)$, from the error computation block. The system output, $y(n-i)$, is the sum of the tap outputs from the current and previous iterations. The error signal is subtracted from the actual output and fed back to the tap block, updating the weights for the next iteration. A serial adder sums the L previous tap outputs and weights for the gain factor and system output.



(a) (b)

Figure 3. Illustration of the key components in the adaptive filter architecture: (a) tap block for managing the input data and weight updates and (b) error computation block for determining the error signal to refine the filter's performance

3.2. Floating point modules architecture

We propose efficient floating-point (FP) arithmetic units, including FP addition, subtraction, multiplication, and division, for fast computation using single-precision IEEE 754 format. This 32-bit format includes a 1-bit sign (1 for negative, 0 for positive), an 8-bit exponent, and a 23-bit mantissa for high-range data representation.

$$X = (-1^{sign} x 2 \exp(1.man)) \tag{8}$$

where $sign$ is sign of the number X, exp is exponential value of a number, and man is mantissa value of the number.

3.2.1. Floating point adder/subtractor

Figure 4 shows the floating-point adder where the larger exponent is taken as common, and the mantissa of the smaller exponent is left-shifted by the exponent difference before addition. The mantissas are added, and any carry is added to the exponent while left-shifting the result. The sign bit is determined by XOR-ing the input signs. The process involves an exponent comparison block to align exponents, a mantissa block for addition/subtraction based on sign bits, and a normalization block for adjusting the final 32-bit result. Normalization shifts the mantissa based on carry/borrow, producing the final exponent and mantissa values.

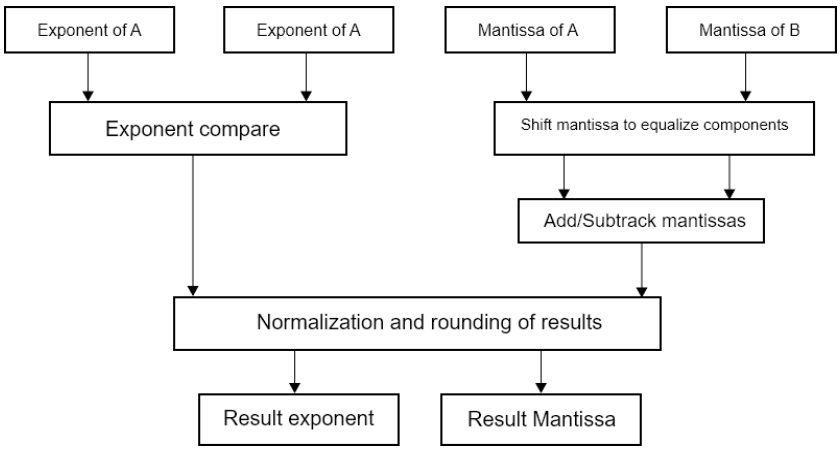


Figure 4. Floating point adder

3.2.2. Multiplication and division of two floating point value

Figure 5 shows the floating-point multiplier and floating-point divider respectively. The sign bit is obtained by XOR-ing the input sign bits. The mantissas are processed through a 23-bit adder and a Vedic multiplier for accuracy, with their results summed by another 23-bit adder. The combined carry from both adders is used in a shifter block and added to the exponent sum, yielding the final result's exponent and

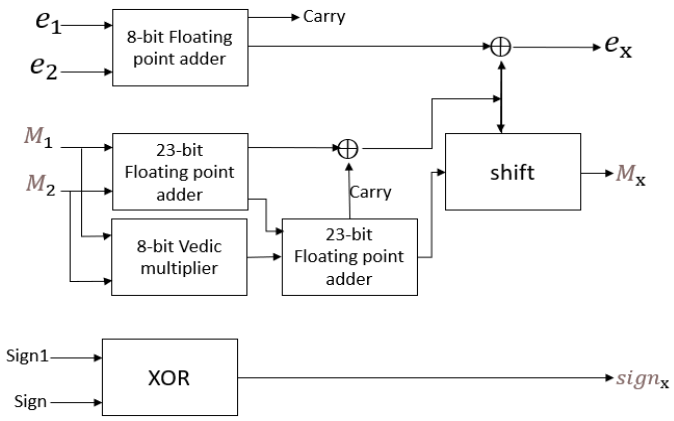
mantissa. Complex multiplication of two 32-bit floating-point (FP) values are broken down into simple module design as shown in Figure 5(a).

