International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2013): 6.14 | Impact Factor (2013): 4.438

Area Efficient Fixed-Point LMS Adaptive Filter

Velloree Khuraijam

Dept. of Electronics and Communication, Karavali Institute of Technology, Mangalore-574143, Visvesvaraya Technological University, Karnataka, India

Abstract: ***The proposed design presents an efficient architecture for the implementation of a delayed least mean square adaptive filter. A novel partial product generator (PPG) is used for area-delay efficient implementation and to reduce the adaptation delay. This paper presents a strategy for optimized balanced pipelining across the time-consuming combinational blocks of the structure. From synthesis result, it has been found that the proposed design offers less area-delay product (ADP) and less energy delay product (EDP) than the best of the existing systolic structures, on average, for filter lengths N= 8, 16 and 32. An efficient fixed-state error is proposed. It has been shown that the steady state mean squared error obtained from the analytical result matches with the simulation result.***

Keywords: Partial product generator, Adaptive filters, least mean square (LMS) algorithms, Adder structure. Modelsim

1. Introduction

An adaptive filter is a computational device that iteratively models the relationship between the input and output signals of the filter. An adaptive filter self-adjusts the filter coefficient according to an adaptive algorithm to minimize the power of error signal iteratively. One of the most well known adaptive algorithms is Least Mean Square (LMS) algorithm. LMS algorithm requires fewer computational resources and memory. The Least Mean Square (LMS) adaptive filter is the most widely used adaptive filter not only because of its simplicity but also because of its satisfactory convergence performance. The direct-form LMS adaptive filter involves a long critical path due to an inner-product computation to obtain the filter output. The critical path is required to be reduced by pipelined implementation when it exceeds the desired sample period. Since the conventional LMS algorithm does not support pipelined implementation because of its recursive behavior, it is modified to a form called the delayed LMS (DLMS) algorithm which allows pipelined implementation of the filter.

The existing work on the DLMS adaptive filter does not discuss the fixed-point implementation issues i.e. location of radix point, choice of word length, and quantization at various stages of computation, although they directly affect the convergence performance, particularly due to the recursive behavior of the LMS algorithm. Therefore fixed-point implementation issues are given adequate emphasis in this paper. Besides, the paper presents here the optimization of design to reduce the number of pipeline delays along with the area, sampling period and energy consumption. The proposed design is found to be more efficient in terms of the power-delay product (PDP) and energy-delay product (EDP) comparing to the existing structures.

2. Block Diagram of Delayed LMS Adaptive Filter

The adaptation delay of m cycles amounts to the delay introduced by the whole of adaptive filter structure consisting of finite impulse response (FIR) filtering and the weight update process. The adaptation delay of conventional LMS can be decomposed into two parts: one part is the delay introduced by the pipeline stages in FIR filtering, and the other part is due to the delay involved in pipelining the weight update process. Based on such a decomposition of delay, the DLMS adaptive filter can be implemented by a structure shown in figure 1.

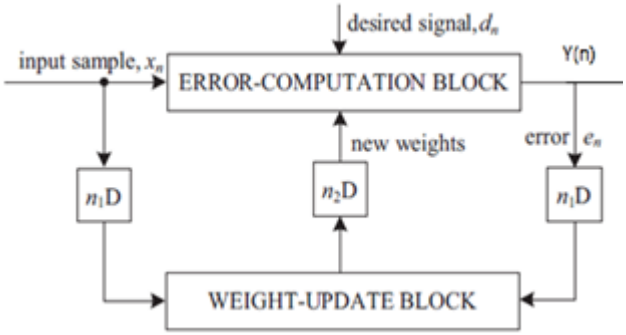

Figure 1: Structure of delayed LMS adaptive filter

3. Modules

- Error-computation block
- Proposed structure of PPG
- AND/OR cell
- Adder-structure
- Weight-update block

4. Module Description

1. Error Computation Block: An error-computation block is proposed to minimize the adaptation delay. The proposed structure of Error- Computation Unit of an N-tap DLMS adaptive filter is shown in fig. 2. It consists of N-2b partial product generator (PPG) corresponding to N multipliers and a cluster of L/2 binary adder trees, followed by a single shift-add tree.

