Indian Journal of Natural Sciences *www.tnsroindia.org.in ©IJONS*

Vol.14 / Issue 78 / June / 2023 *International Bimonthly (Print) – Open Access* *ISSN: 0976 – 0997*

RESEARCH ARTICLE

Implementation of LMS Adaptive Filter using Distributed Arithmetic based HSCG-SCS adder

Telugu Maddileti[1], R. Ajay Kumar Reddy[2]* and K. Maheswari Devi[3]

[1]Associate Professor, Department of ECE, Malla Reddy Engineering College, Secunderabad, Hyderabad, Telangana 500100, India.
[2]PG Scholar, Department of ECE, Malla Reddy Engineering College, Secunderabad, Hyderabad, Telangana 500100, India.
[3]Assistant Professor, Department of ECE, Malla Reddy Engineering College, Secunderabad, Hyderabad, Telangana 500100, India.

Received: 20 Feb 2023 Revised: 25 Apr 2023 Accepted: 30 May 2023

***Address for Correspondence**
R. Ajay Kumar Reddy
PG Scholar, Department of ECE,
Malla Reddy Engineering College,
Secunderabad, Hyderabad,
Telangana 500100, India.
E. Mail: harryhari127@gmail.com

ABSTRACT

Modern wireless communication systems have tightened the requirements of adaptive beam formers when implemented on Field Programmable Gate Array. The set requirements imposed additional constraints such as designing a high throughput, low complexity system with fast convergence and low steady state error. Recently, a parallel multi-stage least mean square (LMS) structure is proposed to mitigate the listed constraints. The LMS is a two stages LMS operating in parallel and connected by error feedback with Half Sum Carry Generation based Sum Carry generator (HSCG-SCG) adder. This was accomplished by replacing the array multiplier by distributed arithmetic (DA) memory units. The DA based HSCG-SCG adders are used by this design in order to successfully complete this task. The memory capacity has been lowered by a factor of at least four as a direct result of this, although the filter order has stayed the same. The simulation results shows that the proposed method resulted in reduced area, delay, power consumption as compared traditional adaptive filter.

Keywords: Adaptive Filter, Distributed Arithmetic, Least Mean square, Sum Carry generator.

INTRODUCTION

The least mean square technique has higher stability and quicker convergence than other algorithms, making it ideal for adaptive filters, which have a broad variety of applications but are most often employed for DSP. This architecture features a very simple register layout. [1]. This feature is explained in this article, along with its potential to support large input sampling rates. Constructing a DLMS adaptive filter that has low-adaptation latency and an efficient pipelined design is conceivable, and doing so will allow for increased convergence performance. When doing calculations using Vedic Math, you can find it helpful to make use of a multiplier that has 8 bits. The DSP [2] relies heavily on adders as a fundamental building block. The DLMS adaptive filter has the capability to minimize the complexity of the registers, the capacity to enable a speedier convergence, a very little adaptation delay, and does not need a pipelining approach. All of these benefits come without the requirement for a pipelining technique. Zero adaptation delay The DLMS method offers superior performance compared to those of other algorithms, in addition to having no adaption delay. The DLMS algorithm needs the least amount of energy (EPS) [3] per given sample, and it also needs the least amount of space compared to other designs. In applications involving electroencephalograms (EEGs), the DLMS method may also be used for the cancellation of interference [4].

Implementing the LMS adaptive algorithm and the changes to it may also be done in a variety of different methods beyond those that have been described. Only a few of the many possible applications of these methods for DSP have been discussed in this article [5], despite the fact that they may be used extensively for those applications. Since the multiplier might cause bottlenecks, the DA method may also be used to replace the multiplier block that is used by the LMS algorithm. This is because the multiplier can slow things down. To simulate the operation of multiplication [6]. Using the radix-8 Booth method, we were able to significantly cut down on the quantity of incomplete products that the DA design produced. In order to do serial bit calculations, DA is used. Because of this, the LMS adaptive filtering approach may use DA throughout the process of developing the VLSI architecture for it. In DA, the limited products of the input model and the filter factors are stored in "two" different LUTs. These LUTs are separate from one another. In direction to realize a higher near of performance, the LUTs are multiplexed after being accessed and their contents have been added together [7].This will happen after the error has been calculated. This is going to happen once they have been removed from the system. By integrating the functionality of two adaptive filters, it is possible to create a DA-based low-complexity pipelined least-mean-square filter. Because of this, the amount of labour needed to install the filter will be reduced. The total step heights are broken out as follows: The convergence performance may be adjusted using adaptive filters; however, in this case, the two ADFs that were previously utilised have been replaced with a single DA-based ADF. Because odd multiples are symmetrical to one another, an adder tree may be used to add offset words [8]. Only a minimal number of adders are required for the creation of odd multiples from longer words, and all that is required is a single offset adder tree to complete the process. The PAS and PSA are the names given to each of these methodologies (BLMS-ADF).When contrasted with the PSA approach, the PAS methodology results in a shorter critical path [9].