$$X1 = (-1)^{sign1} * (2^{e1}) * (1.m1) \tag{9}$$

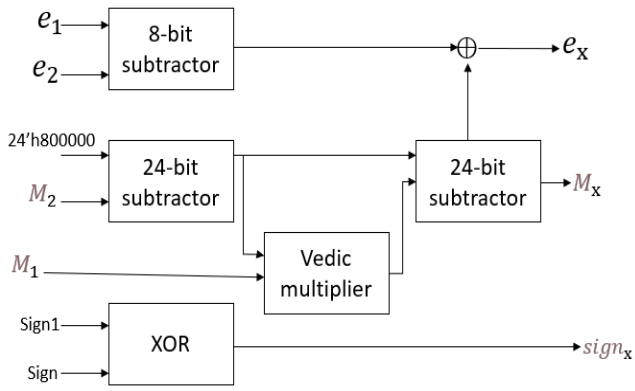
$$X1 = (-1)^{sign2} * (2^{e2}) * (1.m2) \tag{10}$$

where $X1$, $X2$ are values expressed in form of single precision floating point format; $sign1, sign2$ are the sign of number X; $e1$, $e2$ are the exponential value; and $m1$, $m2$ are the mantissa value.

Complex division of two 32-bit floating-point values is simplified as shown in Figure 5(b). The sign bit is obtained by XOR-ing the input signs. The second mantissa is subtracted from $24'h80000024$ and the result is multiplied with the first mantissa using a Vedic multiplier. The outputs are added to form the result's mantissa, while the carry is added to the exponent block for the final result's exponent.



(a)



(b)

Figure 5. Key computational blocks of the floating-point unit: (a) Floating-point multiplier for efficient multiplication operations and (b) Floating-point divider for precise division computations

4. RESULTS AND DISCUSSION

4.1. MATLAB results

This system was built to analyze any type of input signal which is generated by a signal generator, the same signal is been passed to unknown system and adaptive system and results in error convergence effectively. Once we achieved satisfactory results, we started the synthesis. The Simulink diagram can be seen in Figure 6(a). Figure 6(b) presents the Simulink model results for the adaptive filter, showing the desired signal, the filter's output, and the error signal. The third graph demonstrates error convergence over time. Figure 6(c) shows the register transfer language (RTL) schematic of PLMS algorithm using Libero SOC version 11.9 with FPGA A3P1000L from ProASIC3L series.

