ISSN (Online) 2321 – 2004
ISSN (Print) 2321 – 5526

IJIREEICE

International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering

nCORETech

LBS College of Engineering, Kasaragod

Vol. 3, Special Issue 1, February 2016

An Efficient DLMS Adaptive Filter Architecture

Swathi K V[1], Pramod P[2]

PG Scholar, Electronics and Communication Engineering, LBS College of Engineering, Kasaragod, India[1]

Assistant Professor, Electronics and Communication Engineering, LBS College of Engineering, Kasaragod, India[2]

Abstract: An efficient adaptive filter architecture using booth multiplier has been developed. For deriving low-complexity and high-speed implementation a precise analysis of the critical path of the least-mean-square (LMS) adaptive filter has been done in this paper. For achieving area-delay-power efficient implementation and lower adaptation-delay, we use efficient partial product generator and propose a method for optimization of the time-consuming combinational blocks of the structure by using pipeline stages. We proposed an area-delay-power efficient lower adaptation delay architecture for implementation of LMS adaptive filter. We used partial product generator for efficient implementation of bitwise multiplications and inner-product computation by sharing the common sub expressions. In multiplier major part of the delay is contributed by partial product addition. In order to reduce the delay Carry look ahead adder is used as the adder to perform partial product addition in the adder tree. The propagation delay occurred in the parallel adders can be eliminated by carry look ahead adder. The proposed structures involves significantly less adaptation delay and provide significant saving of ADP compared to the existing structures. We use Xilinx 14.7 to provide VHDL coding for our architecture. Result shows that proposed structure has better performance than existing algorithms.

Keywords: Adaptive filter, Least Mean Square, Carry Look Ahead Adder, Booth Multiplier.

I. INTRODUCTION

The LEAST MEAN SQUARE (LMS) adaptive filter is most popular and widely used adaptive filter, mainly because of its satisfactory convergence performance and simplicity. Adaptive filters are widely used in signal processing applications. Widrow–Hoff least mean square (LMS) algorithm is widely used adaptive filter for updating filter weights. The direct form configuration of the FIR filter on the forward path results in a long critical path due to the computation of filter output which involves inner product computation. When the structure exceeds the desired sample period the critical path should be reduced by using pipelined implementation. Since the conventional LMS algorithm does not support pipelined implementation mainly because of its recursive behavior, it is modified to form the delayed LMS (DLMS) algorithm, which allows pipelined implementation of the filter. Adaptive algorithms are required in order to continuously update the filter coefficients. In adaptive filters the filter coefficients or the weights are updated to minimize the error between filter output and desired output. Thus a good algorithm is required to update the filter weights accordingly to attain fast convergence. The real challenge in designing an adaptive filter resides with the adaptive algorithm. The required algorithm must be practical to implement, adapt the coefficients quickly and provide the desired response. A lot of work has been done to obtain area, delay and power efficient implementation of LMS adaptive filters. In most of the works pipelining across the computational blocks has been used. Due to an inner-product computation to obtain the filter output the direct-form LMS adaptive filter involves a long critical path. When the structure exceeds the desired sample period the critical path is required to be reduced by pipelined. For most practical cases no pipelining is required to implement a direct-form LMS adaptive filter, and can be realized with a very small adaptation delay in cases where a very high sampling rate is required [1].As LMS algorithm does not support pipelining due to its recursive behaviour the algorithm has been modified to a form called Delayed LMS (DLMS) algorithm [2], [3]. Surya Prakash and Rafi [3] have proposed a high-performance implementation scheme for a least mean square adaptive filter based on distributed arithmetic. Meher and Park have proposed a 2-bit multiplication cell, and it is used with an efficient adder tree for pipelined inner-product computation to minimize the critical path and silicon area without increasing the number of adaptation delays [4], [5]. Further effort has been made by Meher and Maheshwari [6] to reduce the number of adaptation delays.Ting et al. [7] have proposed a fine-grained pipelined design which limits the critical path to the maximum of one addition time, it supports high sampling frequency, but involves a lot of area overhead for pipelining and higher power consumption than in [8], due to its large number of pipeline latches. Adaptation delay is an important parameter of an adaptive filter that has to be reduced to a desired value for fast convergence.Van and Feng [8] have proposed a systolic architecture, where they use relatively large number of processing elements (PEs) for achieving lower adaptation delay with the critical path of one MAC operation.The work done to implement the DLMS algorithm in systolic architecture to increase the maximum usable frequency [9], [10] involve an adaptation delay of N cycles for the filter length N, which is quite high for large order filters. This paper