Literature Survey

Khan *et al.* [10] developed three designs built for pipelined DA based LMS adaptive filter with optimal complexity. These architectures were created with the goal of reducing the amount of computational work required. Mixtures of input samples are implemented on hardware using the offset-binary-coding (OBC) technique may aid in reducing the complexity of the structures that are proposed. Nevertheless, some outputs that are not OBC are made, and later on, when the clock is being cycled through its initial phases, these outputs are removed from the calculation that determines whether or not there was an error. This occurs when the clock is cycling through its earliest stages. These implementations are all totally different from one another. The phrase that came before this one mentioned all of these things. In summary, the results of the studies show that using the 32nd order filter may result in power-per-throughput cost reductions of 68.4, 47.33 percent, 33.72 percent, and 43.19% respectively.

Within the parameters of their research, Khan *et al.* [11] proposed the use of two-optimized partial look-up table (LUT) designs as a low-complexity implementation of DA based block LMS (BLMS) adaptive filtering. It has been shown in this article that making use of the redundancies that are present between the partial inner products and the sliding window input block is an essential component of a unique strategy for optimising results. It has been shown that this approach is one of the primary components of the method. A comparison between the two approaches may be done, and it has been observed that the PAS technique results in a shorter critical-path-delay than the PSA methodology does. This is one way in which the two methodologies can be contrasted with one another. Each of these percentages is lower than the finest job that has previously been done. According to these findings, the PAS-based BLMS ADF has a more limited coverage area and generates less power than the PSA-based BLMS ADF does. Both of these results are better than the most current and most significant work that has been done up to this point.

The sign–magnitude representation of the error signal was used by Meo *et al.* [12] to suggest the usage of approximately fixed-width and static section multipliers in the construction of Delayed LMS (DLMS) adaptive filters. These multipliers were used in the construction of DLMS adaptive filters. These multipliers were used throughout the development of DLMS adaptive filtering systems. Both of these strategies are intended to minimise the approximation error. These traits, when combined, are supposed to reduce the amount of inaccuracy introduced by the approximation. Both of these qualities were developed with the intention of bringing the level of inaccuracy brought on by the approximation down to a more acceptable level. When representing the error signal, using a symbol–magnitude representation may assist to decrease the amount of switching action, which in turn can help to reduce the amount of power that is lost. This is place at the period in which the filter is learning. In conclusion, the adaptive filters that were investigated were able to surpass the state-of-the-art by reducing the amount of space that they occupied and the amount of power that they consumed in comparison to the conventional implementation by up to -18.0% and -64.1%, respectively, while still maintaining their capacity for learning.

The work that Mula *et al.* [13] did allowed for the Pt-NLMS algorithms to be realised as physical implementations, bringing them one step closer to reality. Many other reformulations have been proposed in attempt to make the original Pt-NLMS algorithms suitable for realtime VLSI implementations. This has been done in an effort to improve performance. These rewrites are attempts to make the algorithms more straightforward. Both of these algorithms are known as the delayed -law proportionate LMS (DMPLMS). These two algorithms are together known as the delayed-law proportionate LMS (DMPLMS). In conclusion, the findings of the simulations demonstrated that the performance loss that would be caused by implementing the recommended reformulations would be, at most, insignificant. This was demonstrated by the fact that the loss in performance would not even reach the threshold of being noticeable.