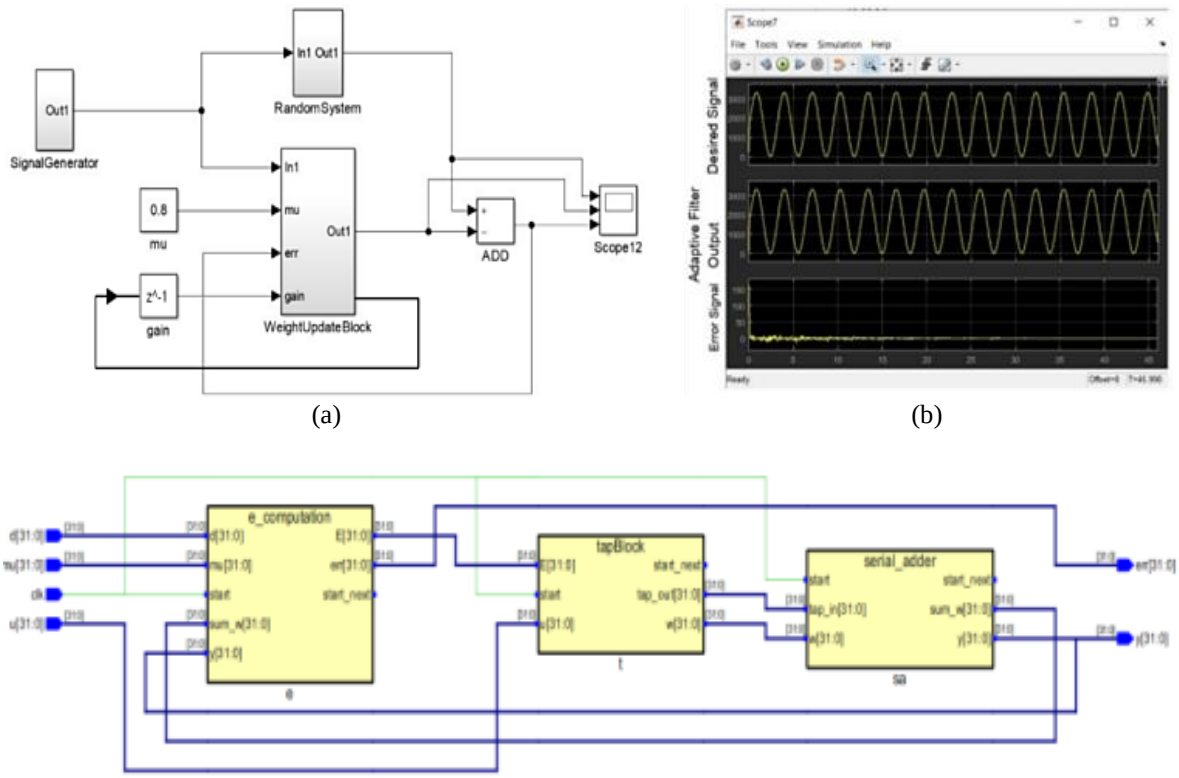


Figure 6. Comprehensive analysis and implementation of the proposed PLMS architecture: (a) Simulink model of the proposed PLMS architecture, (b) Simulink result for sine wave demonstrating the system response, and (c) RTL schematic representing the hardware implementation of the PLMS algorithm

4.2. Synthesis results

Tables 2 and 3 represents the characteristics of proposed PLMS and comparison with existing architectures respectively. Figures 7(a) and 7(b) show the graphical comparison power-delay comparison and timing comparison with referred algorithms with existing DLMS and DWMPLMS architectures respectively. Table 3 shows that the proposed design reduces power consumption by up to 95%, with savings of 95% and 88% compared to the DWMPLMS and DMPLMS architectures, respectively. This efficiency stems from replacing the logarithmic multiplier with a floating-point Vedic multiplier, which enhances power efficiency. The pipelined tap block design further improves timing, making the proposed architecture 30 times faster than logarithmic methods and the DLMS design. Fixed-point computations in DLMS are less efficient, with the proposed floating-point blocks achieving 84% power savings.

Table 2. Characteristics of proposed PLMS

Architecture	Power (mW)	Delay (ns)
DWMPLMS (Mula *et al.* [2])	19.43	3.31
DLMS (Meher and Park [1])	10.06	3.28
DLMS (Fan *et al.* [21])	12.56	3.27
DMPLMS [2]	14.13	3.31
Proposed (PLMS)	1.067	104.45

Table 3. Comparison with existing architectures

Metric	Value
Number of slices LUT's used	4094
Number of IO cells used	161
Device power consumption	1.067 mW
Operating frequency/timing	9.5 MHz
Operating timing	104 ns