IJIREEICE
ISSN (Online) 2321 – 2004
ISSN (Print) 2321 – 5526
International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering
nCORETech
LBS College of Engineering, Kasaragod
Vol. 3, Special Issue 1, February 2016

mainly focuses on less power consumption and lower area complexity without compromising the desired processing throughput.

II. DLMS ALGORITHM

LMS algorithm does not support pipelining due to its recursive nature. So it is modified to delayed LMS (DLMS) algorithm. The weights of LMS adaptive filter during nth iteration are updated based on the following equation:

$$w_{n+1} = w_n + \mu e_n \quad (1)$$

$$e_n = d_n - y_n \; ; \; y_n = w_n^T x_n \quad (2)$$

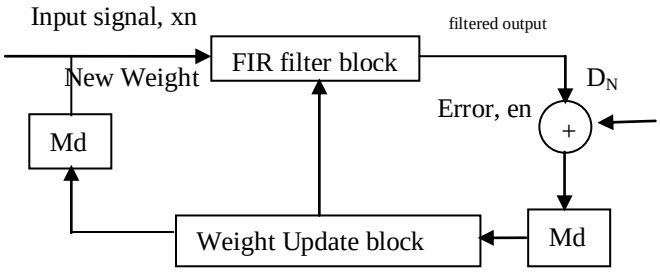

Fig.1 Structure of conventional LMS adaptive filter

In the case of pipelined designs with m stages of pipeline, the error e_n becomes available after m cycles, where m is the adaptation delay. The DLMS algorithm thus uses the delayed errore_{n-m}, i.e., the error corresponding to (n-m)th iteration for updating the current weight instead of the recent-most error. Thus we have

$$w_{n+1} = w_n + \mu e_{n-m} x_{n-m} \quad (3)$$

It was shown that the adaptation delay of conventional LMS is divided into two parts: one part is the delay introduced by the pipeline stages in FIR filtering, and the other one is due to the delay involved in pipelining the weight-update process. Depending on such a decomposition of delay, the DLMS adaptive filter can be implemented by a structure shown in Figure 2. Assuming that the latency of computation of error is n_1 cycles, the error computed by the structure at the nth cycle is e_{n-n_1}, which is used along with the input samples delayed by n_1 cycles to generate the weight-increment term. The weight-update equation of the DLMS algorithm is given by

$$w_{n+1} = w_n + \mu e_{n-n_1} x_{n-n_1} \quad (4)$$

$$e_{n-n_1} = d_{n-n_1} - y_{n-n_1} \quad (5)$$

$$y_n = w_{n-n_2}^T x_n \quad (6)$$

Since all weights are updated concurrently in every cycle to compute the output, direct-form realization of the FIR filter is commonly used for implementation. However, the direct-form LMS adaptive filter is believed to have longer critical path due to computation of inner product to obtain the filter output. This is mainly based on the assumption that only after the complete input operand words are available/ generated an arithmetic operation starts.