1.1 Proposed Structure of PPG: The propose design presents PPG for efficient implementation of general multiplication and inner product computation by common sub expression sharing. The structure of each PPG is shown in fig. 3. It consists of L/2 number of 2-to-3 decoders and the same number of AND/OR cells (AOC). Each of the 2-to-3 decoders takes a 2-b digit (u1u0) as input and produces three outputs b0=1 for (u1u0) =1, b1=1 for (u1u0) =2, and b2=1 for (u1u0) =3. The decoder output b0, b and b2 are fed to an AOC. Along with this, w, 2w and 3w are fed to the AOC

where w, 2w and 3w are in 2's complement representation and sign-extended to have (W+2) bit each. The AOC (L/2−1) is fed with w, −2w and –w as input while computing the partial product corresponding to the most significant digit (MSD) i.e. ($u_{L-1}u_{L-2}$) of the input sample.

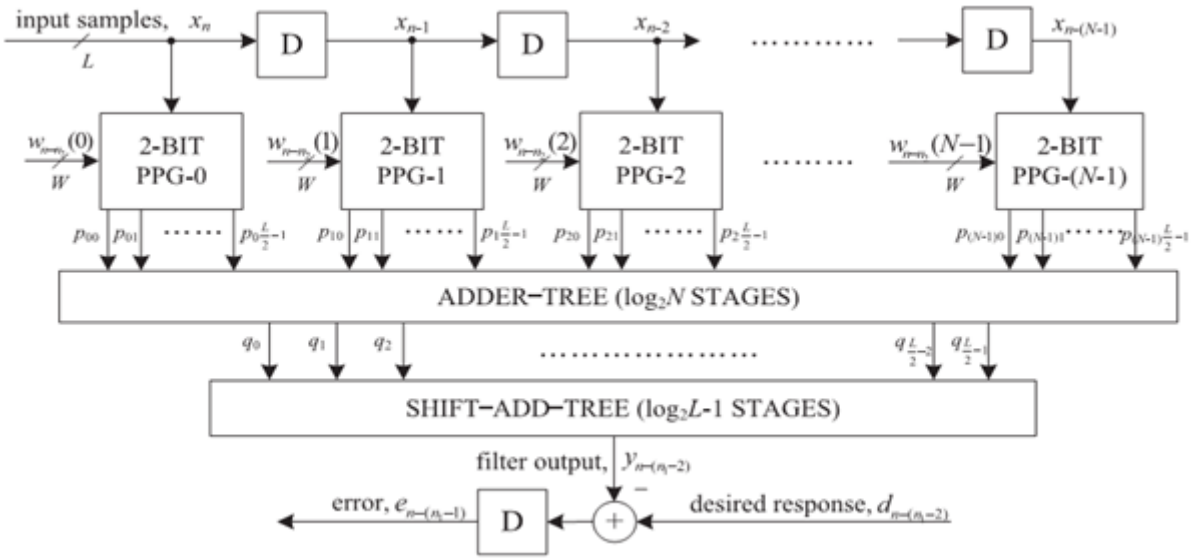

Figure 2: Proposed structure of Error Computation Block

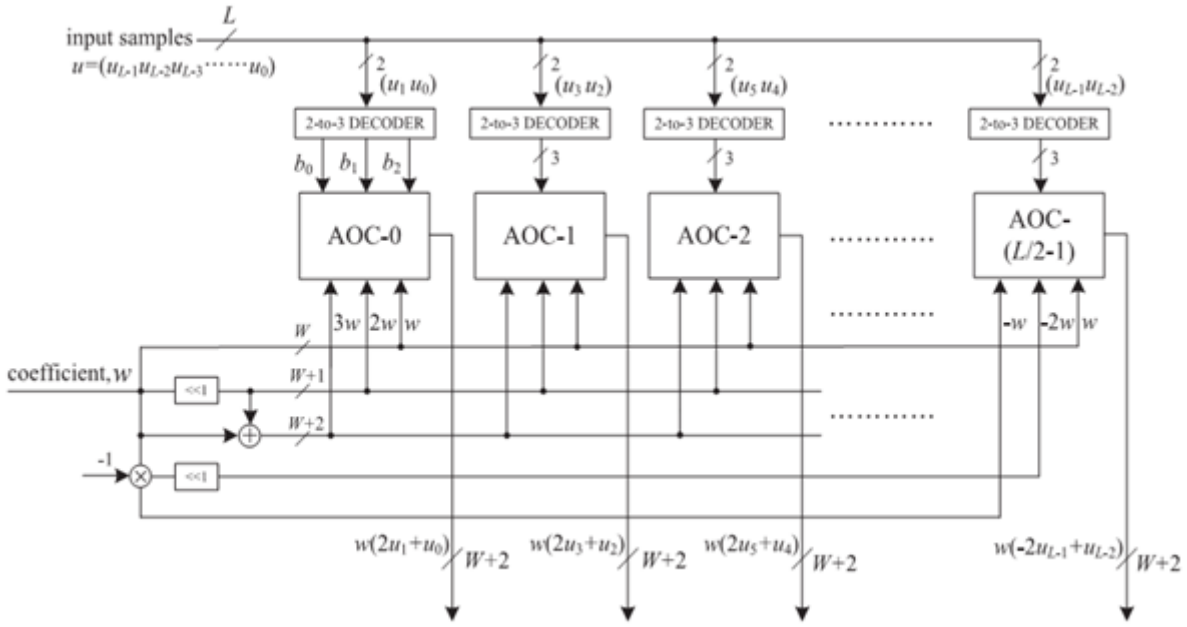

Figure 3: Proposed structure of PPG, AOC stands for AND/OR cell

1.2. AND/OR Cell: The proposed PPG have L/2 number of AOC. The structure and function of an AOC are depicted in fig. 4. Each AOC consists of three AND cells and two OR cells. The structure and function of AND cells and OR cells are depicted by fig. 4(a) and 4(b) respectively. Each AND cell consists of n AND gates and it takes two inputs i.e. an n-bits of input of D and a single bit input b. the n bits of input D is distributed to its n AND gates, this being one of the input to n AND gates and the other input of all the n AND gates are fed with the single-bit input b. As shown in fig.4(c), each OR cell similarly takes a pair of n-bit input words and has n OR gates. A pair of bits in the same bit position in B and D is fed to the same OR gate. The output of an AOC is w, 2w and 3w corresponding to the decimal values 1, 2 and 3 of the 2-bit input (u1u0), respectively. The decoder along with the AOC performs a multiplication of input operand w and a 2-b digit (u1u0), such that the PPG of fig.3 performs L/2 parallel multiplications of input w with a 2-b digit to produce L/2 partial products of the product word wu.

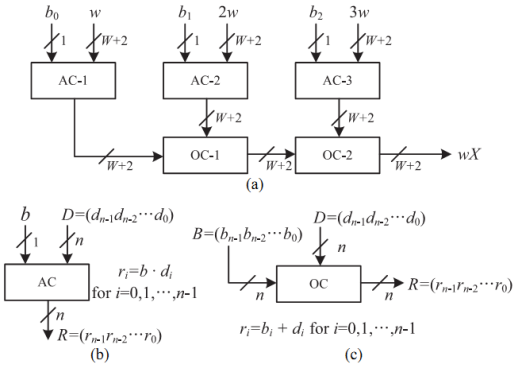

Figure 4: Structure and function of AND/OR cell. Binary operations · and + in (b) and (c) are implemented using AND and OR gates, respectively