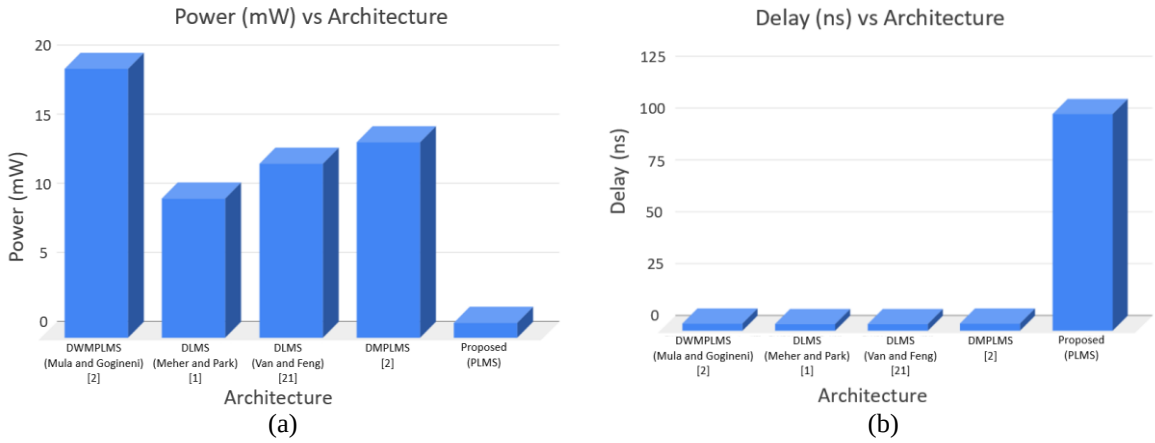


Figure 7. The graphical: (a) power comparison and (b) timing comparison, with referred algorithms

5. CONCLUSION

This paper proposes an adaptive filter design that leverages the PLMS algorithm, which is applied to a 32-bit filter length. The PLMS algorithm is selected due to its ability to achieve a lower mean-square-deviation (MSD) compared to the traditional LMS algorithm, resulting in better performance. Additionally, PLMS offers faster convergence than the DLMS algorithm, making it a more efficient choice in terms of area, power, and timing.

The proposed design replaces the logarithmic approach in existing DWMPLMS and DMPLMS architectures with floating-point computation, a Vedic multiplier, and a proportionate gain block. A pipelined architecture in the tap block enhances efficiency, while the design includes approximate multipliers, floating-point adders, and divider blocks. FPGA synthesis shows a 92% power reduction compared to existing architectures. Future work focuses on reducing area, improving timing, and fine-tuning output performance.

REFERENCES

[1] P. K. Meher and S. Y. Park, "Area-delay-power efficient fixed-point LMS adaptive filter with low adaptation-delay," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 2, pp. 362–371, Feb. 2014, doi: 10.1109/TVLSI.2013.2239321.

[2] S. Mula, V. C. Gogineni, and A. S. 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*, vol. 26, no. 9, pp. 1750–1762, Sep. 2018, doi: 10.1109/TVLSI.2018.2828165.

[3] M. O. Sayin, N. D. Vanli, and S. S. Kozat, "A novel family of adaptive filtering algorithms based on the logarithmic cost," *IEEE Transactions on Signal Processing*, vol. 62, no. 17, pp. 4411–4424, Sep. 2014, doi: 10.1109/TSP.2014.2333559.

[4] S. Paul, N. Jayakumar, and S. P. Khatri, "A fast hardware approach for approximate, efficient logarithm and antilogarithm computations," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 17, no. 2, pp. 269–277, Feb. 2009, doi: 10.1109/TVLSI.2008.2003481.

[5] B. S. Premananda, S. S. Pai, B. Shashank, and S. S. Bhat, "Design and implementation of 8-Bit Vedic multiplier," *International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering*, vol. 2, no. 12, pp. 5877-5882, Dec. 2013.

[6] U. R, "Area, delay and power comparison of adder topologies," *International Journal of VLSI Design & Communication Systems*, vol. 3, no. 1, pp. 153–168, Feb. 2012, doi: 10.5121/vlsic.2012.3113.

[7] K. Wagner, "Analysis and design of proportionate-type normalized least mean square algorithms," *Physics*, 2001.

[8] M. D. Meyer and D. P. Agrawal, "A high sampling rate delayed LMS filter architecture," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 40, no. 11, pp. 727–729, 1993, doi: 10.1109/82.251841.

[9] S. Ramanathan and V. Visvanathan, "A systolic architecture for LMS adaptive filtering with minimal adaptation delay," in *Proceedings of 9th International Conference on VLSI Design*, 1996, pp. 286–289, doi: 10.1109/ICVD.1996.489612.