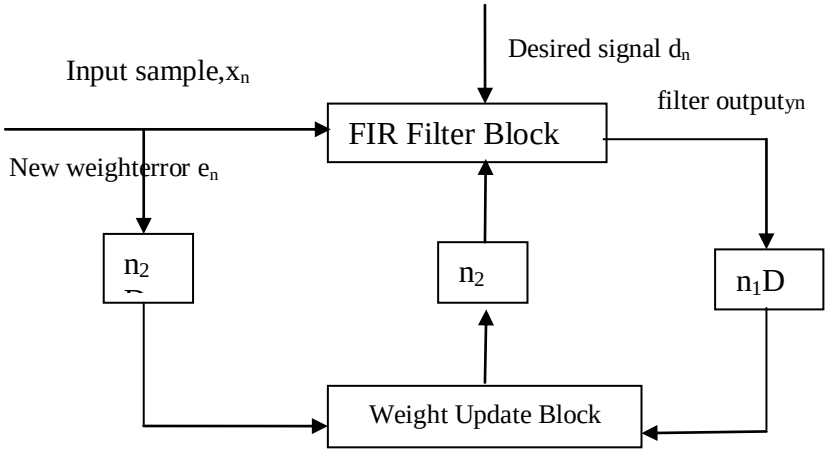

Fig. 2 structure of D LMS adaptive filter

IJIREEICE
ISSN (Online) 2321 – 2004
ISSN (Print) 2321 – 5526
International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering
nCORETech
LBS College of Engineering, Kasaragod
Vol. 3, Special Issue 1, February 2016

III.PROPOSED STRUCTURE

In this section, we discuss area, delay, and power efficient implementation of adaptive filter using modified booth multiplier. Here zero adaptation delay filter is implemented.

A. Zero Adaptation Delay Filter

There are two main computing blocks in direct form LMS adaptive filter as shown in figure 3 ,i) error computation block ii) weight update block. When we realise both error computing block and weight update block separately we find many area intensive components are common in the both blocks: the multipliers, weight registers, tapped delay line.

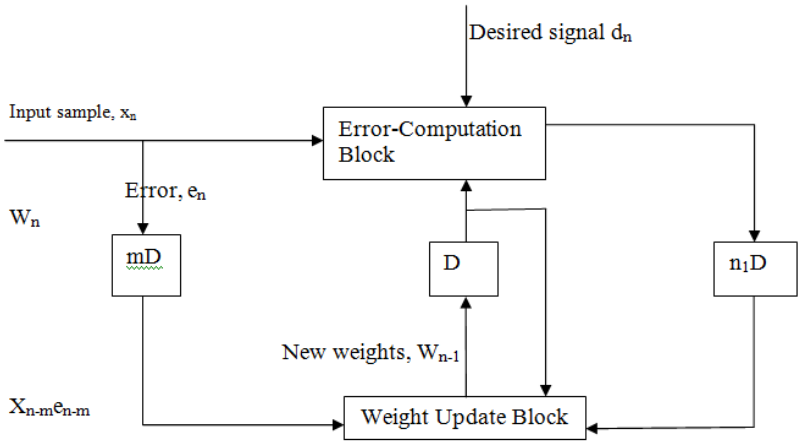

Fig. 3 Generalised block diagram of DLMS adaptive filter

The adder and subtractor which constitute only small part of the circuit are different in these two computing blocks. For implementing zero adaptaion delay, the computation of both these blocks is required to be performed in the same cycle. Moreover since the structure is non-pipelined type, weight updating and error computation cannot occur concurrently. Therefore, the multiplications of both these phases could be multiplexed by the same set of multipliers. Here for decreasing delay and power we use a modified booth multiplier. The same registers are used for both these phases if error computation is performed in the first half cycle, while weight update is performed in the second half cycle. Here time-multiplexed-zero adaptation delay structure is proposed for a direct form N-tap LMS adaptive filter is shown in figure 4.