Ahmad *et al.* [14] developed a composite architecture that uses DA to replace the bottleneck multiplier with memory units that store Partial Products (PPs) to simulate multiplication. This design is referred to as an imitation of multiplication. Because of the design that was advised, considerable gains have been made with regard to output, critical route latency, power consumption, and use of FPGA properties. The last approach, which is referred to as the Half-Delay technique, decreased the latency by a factor of 2, while concurrently raising the power and area quality to some degree. The authors Esposito, *et al.* [15] built state-of-the-art approximation multipliers and evaluated the effect that these multipliers had on the LMS algorithm's performance. In addition, an unique approximation multiplier that has been presented, which has precision that can be tweaked at the time of design in order to better fit to the application situation. We are able to analyse the power vs. quality trade-off of the LMS approximation filters because of the results of the implementation in the 28-nm CMOS technology. These filters were examined in two different applications.

Shen *et al.* [16] used the conventional LMS method with a fixed number of steps. The traditional fixed-step LMS algorithm has the benefit of being straightforward and easy to put into practise. On the other hand, one of its inherent drawbacks is that it raises the step factor in order to accelerate the convergence, which in turn results in a significant increase in the steady-state error. An enhanced LMS algorithm that is based on the nonlinear relationship

function between x(k) and (k) is provided on the basis of the findings of an investigation of the performance of the LMS algorithm. Comparison of the v LMS approach is shown to be achievable, according to the findings of the theoretical study as well as the simulations. It is beneficial to overcome the shortcomings that are inherent to the LMS algorithm. F.Lavalle-Aviles and colleagues [17] proposed a completely low-pass filter (LPF) with programmable cutoff frequency, adaptive powerfully differential is what the FD designation refers to, while low voltage is what the LV designation refers to. These features were included into the design of the filter. In order to fulfil the performance requirements of the stringent filter, a low-voltage field-depletion (LV FD) amplifier with feedforward gain-boosting implementation is used.

Vinitha and others [18] made a proposal. An innovative new DA design for the adaptive FIR filter. The LMS algorithm is cast-off in order to bring the filter's weights up to date. We were able to increase the filter's area efficiency as well as its power efficiency by combining the DA approach and the CSD number in this design. LUT-less DA calculation is done. Instead, a calculation that is based on registers is employed. After all is said and done, the proposed architecture is given a VHDL coding. After that, it is simulated, synthesised, and ultimately applied in a Virtex FPGA device. The trick consumption report for the construction reveals that the structure's improved competence in terms of both space and power compares well to designs from the past. Prasad *et al.* [19] came up with the idea for an adaptive filter that is founded on the LMS method and has a lower computation delay. This filter takes up less space than other filters and requires less memory. To begin computing each filter partial product of a FIR filter, the method that has been suggested takes use of a single multiplier as its starting point. After that, a time-shared architecture for LMS-ADF is used in order to calculate the coefficient increment term, and this multiplier is put to use in order to do so. The LMS algorithm is what's utilised to slowly ratchet up the weights, however. The technique consists of using a single multiplier for one purpose during one half cycle of the process and then again for another reason during the second half cycle of the process. Finally, according to the findings, it has been determined that the suggested design would utilise about half the area of the traditional design while simultaneously boosting the operating speed in an efficient manner.

Student *et al.* [20] came up with a straightforward implementation that is effective. [20] In the delayed LMS method, the conventional multiplier is what makes it possible run more slowly; for this reason, the most recent high speed Because of the great convergence rate that it has, the Vedic Multiplier is the multiplier of choice. In addition, the Vedic Multiplier is being researched for its ability to reduce the number of logic levels and timing levels, as well as for its potential to shorten the amount of time required for logic operations. Both of these possibilities are being examined. Adders that are as power-, size-, and delay-efficient as possible are put to use in applications that include DSP in order to reduce the amount of power that is needed.

Proposed Method

LMS Adaptive Filter

The following is an example of a potential encoding for the N-tap involvement signal vector, which is referred to as S(k).

$$S(k) = [s(k), s(k-1), \dots, s(k-N+1)]^T \quad (1)$$

where the letter s denotes an input signal at the kth time occurrence and the symbol T denotes a transposition of the vector. k is the number of times the signal will occur. An input signal at the kth instance is denoted by the symbol s(k). The value of k represents the total number of times the signal has been seen.

One possible way to write down a representation of the signal that is produced by an executive filter is as follows:

$$y(k) = ST(k)W(k) \quad (2)$$

where the vector is denoted by W(k)that corresponds to the kth coefficient and N is the entiresum of factors in the expression.

$$W(k) = [w_0(k), w_1(k), \cdots, w_{N-1}(k)]^T \quad (3)$$

For those interested in gaining even more comprehension on this subject, the wi k is the i-th tap coefficient of the adaptive filter.