1.3. Adder Structure: Conventionally, we should have performed the shift-add operation on the partial products of each PPG separately to obtain the product value and then added all the N products values to compute the desired inner product. However, the shift-add operation to obtain the product value increases the word length, and consequently increases the adder size of N−1 additions of the product values. To avoid such increase in word size of the adders, we add all the N partial products of the same place value from all the N PPGs by one adder tree. The adder trees are shown in bellow Figure 6.

Table 1: Location of pipeline latches for L=8 and N=8, 16 and 32

N	Error Computation-Block		Weight Update Block
	Adder Tree	Shift-add Tree	Shift-add Tree
8	Stage-2	Stage-1 and 2	Stage-1
16	Stage-3	Stage-1 and 2	Stage-1
32	Stage-3	Stage-1 and 2	Stage-2

2. Weight-Update Block: The proposed structure for the weight-update block is shown in fig. 7. It performs N multiply-accumulate operations of the form $(\mu \times e) \times x_i + w_i$ to update N filter weight. The step size μ is taken as a negative power of 2 to realize the multiplication with recently available error of 2 to realize the multiplication with recently available error only by a shift operation. Each of the MAC units therefore performs the multiplication of the shifted value of error with the delayed input samples x_i followed by the additions with the corresponding old weight values w_i. All the N multiplications for the MAC operations are performed by N PPGs, followed by N-shift-add trees. Each of the PPGs generates L/2 partial products corresponding to the product of the recently shifted error values $\mu \times e$ with L/2, the number of 2-b digits of the input word x_i, where the sub expression $3\mu \times e$ is shared within the multiplier. Since the scaled error $(\mu \times e)$ is multiplied with the entire N delayed input values in the weight-update block, this sub expression can be shared across all the multipliers as well. This leads to substantial reduction of the adder complexity. The final output of MAC units constitute the desired updated weights to be used as inputs to the error-computation block as well as the weight-update block for the next iteration.