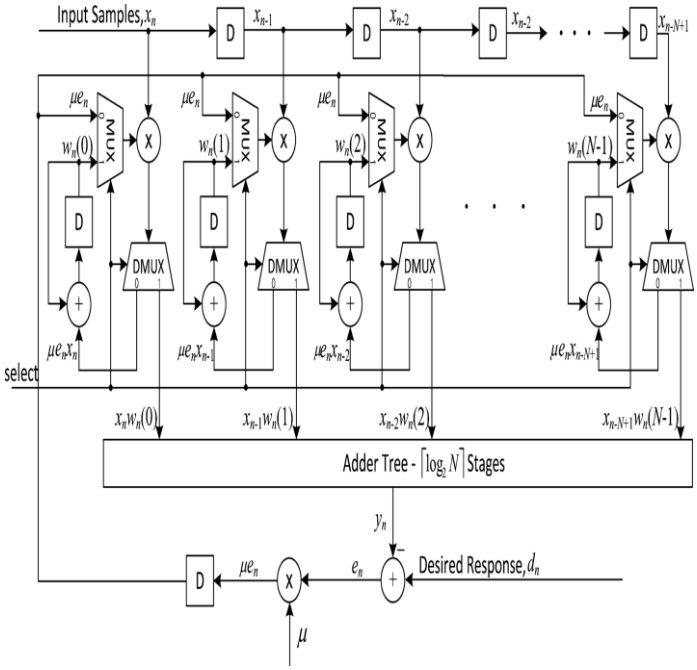

Fig. 4 Proposed structure for zero adaptation delay LMS adaptive filter

IJIREEICE
ISSN (Online) 2321 – 2004
ISSN (Print) 2321 – 5526
International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering
nCORETech
LBS College of Engineering, Kasaragod
Vol. 3, Special Issue 1, February 2016

In this structure the input samples are fed to multipliers from common tapped delay line. The N weight values are stored in N registers and the estimated error value obtained are fed to multipliers as the other input through a 2:1 multiplexer. The proposed structure requires N adders and an adder tree to add the output of N multipliers to compute output of the filter. To compute error value it uses a subtractor. N 2:1 de-multiplexers is used to move the product values either towards the adder tree or weight update circuit. All these multiplexers and de-multiplexers are controlled by clock signal.The registers in the delay line are clocked at the rising edge of the clock pulse and it remains unchanged for a complete clock period since the structure is required to take one new sample in every clock cycle. During the first half of each clock period, the weight values stored in different registers are fed to the multiplier through the multiplexers to compute the filter output. The products are then fed to the adder tree though the de-multiplexers. The filter output is computed by the adder tree and the error value is computed by a subtractor. Then the computed error value is right-shifted to obtain μe_n and is broadcasted to all N multipliers in the weight-update circuits. Note that the LMS adaptive filter requires at least one delay at a suitable location to break the recursive loop. A delay could be inserted either after the adder tree, after the e_n computation, or after the μe_n computation. If the delay is placed just after the adder tree, then the critical path shifts to the weight-updating circuit and gets increased by T_{ADD}. Therefore, we should place the delay after computation of e_n or μe_n, but preferably after μe_n computation to reduce the register width.If we get a new set of weight values computation of second half cycle is completed. The updated weight values are used in first half cycle of next clock for computation of filter output and for subsequent error estimation. The weight registers are updated by new weight values when next cycle begins. Therefore weight registers are also clocked at the rising edge of each clock pulse. We cannot exactly determine when the error computation is over and when weight updating is completed. The time required for error computing is more than that of weight updating.

B. Adders and Multipliers

The main important component in the proposed structure is adders and multiplier unit. While keeping in mind optimization we mainly focus on adder and multiplier blocks. Basically implementation of multiplier and adder requires large chip area, increased latency and consumes considerably large amount of power. Therefore design of low-power multiplier and low-power adder is an important concept in low- power VLSI system. The fast multipliers and adders are essential in digital signal processing (DSP) systems. The speed of multiplication has great significance in digital signal processing and other processors.

For arithmetic multiplication, we have variety of multiplication architectures like array multiplier, Booth multiplier and Wallace tree multiplier have been analyzed. Then it has been found that Booth multiplier is most efficient among all the multipliers, giving optimum delay, area and power for multiplication. Low power modified Booth decoder and pipelining techniques have been used to reduce power and delay.