The LMS process developed by Widrow and colleagues in 1975 may be expressed as

$$W(k+1) = W(k) + 2\mu e(k)\, S(k) \quad (4)$$

wherever e(k), which stands for an error signal, d(k), which stands for a step-size parameter, and d(k), which stands for the desired signal, respectively, represent ablunder signal, a step-size restriction, and the intended signal. Both the pace of convergence and the accuracy of the estimate are determined by the step-size parameter, which is responsible for generating the value. The following procedures are required in order to get the error signal:

$$e(k) = d(k) - y(k) \quad (5)$$

Figure 1 presents an overview of the LMS adaptive filter's primary architectural components. The signal for the filter input, which is indicated by the notation s(k), is sent across the delay line is moved and shifted in the appropriate direction for each and every sampling event that takes place. The delayed input signal is generated by the taps on the delay line, which directly correspond to the delay components in the circuit. The number of taps serves as a determinant of the extent to which the delay is applied. The LMS adaptive filter calculates the value of the filter's output by adding the products that are obtained by multiplying the outputs of the taps by the coefficients that are connected with those taps. These products are generated by multiplying the outputs of the taps by the coefficients that are connected with those taps. The difference between the signal that is sought and the signal that is formed by the filter is referred to as the "error signal," and it is referred to by that term. This discrepancy is referred to as the error signal. It's feasible to use an arrow to indicate this particular difference. First, the input signals are multiplied by the scaled fault signal in command to modify the tap coefficients. Next, the result of that multiplication is multiplied by itself in order to have the tap coefficients adjusted. This process is repeated until the tap coefficients have been adjusted.

Distribute Arithmetic based HSCG-SCS Adder

The computation of an inner product utilising a table lookup approach is the basis of the DA, an effective calculation technique. Now let's look at the interior decoration that has been finished.

$$y = a^T v = \sum_{i=1}^{N} a_i v_i \quad (6)$$

that corresponds to the constant vector of the Nth order

$$a = [a_0, a_1, \cdots a_{N-1}]^T \quad (7)$$

then the variable vector

$$v = \left[v_0, v_1, \cdots v_{(N-1)}\right]^T \quad (8)$$

The vi is presented as a representation based on B-bit static point and 2's complement in equation (8), which can also be written as,

$$-1 \le v_i < 1 \quad (9)$$

and

$$v_i = -v_i^0 + \sum_{k=1}^{B-1} v_i^k 2^{-k}, i = 0,1,\cdots,N-1 \quad (10)$$

In equation (10), vik denotes the k-th bit of vi, which may be either 0 or 1. By changing Equation (10) for Equation (6),

$$y = -\Phi(v_0^0, v_1^0, \cdots, v_{N-1}^0) + \sum_{k=1}^{B-1} \Phi\left(v_0^k, v_1^k, \cdots, v_{N-1}^k\right) 2^{-k} \quad (11)$$

is accomplished. The definition of the function that, when called, gives back the partial-product that corresponds to the input is.

$$\Phi(v_0^k, v_1^k, \cdots, v_{N-1}^k) \equiv \sum_{i=0}^{N-1} a_i v_i^k \quad (12)$$

In equation 11, the inner product of y may be determined by doing a weighted summation of the partial products. This can also be done in practise. On the right-hand side of the equation, the primary period has a weight of -1, which represents the sign bit. This weight is applied to the primary period. The significance of the weight of the 2-k symbol is ascribed to each of the phrases that follow it. The essential structure of the FIR filter, which makes use of the DA, is shown in figure 2, which offers a visual depiction of the fundamental structure. (DA-FIR). The operation of right-shifting is performed with the assistance of an adder, and the operation of addition is implemented with the assistance of a register. These partial-products were saved in the ROM. According to the above considerations, the amount of time that is necessary for the operation is essentially solely dependent on the word length B, and not on the total amount of terms N. This is because the word length B is the only variable that changes throughout the process. This suggests that the length of the B word is the only element that affects the output delay. [Cause and effect] It is possible to reduce the quantity of hardware by implementing the FIR filter using the DA rather than requiring multipliers. This would allow for the use of fewer multipliers. This indicates that it is possible to construct the filter.