5. Fixed-Point Consideration

The fixed point implementation of the proposed DLMS adaptive filter shows the bit level pruning of the adder tree, to reduce the hardware complexity without the degradation steady state Mean-Squared-Error (MSE). For fixed-point implementation, the word lengths and radix points for input samples, weights and internal signals are needed to be decided. Fig.5. shows the fixed-point representation of binary number. Table II shows the fixed representation of the desired signals; its quantization is usually given as an input. For this purpose, the specific scaling/sign extension and truncation/zero padding are required. Since the LMS algorithm performs learning so that y has a same sign as d, the error signal e can also be set to have the same representation as y without overflow after the subtraction.

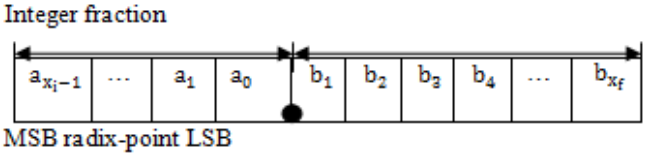

Figure 5: fixed-point representation of a binary number (x_i: integer word-length; x_f: fractional word-length

6. Adder Structure Optimization

The adder -tree and shift-add tree computation can be pruned for further optimization of area, delay, and power complexity. The adder structure is given in fig.6. to reduce the computational complexity, some of the LSBs of inputs of the adder tree can be truncated and the guard bits can be used to minimize the impact of truncation of the error performance of the adaptive filter. To have more hardware saving, the bits to be truncated are not generated by the PPGs, so the complexity of PPGs also gets reduced. To have more hardware saving, the bits to be truncated are not generated by the PPGs, so the complexity of PPGs also gets reduced. To have more hardware saving, the bits to be truncated are not generated by the PPGs, so the complexity of PPGs also gets reduced.