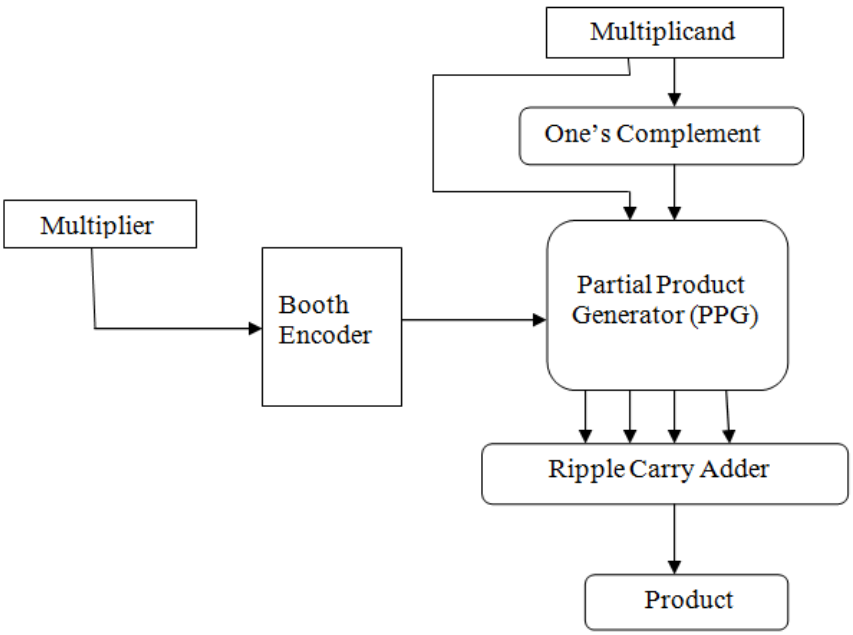

Fig. 5 Proposed structure for modified Booth Multiplier

IJIREEICE
ISSN (Online) 2321 – 2004
ISSN (Print) 2321 – 5526
International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering
nCORETech
LBS College of Engineering, Kasaragod
Vol. 3, Special Issue 1, February 2016

In booth multiplier [11] by recording the multiplier bit into groups that select multiples of multiplicand the number of summands is reduced. From the basics of Booth Multiplication it can be proved that the addition/ subtraction operation can be skipped if the successive bits in the multiplicand are same. To achieve high performance, the modified. To reduce the number of partial products by a factor of two we use booth encoding which has been widely adopted all parallel multipliers. Addition is the most common operation in arithmetic applications. Adders are some of the most critical data path circuits requiring considerable design effort in order to gain as much performance as possible. Various adder structures can be used to perform addition such as parallel and serial structures. Adders like ripple carry adder, carry select adder, carry look ahead adder, carry skip adder, carry save adder etc exist numerous adder implementations each with good attributes and some drawbacks.

A fast method used for adding numbers is called carry-look ahead. This method does not require the carry signal to propagate through all stages, causing a bottleneck. Instead it uses additional logic to expedite the generation and propagation of carry information, allowing fast addition at the expense of more hardware requirements. A carry-look ahead adder increase speed by reducing the amount of time required to determine the carry bits. It can be contrasted with the simpler ripple carry adder, but usually slower, for which the carry bit is calculated alongside the sum bit, and until the previous carry has been calculated each bit must wait to begin calculating its own result and carry bits.

The carry-look ahead adder [12] calculates one or more than one carry bits before the sum, which reduces the wait time to calculate the output of the larger value bits. The CLA solves the problem of delay it takes to propagate the carry, by calculating the carry signal in advance based on the input signal. The working of this adder can be understood by determining Boolean expressions dealing with full adder. The propagate 'Pi' and generate 'Gi' in a full adder is given by

Carry propagate

$$P_i = x\_in \text{ xor } y\_in \qquad (7)$$

Carry generate

$$G_i = x\_in \text{ and } y\_in \qquad (8)$$

Both propagate and generate signals depend only on the input bits and thus will be valid after one gate delay. The new expressions for the output sum and the carryout are given by:

$$\text{Sum} = S_i = P_i \text{ xor } C_{i-1} \qquad (9)$$

$$\text{Carry\_out} = C_{i-1} = G_i + P_i \text{ and } C_i \qquad (10)$$

Carry look-ahead adder's structure can be divided into three parts: the propagate/generate generator, the sum generator, carry generator. The architecture of 4-bit Carry Look-Ahead adder is shown in Figure 6.