HSCG-SCS Adder

An adder will produce the sum bit that corresponds to a given input bit position based on its knowledge of any carry inputs that may be present. This awareness is based on the fact that an adder is aware of any carry inputs that may be present. The carry might be generated and/or distributed internally amongst the multiple bit positions that comprise the input. This could happen either way. When an adder is used, a production of the sum bit that corresponds to a particular input bit position takes place. This happens when the adder is put into operation. To put it another way, the generation of the sum bit is contingent on having prior knowledge of any carry input. This is because the sum bit is used to calculate the total. This is due to the fact that the sum bit is used in the process of calculating the total amount. Because of this, it is feasible to transform the linear time that is experienced during an RCA into the necessary logarithmic time addition. Both of these operations are performed on the bits XQ and YQ. Both of these operations are performed on the bits XQ and YQ. Both of these operations are performed on the bits XQ and YQ. Equation may be used to explain any of these two processes (1). In line with the equation, the generation of the amount bit that corresponds to the Qth bit spot takes place (14).

$$CQ + 1 = GQ + PQCQ \quad (13)$$
$$SumQ = PQ \oplus CQ \quad (14)$$

Because equation (15) is naturally recursive, this property may be used to produce carries that correspond to following bit positions in advance. These carries are referred to as look-ahead carries since they involve looking forward. Equations (15)– (17). The bit locations in question are places 1, 2, 3, and 4. Utilizing these equations will allow one to get an understanding of the look-ahead carry. In the equations (15)– (17), CK represents the carry input to a 4-bit HSCG-SCS adder; PK+3 to PK represents the propagate signals; GK+3 to GK represents the generate signals; and CK+4 to CK+1 represents the look-ahead carry outputs received. Figure 3 is an illustration of the conventional HSCG-SCS adder that was created by putting the Equations (15)– (17) into practise.

$$CK + 1 = GK + PKCK \quad (15)$$
$$C_{K+2} = G_{K+1} + P_{K+1}C_{K+1} = G_{K+1} + P_{K+1}G_{K+1}P_{K+1}PKCK \quad (16)$$
$$C_{K+3} = G_{K+2} + P_{K+2}C_{K+2} = G_{K+2} + P_{K+2}G_{K+1} + P_{K+2}P_{K+1}G_{K+1}P_{K+2}P_{K+1}P_KC_K \quad (17)$$

Given that this is the most common method, it is common practise to construct an N-bit HSCG-SCS adder by first producing a cascade of M-bit HSCG-SCS adders. This is done in order to get the desired result. It is necessary that

both N and M have an even number of occurrences, and the value of N must be 0 when modulo M. This requirement can only be satisfied if both N and M are even. For instance, it is feasible to construct a HSCG-SCS adder with 32 bits by cascading eight HSCG-SCS adders, each of which only has four bits. Using the cascade operator is one way to accomplish this goal. In line with the structure of an N-bit HSCG-SCS adder, which is shown in the block diagram of an N-bit HSCG-SCS adder displayed in Figure 4, an N-bit HSCG-SCS adder may be formed from a number of 4-bit HSCG-SCS adders. This is depicted in the structure of an N-bit HSCG-SCS adder. The essential path, which can be recognised by the orange dotted line, is the one that absolutely has to be taken. It is the path that would be travelled by the essential path. The HSCG-SCS adder would be traversed along this route while it was being investigated. If the crucial route that the 4-bit HSCG-SCS adder was going to take was like the one depicted in Figure 3, then the following would apply, then this would be the case. The number of AND gates and OR gates that can be included into a conventional digital cell library at the present time is limited to a maximum of four in each case. Deconstructing an AND/OR gate that has five inputs into a combination of two AND/OR gates that each have three inputs is thus doable. If the 4-bit HSCG-SCS adder that is shown in Figure 3 is also present in the last step of Figure 4, at that time the dashed blue line that is shown in Figure 3 draws attention to the significant route that would be taken. This is because it highlights the fact that the route would be travelled. This would be the case if both figures are accurate.

It is clear from looking at Figure 4 that the first 4-bit HSCG-SCS adder, which is the one with the least significance, does not have a carry input, but the carry input is present in all of the other 4-bit HSCG-SCS adders. Figure 3 shows a representation of what the gate-level realisation of a 4-bit HSCG-SCS adder that does not contain any carry participation would look like. The attention-getting pink dashed line that emphasises the essential path brings focus to this particular aspect of the design. This is because a 4-bit HSCG-SCS adder's gate-level realisation without a carry input would be as explained above in the absence of a carry input. This is because in the lack of a carry input, that is, when assuming CK to be equal to zero in the equations, this result occurs.