[10] P. K. Meher and M. Maheshwari, "A high-speed FIR adaptive filter architecture using a modified delayed LMS algorithm," in *Proceedings - IEEE International Symposium on Circuits and Systems*, May 2011, pp. 121–124, doi: 10.1109/ISCAS.2011.5937516.

[11] V. C. Gogineni and S. Mula, "Improved proportionate-type sparse adaptive filtering under maximum correntropy criterion in impulsive noise environments," *Digital Signal Processing: A Review Journal*, vol. 79, pp. 190–198, Aug. 2018, doi: 10.1016/j.dsp.2018.04.011.

[12] S. Mula, V. C. Gogineni, and A. S. Dhar, "Robust proportionate adaptive filter architectures under impulsive noise," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. 5, pp. 1223–1227, May 2019, doi: 10.1109/TVLSI.2019.2892383.

[13] S. Mula, V. C. Gogineni, and A. S. Dhar, "Algorithm and architecture design of adaptive filters with error nonlinearities," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 9, pp. 2588–2601, Sep. 2017, doi: 10.1109/TVLSI.2017.2702171.

[14] S. Haykin and B. Widrow, *Least-mean-square adaptive filters*. Wiley, 2003.

[15] H. Deng and M. Doroslovački, "Improving convergence of the PNLMS algorithm for sparse impulse response identification," *IEEE Signal Processing Letters*, vol. 12, no. 3, pp. 181–184, Mar. 2005, doi: 10.1109/LSP.2004.842262.

[16] K. C. Ho and S. D. Blunt, "Adaptive sparse system identification using wavelets," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 49, no. 10, pp. 656–667, Oct. 2002, doi: 10.1109/TCSII.2002.807263.

[17] P. K. Meher and S. Y. Park, "Low adaptation-delay LMS adaptive filter part-I: Introducing a novel multiplication cell," in *Midwest Symposium on Circuits and Systems*, Aug. 2011, pp. 1–4, doi: 10.1109/MWSCAS.2011.6026642.

[18] P. K. Meher and S. Y. Park, "Low adaptation-delay LMS adaptive filter part-II: An optimized architecture," in *Midwest Symposium on Circuits and Systems*, Aug. 2011, pp. 1–4, doi: 10.1109/MWSCAS.2011.6026643.

[19] Y. Yi, R. Woods, L. K. Ting, and C. F. N. Cowan, "High speed FPGA-based implementations of delayed-LMS filters," *Journal of VLSI Signal Processing Systems for Signal, Image, and Video Technology*, vol. 39, no. 1-2 SPEC.ISS., pp. 113–131, Jan. 2005, doi: 10.1023/B:VLSI.0000047275.54691.be.

[20] L. Da Van and W. S. Feng, "An efficient systolic architecture for the DLMS adaptive filter and its applications," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 48, no. 4, pp. 359–366, Apr. 2001, doi: 10.1109/82.933794.

[21] L. Fan, C. He, D. Wang, and L. Jiang, "Efficient robust adaptive decision feedback equalizer for large delay sparse channel," *IEEE Transactions on Consumer Electronics*, vol. 51, no. 2, pp. 449–456, May 2005, doi: 10.1109/TCE.2005.1467986.

[22] S. Attallah, "The wavelet transform-domain LMS algorithm: a more practical approach," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 47, no. 3, pp. 209–213, Mar. 2000, doi: 10.1109/82.826747.

[23] A. H. Sayed, *Fundamentals of adaptive filtering*. USA: Wiley: Hoboken, 2003.

[24] V. Paliouras and T. Stouraitis, "Logarithmic number system," *Arithmetic Circuits for DSP Applications*. Wiley, pp. 237–272, Aug. 2017, doi: 10.1002/9781119206804.ch7.

[25] P. K. Meher and S. Y. Park, "Critical-path analysis and low-complexity implementation of the LMS adaptive algorithm," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 61, no. 3, pp. 778–788, Mar. 2014, doi: 10.1109/TCSI.2013.2284173.