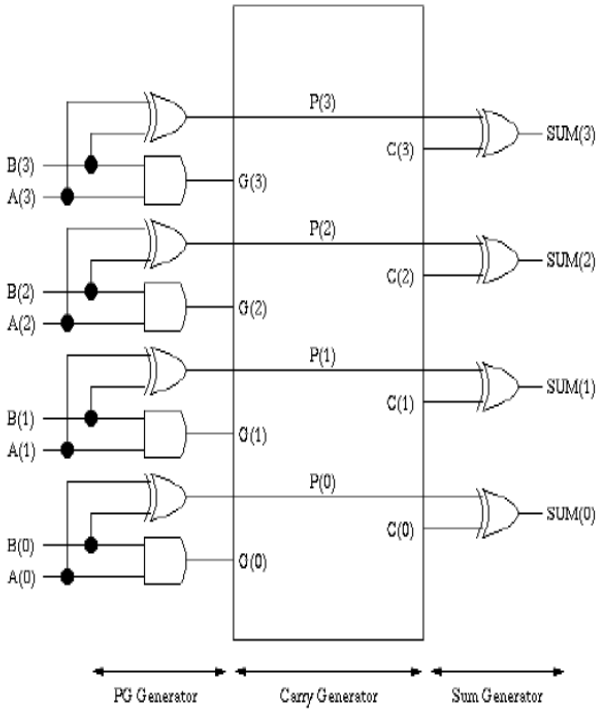

Fig. 6 4 bit Carry look ahead adder

IJIREEICE
ISSN (Online) 2321 – 2004
ISSN (Print) 2321 – 5526
International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering
nCORETech
LBS College of Engineering, Kasaragod
Vol. 3, Special Issue 1, February 2016

The major goal that requires ultimate attention is power consumption. Hence adders are designed to reduce the propagation delay which is also a cause for power consumption.

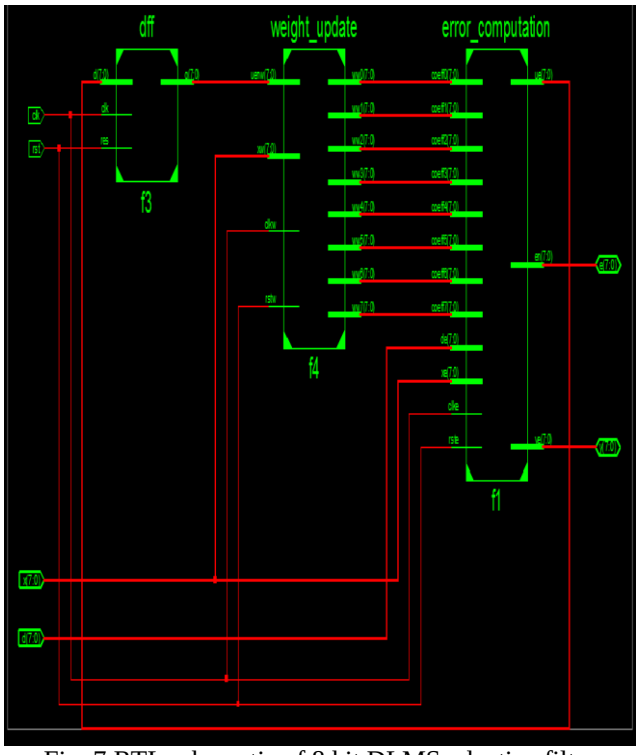

Fig. 7 RTL schematic of 8 bit DLMS adaptive filter

IV. SIMULATION RESULTS

The DLMS adaptive filter proposed structure is simulated using Xilinx 14.7. In order to reduce the adaptation delay in LMS algorithm, various architectures have been proposed. The changes have been made in the error computation block and weight update block. The major changes were made at multiplier and adder block. DLMS adaptive filter had a shorter critical path than the traditional system. Efficient addition scheme for inner product computation reduced adaption delay which improved convergence performance and reduced critical path to support high input sample rates. Optimized balanced pipelining across time consuming blocks reduced adaption delay and power consumption.

Fig. 8 Simulation result of 8 bit DLMS adaptive filter

IJIREEICE
ISSN (Online) 2321 – 2004
ISSN (Print) 2321 – 5526
International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering
nCORETech
LBS College of Engineering, Kasaragod
Vol. 3, Special Issue 1, February 2016

Comparison of the proposed method with the existing is shown in Table 1 as follows. Delay and power consumed has been reduced while comparing proposed with the existing method.