RESULTS AND DISCUSSIONS

This section gives the detailed analysis of various simulation results, which are implemented using XILINX-ISE software and Verilog programming. Figure 4 shows the simulation waveforms, where proposed LMS adaptive filter functionality is justified. The design description (RTL schematic) of the suggested technique may be seen in Figure 5. The timing breakdown of the suggested technique is shown in Figure 6. In this situation, the suggested technique incurred a total time delay of 6.7690ns, of which 5.759ns was considered to be logical delay and 1.01ns was considered to be route delay. The report of the suggested proposed LMS adaptive filter's power consumption may be seen in Figure 7. In this case, the suggested propsoed LMS adaptive filter had a power consumption of 0.065. Figure 8 shows the design summary of proposed method. Table 1 compares the performance of various approaches. Here, the proposedLMS adaptive filter, resulted in reduced LUTs, power consumption, Flip-Flops, delay, as compared to conventional CONV-FOAF filter [12], Fixed FOAF filter [17], DFG FOAF filter [13].

CONCLUSION

In order to alleviate the problems caused by the stated limitations, a parallel multi-stage least mean square structure has been developed. The LMS consists of two stages that work in parallel and are coupled to each other through error feedback. The sum carry generator adder is based on half sum carry generation. In order to fulfil this goal, the array multiplier was switched out for distributed athematic memory units. In order for this design to be able to properly execute this duty, the DA-based HSCG-SCG adders are used. As a direct consequence of this, the memory capacity has been reduced by a factor of at least four, despite the fact that the filter order has remained same. The findings of the simulation indicate that the implementation of the suggested technique led to a reduction in area, latency, and power consumption when compared to the conventional adaptive filter. Further, this work can be extended with hybrid architectures for improved performance.

REFERENCES

1. Liu, G.; Zheng, L.; Wang, G.; Shen, Y.; Liang, Y. A carry lookahead adder based on hybrid CMOS-memristor logic circuit. IEEE Access 2019, 7, 43691–43696.
2. Balasubramanian, P.; Mastorakis, N. Performance comparison of carry-lookahead and carry-select adders based on accurate and approximate additions. Electronics 2018, 7, 369.
3. Gogineni, Vinay Chakravarthi, *et al.* "Algorithm and Architecture Design of Random Fourier Features-Based Kernel Adaptive Filters." IEEE Transactions on Circuits and Systems I: Regular Papers (2022).
4. Praveen Sundar, P. V., *et al.* "Low power area efficient adaptive FIR filter for hearing aids using DA architecture." International Journal of Speech Technology 23.2 (2020): 287-296.
5. Khan, MohdTasleem, *et al.* "Two DA Based High Throughput Architectures of Non-Pipelined LMS Adaptive Filters." IEEE Access 10 (2022): 76693-76706.
6. Khan, MohdTasleem, *et al.* "Two DA Based High Throughput Architectures of Non-Pipelined LMS Adaptive Filters." IEEE Access 10 (2022): 76693-76706.
7. Jiang, Honglan, *et al.* "A high-performance and energy-efficient FIR adaptive filter using approximate DA circuits." IEEE Transactions on Circuits and Systems I: Regular Papers 66.1 (2018): 313-326.
8. Shrivastava, Prabhat Chandra, *et al.* "Efficient Architecture for the Realization of 2-D Adaptive FIR Filter Using DA." Circuits, Systems, and Signal Processing 40.3 (2021): 1458-1478.
9. Kalaiyarasi, D., and T. Kalpalatha Reddy. "Design and implementation of least mean square adaptive FIR filter using offset binary coding based DA." Microprocessors and microsystems 71 (2019): 102884.
10. M. T. Khan and R. A. Shaik, "Optimal Complexity Architectures for Pipelined DA-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.
11. 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.
12. Di Meo, Gennaro, *et al.* "A Novel Low-Power High-Precision Implementation for Sign–Magnitude DLMS Adaptive Filters." Electronics 11.7 (2022): 1007.
13. Mula, Subrahmanyam, Vinay ChakravarthiGogineni, and AnindyaSundarDhar. "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.
14. Ahmad, Shawez, *et al.* "A novel multiplier-less LMS adaptive filter design based on offset binary coded DA." IEEE Access 9 (2021): 78138-78152.
15. Esposito, Darjn, *et al.* "Low-power hardware implementation of LMS adaptive filters using approximate arithmetic." Circuits, Systems, and Signal Processing 38.12 (2019): 5606-5622.
16. Shen, Binbin, Xiafu Lv, and Shuang Zhang. "An improved LMS adaptive filtering algorithm and its analysis." 2019 International Conference on Intelligent Computing, Automation and Systems (ICICAS). IEEE, 2019.
17. F. Lavalle-Aviles and E. Sánchez-Sinencio, "A 0.6-V Power-Efficient Active-RC Analog Low-Pass Filter With Cutoff Frequency Selection," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 28, no. 8, pp. 1757-1769, Aug. 2020, doi: 10.1109/TVLSI.2020.2999414.
18. Vinitha, C. S., and R. K. Sharma. "Area and energy-efficient approximate distributive arithmetic architecture for LMS adaptive FIR filter." 2020 International Conference for Emerging Technology (INCET). IEEE, 2020.
19. Prasad, Rajendra, E. R. Varun, and Roshan Zameer Ahmed. "Time-Shared LUT-Less Pipelined LMS Adaptive Filter." 2022 IEEE 2nd Mysore Sub Section International Conference (MysuruCon). IEEE, 2022.
20. Student, P. "FPGA implementation of efficient VLSI architecture of DLMS adaptive filter algorithm." Turkish J. Comput. Math. Educ. 12.14 (2021): 478-489.