[26] W. Ma, D. Zheng, Z. Zhang, J. Duan, and B. Chen, "Robust proportionate adaptive filter based on maximum correntropy criterion for sparse system identification in impulsive noise environments," *Signal, Image and Video Processing*, vol. 12, no. 1, pp. 117–124, Jun. 2018, doi: 10.1007/s11760-017-1137-0.

[27] 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," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 28, no. 1, pp. 188–197, Jan. 2020, doi: 10.1109/TVLSI.2019.2935399.

[28] L. K. Ting, R. Woods, and C. F. N. Cowan, "Virtex FPGA implementation of a pipelined adaptive LMS predictor for electronic support measures receivers," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 13, no. 1, pp. 86–94, Jan. 2005, doi: 10.1109/TVLSI.2004.840403.

[29] M. Cornea, "IEEE 754-2008 decimal floating-point for intel®architecture processors," in *Proceedings - Symposium on Computer Arithmetic*, Jun. 2009, pp. 225–228, doi: 10.1109/ARITH.2009.35.

[30] L. Daoud, D. Zydek, and H. Selvaraj, "A survey on design and implementation of floating point adder in FPGA," in *Advances in Intelligent Systems and Computing*, vol. 1089, Springer International Publishing, 2015, pp. 885–892.

[31] A. Ehliar, "Area efficient floating-point adder and multiplier with IEEE-754 compatible semantics," in *Proceedings of the 2014 International Conference on Field-Programmable Technology, FPT 2014*, Dec. 2014, pp. 131–138, doi: 10.1109/FPT.2014.7082765.

BIOGRAPHIES OF AUTHORS

Gangadharaiah Soralamavu Lakshmaiah obtained his M.Tech. in digital electronics and advanced communication from KREC, Surathkal, obtained Ph.D. in the area of VLSI signal processing from VTU Belagavi. Presently he is working as professor, Department of Electronics and Communication Engineering, CIT, Bengaluru and Principal Scientist CCCIR, CIT. His areas of interest are RTL design and design verification, VLSI signal processing and machine learning. He can be contacted at email: gdhar75@gmail.com.

Narayanappa Chikkajala Krishnappa received Ph.D. from VTU, Belagavi. He is currently working as professor and H.o.D at the Department of Medical Electronics, M.S. Ramaiah Institute of Technology, Bengaluru. His research interests include signal and image processing and control systems. He is the member of ISTE, IETE and BMESI. He is also a fellow at The Institution of Engineers (India). He can be contacted at email: c_k_narayanappa@msrit.edu.

Poornima Golluchinnappanahalli Ramappa received her M.Tech. in electronics from BMSCE, Bangalore, Ph.D. from VTU Belagavi. She is presently working as professor and dean in Department of Electronics and Communication Engineering, Sri Venkateshwara College of Engineering. Bangalore. Her research interests are analog and digital VLSI. She can be contacted at email: poornima.gr_ece@svceengg.edu.in.

Divya Muddenahally Narasimhaiah received her M.Tech. in electronics from BMSCE, Bangalore, obtained Ph.D. from VTU, Belagavi. Presently working as assistant professor in the School of Electronics and Communication Engineering, Reva University Bengaluru, her areas of interest are aerospace electronics, signal processing and machine learning. She can be contacted at email: draophd@gmail.com.

Umesharaddy Radder obtained his M.Tech. in VLSI design and embedded systems, from VTU Belagavi, and Ph.D. in the area of VLSI design from VTU Belagavi in 2006 and 2018 respectively. Presently, he is working as associate professor in the Department of Electronics and Communication Engineering, East West Institute of Technology, Bengaluru. His areas of interest are VLSI design for communication, embedded system design with RISCV, deep learning and machine learning. He can be contacted at email: umeshradder@gmail.com.

Chakali Chandrasekhar obtained his M.Tech. in digital electronics from VTU Belagavi, and Ph.D. in the area of image fusion and compression from SV University, Tirupathi in 2003 and 2014 respectively. Presently he is working as professor in the Department of Electronics and Communication Engineering, Sri Venkateswara College of Engineering, Tirupati, Andhra Pradesh, India. His areas of interest are image processing and VLSI. He can be contacted at email: chandra.bti2009@gmail.com.