TABLE I Comparison of proposed method with existing

Design	Filter Length N	Delay (ns)	Power (mW)
Existing DLMS Design	8	7.54	0.79
Proposed Design	8	6.89	0.321

V. CONCLUSION

Based on a precise critical-path analysis, low-complexity architectures for the LMS adaptive filter is proposed. The direct-form and transpose-form LMS adaptive filters have nearly the same critical-path delay. The direct-form LMS adaptive filter involves less register complexity and provides much faster convergence than its transpose-form counterpart since the latter inherently performs delayed weight adaptation.
Besides, proposed an efficient addition scheme for computing inner-product to reduce the adaptation delay significantly in order to achieve faster convergence performance and to reduce the critical path to support high input sampling rates. Apart from this, a strategy for optimization pipelined stages across the time consuming blocks of the structure in order to reduce the adaptation delay and power consumption, as well. The proposed structure involve significantly less adaptation delay and provide significant saving of ADP when compared to the existing structures. The usage of booth multiplier and carry look ahead adder further reduced the delay and power. Hence an efficient architecture for DLMS adaptive filter in terms of area, delay, power is proposed in this paper.

ACKNOWLEDGMENT

The authors would like to express special thanks to teachers, parents and friends who provided insight and expertise that greatly assisted the research. The authors would also like to thank the principal and the institute for providing all the facilities for successfully completing the paper work. Above all sincere thanks to god who is the power of the strength in each step of progress towards its successful completion.

REFERENCES

[1] P. K. Meher and S. Y. Park, (2014) "Critical-Path Analysis and Low-Complexity Implementation of the LMS Adaptive Algorithm" IEEE transactions on circuits and systems, vol. 61, no. 3, pp. 778–788.
[2] P. K. Meher and S. Y. Park, (2014) "Area-delay-power efficient fixed-point LMS adaptive filter with low adaptation-delay" IEEE Transanctions on Very Large Scale Integration (VLSI) Signal Process, vol. 22, no. 2, pp. 362–371.
[3] M Surya Prakash and Rafi Ahamed Shaikh (2013) "Low-Area and High-Throughput Architecture for an Adaptive Filter Using Distributed Arithmetic" IEEE Transaction on circuits and systems, vol. 60, no. 11, pp. 781–785.
[4] P. K. Meher and S. Y. Park, (2011) "Low adaptation-delay LMS adaptive filter part-I: Introducing a novel multiplication cell" in Proc. IEEE Int. Midwest Symp. Circuits Systems, pp. 1–4.
[5] P. K. Meher and S. Y. Park, (2011) "Low adaptation-delay LMS adaptive filter part-II: An optimized architecture" in Proc. IEEE Int. Midwest Symp. Circuits Systems, pp. 1–4.
[6] P. K. Meher and M. Maheshwari, (2011) "A high-speed FIR adaptive filter architecture using a modified delayed LMS algorithm" in Proc. IEEE Int. Symp. Circuits Systems, pp. 121–124.
[7] L.-K. Ting, R. Woods, and C. F. N. Cowan, (2005) "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. 86–99.
[8] L. D. Van and W. S. Feng, (2001) "An efficient systolic architecture for the DLMS adaptive filter and its applications" IEEE Transactions on Circuits Systems II, Analog Digital Signal Processing, vol. 48, no. 4, pp. 359–366.
[9] M. D. Meyer and D. P. Agrawal, (1993) "A high sampling rate delayed LMS filter architecture" IEEE Transaction on Circuits Systems II, Analog Digital Signal Processing, vol. 40, no. 11, pp. 727–729.
[10] H. Herzberg and R. Haimi-Cohen, (1992) "A systolic array realization of an LMS adaptive filter and the effects of delayed adaptation" IEEE Transactions on Signal Processing, vol. 40, no. 11, pp. 2799–2803.
[11] M. J. Rao, S. Dubey, "A high speed and area efficient Booth recoded Wallace tree multiplier for Fast Arithmetic Circuits," in Asia Pacific Conference on Postgraduate Research in Microelectronics and Electronics (PrimeAsia), Hyderabad, India, 5-7 Dec. 2012, pp.220-223.
[12] Basant Kumar Mohanty, Sujit Kumar Patel, (2014) "Area–Delay–Power Efficient Carry-Select Adder," in IEEE Transaction on Circuits Systems II, Express Breifs, vol. 61, no. 6.