Table 1. Comparison Table.

Parameter	CONVFOAF [12]	Fixed FOAF [17]	DFG FOAF [3]	Proposed LMS filter
Time delay (ns)	20.118	13.35	10.34	1.486
Power utilised (uw)	1.293	2.356	1.46	0.065
Slice Lookup tables	271	562	72	66
Slice registers	237	395	103	12

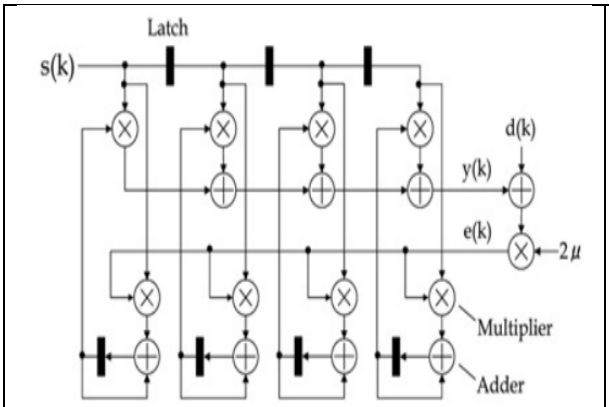

Fig. 1.The 4-tap LMS adaptive filter's composition.

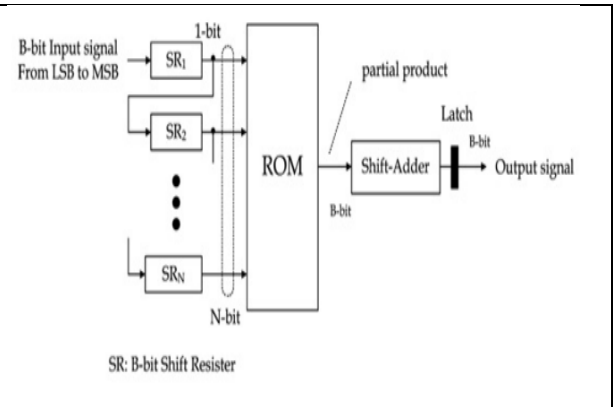

Fig. 2.DA-based LMS Adaptive Filterarchitecture.

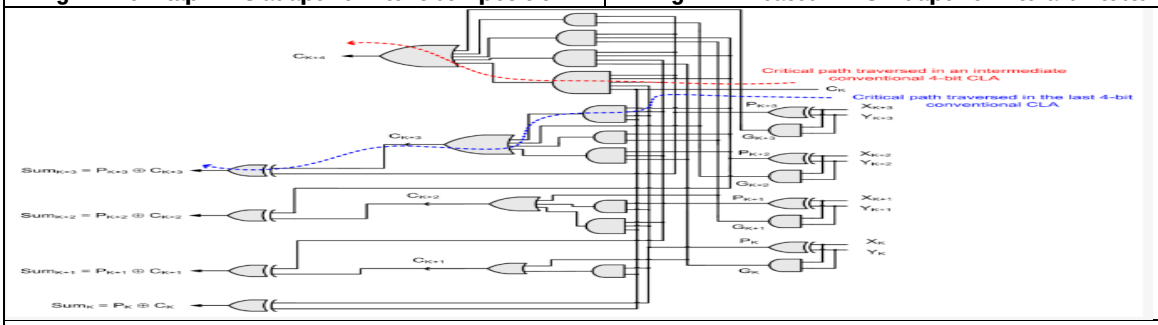

Fig. 3. Comprehension of 4-bit HSCG-SCS adder at the gate level with the carry inputC_K.

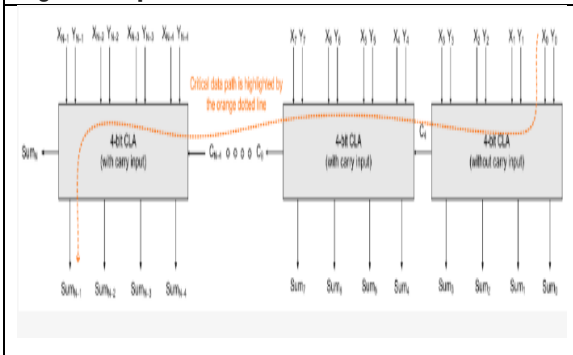

Fig. 4. Block illustration of an even N-bit HSCG-SCS adder implemented with a series of 4-bit HSCG-SCS adders.

Fig.5. Simulation Results.

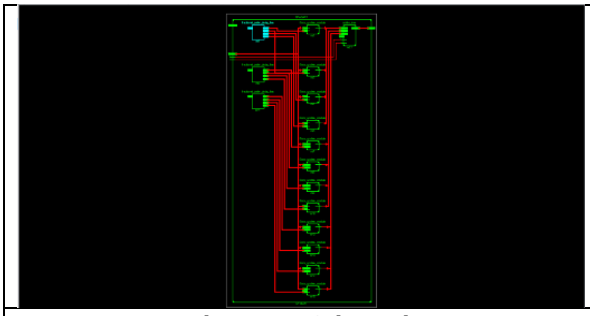

Fig. 6. RTL Schematic

```
Minimum period: 1.563ns (Maximum Frequency: 639.795MHz)
Minimum input arrival time before clock: 3.427ns
Maximum output required time after clock: 1.704ns
Maximum combinational path delay: 1.486ns
```

Fig. 7. Time summary.

Device	
Family	Zynq-7000
Part	xc7z010
Package	clg400
Temp Grade	Commercial
Process	Typical
Speed Grade	-2

Environment	
Ambient Temp (C)	25.0
Use custom TJA?	No
Custom TJA (C/W)	NA
Airflow (LFM)	250
Heat Sink	Medium Profile
Custom TSA (C/W)	NA
Board Selection	Medium (10"x10")
# of Board Layers	8 to 11
Custom TJB (C/W)	NA

On-Chip	Power (W)	Used	Available	Utilization (%)
Logic	0.000	34	17600	0
Signals	0.000	75	--	--
IOs	0.000	50	230	22
Leakage	0.065			
Total	0.065			

Thermal Properties	Effective TJA (C/W)	Max Ambient (C)	Junction Temp (C)
	4.0	84.7	25.3

Supply Summary		Total	Dynamic	Quiescent
Source	Voltage	Current (A)	Current (A)	Current (A)
Vccint	1.000	0.005	0.000	0.005
Vccaux	1.800	0.006	0.000	0.006
Vcco18	1.800	0.001	0.000	0.001
Vccbram	1.000	0.000	0.000	0.000
Vccpint	1.000	0.020	0.000	0.020
Vccpaux	1.800	0.013	0.000	0.013
Vcco_ddr	1.500	0.002	0.000	0.002

Supply Power (W)	Total	Dynamic	Quiescent
	0.065	0.000	0.065

Fig.8. Power summary

Device Utilization Summary (estimated values)

Logic Utilization	Used	Available	Utilization
Number of Slice Registers	12	126800	0%
Number of Slice LUTs	66	63400	0%
Number of fully used LUT-FF pairs	9	69	13%
Number of bonded IOBs	25	210	11%
Number of BUFG/BUFGCTRLs/BUFHCEs	1	128	0%

Fig.9. Device Utilization Summary.