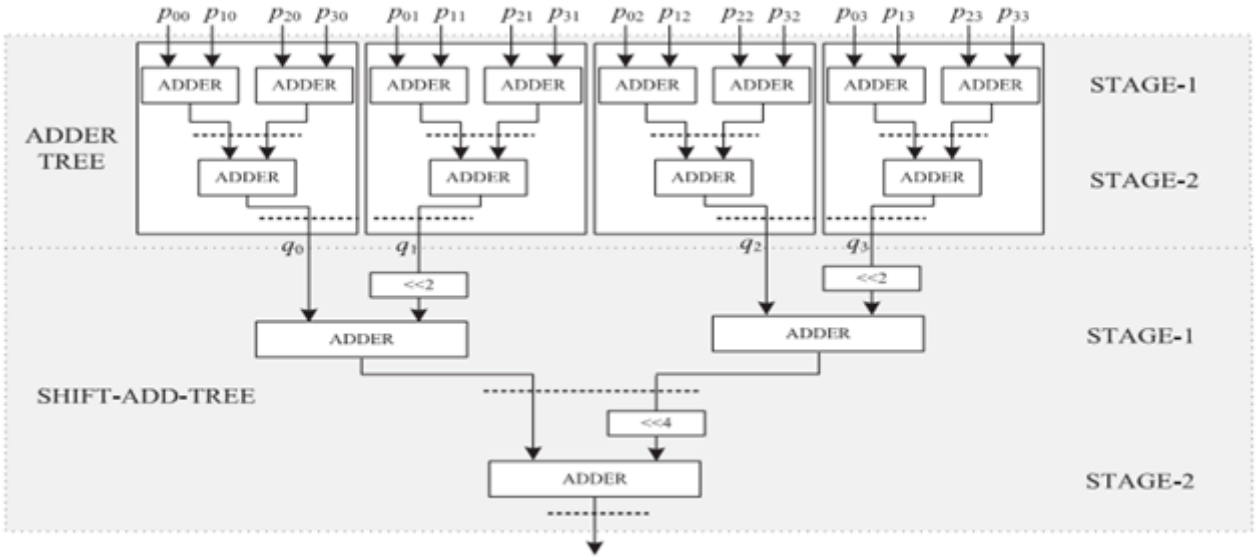

Figure 6: Adder-Structure of the filtering unit for N=4 and L= 8

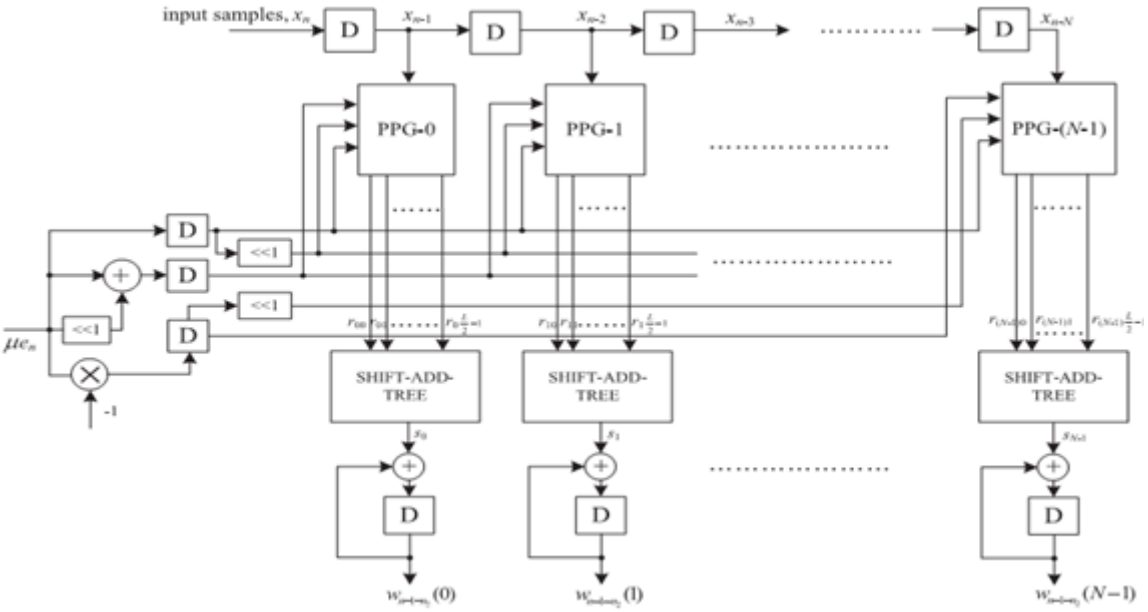

Figure 7: Proposed structure of weight-update block

7. Result Analysis

The simulation results are carried out for Fixed-point LMS adaptive filter to find out the low adaptation delay. The simulation is carried out by Modelsim 6.4c as a simulator tool. The performance of the delay block is simulated by giving various inputs to the weight-update block with various weights. The simulation result is shown in fig. 8, fig. 9 and fig. 10 as a snapshot.

Area Delay Comparison:

The comparison of area and delay between the existing LMS adaptive filter and the proposed modified delay LMS adaptive filter is given below in Table II.

Snapshots:

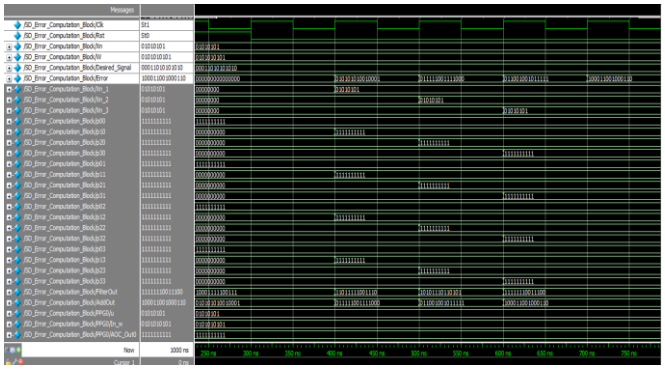

Figure 8: Simulation model of error-computation block

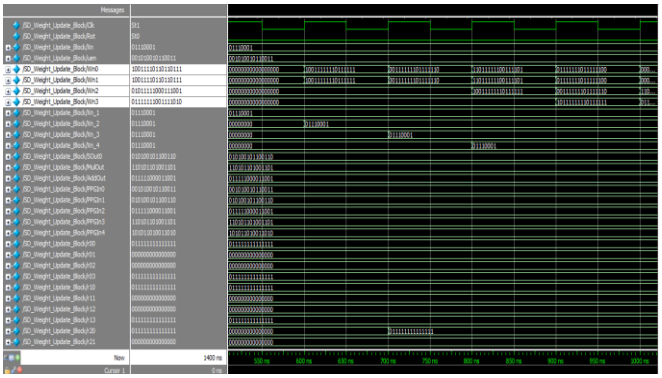

Figure 9: Simulation model of weight-update block during the process

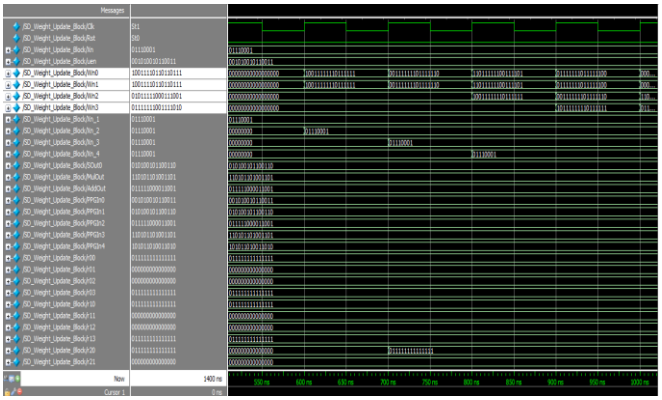

Figure 10: Simulation model with Clk =1 and Rst =0

Table 2

Method name	Area in number of LUT			Memory in kilobytes	Delay		
Filter name	LUT	Gate count	Slices	Size	Delay	Gate or logic delay	Path or route delay
Existing LMS adaptive filter	117	8606	504	198096 kilobytes	0.937ns	12.979ns	9.252ns
Proposed DLMS adaptive filter	108	7902	439	193808 kilobytes	0.936ns	15.643ns	7.774ns

8. Conclusion

An area-delay efficient with low adaptation delay architecture for fixed-point implementation of DLMS adaptive filter are achieved by using a novel PPG for efficient implementation of general multiplications and inner- product computation by common sub expression. From this, proposed strategy an optimized balanced pipelining across the time consuming blocks is to reduce the adaptation delay. The proposed structure involved significantly less adaptation delay and provided significant saving of ADP and EDP compared to the existing structures.

9. Future Scope

The efficient addition scheme reduces the adaptation delay to achieve the faster performance and reduction in the critical path supports the high sampling-rates. The proposed design can be modified by reducing the area and delay of the design for future enhancement.

Reference

[1] G. Long, F. Ling and J. G. Proakis, "The LMS algorithm with delayed coefficient adaptation," IEEE Trans. Acoust, Speech, Signal Process, vol. 37, no. 9, pp. 1397-1405, Sep. 1989.
[2] M. D. Meyer and D. P. Agrawal, "A modular pipelined implementation of a delayed LMS transversal adaptive filter," in Proc. IEEE Int. Symp. Circuits Syst., May 1990, pp. 1943-1946.
[3] H. Herzberg and R. Haimi-Cohen, "Asystolic array realization of an LMS adaptive filter and the effects of delayed adaptation," IEEE Trans. Signal Process, vol. 40, no. 11, pp. 2799-2803, Nov. 1992.
[4] M. D. Meyer and D. P. Agrawal," A high sampling rate delayed LMS filter architecture," IEEE Trans. Circuits Syst. II, Analog Digital Signal Process., vol. 40, no. 11, pp. 727-729, Nov. 1993.
[5] S. Ramanathan and V. Vishvanathan, "A systolic architecture for LMS adaptive filtering with minimal adaptation delay," in Proc. Int. Conf. Very Large Scale Integr.(VLSI) Design, Jan. 1996, pp. 286-289.
[6] R. Rocher, D. Menard, O. Sentieys, and P. Scalart, "Accuracy evaluation of fixed-point LMS algorithm," in Proc. IEEE Int. Conf. Acoust., Speech, Signal Process., May 2004, pp. 237-240.
[7] Y. Yi, R. Woods, L-K Ting, and C. F. N. Cowan, "High speed FPGA-based implementations of delayed-LMS filters," J. Very Large Scale Integr. (VLSI) Signal Process, vol. 39, nos. 1-2, pp. 113-131, Jan. 2005.
[8] 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 Trans. Very Large Scale Integr. (VLSI) Syst., vol. 13, no. 1, pp. 86-99, Jan 2005.
[9] P. K Meher and M. Maheshwari, "a high-speed FIR adaptive filter architecture using a modified delayed LMS algorithm," in Proc. IEEE Int. Symp. Circuits Syst., May 2011, pp. 121-124.
[10] P. K. Meher and S. Y. Park, "Low adaptation-delay LMS adaptive filter part-I: Introducing a novel multiplication cell," in Proc. IEEE Int. Midwest Symp. Circuits Syst., Aug. 2011, pp. 1-4.
[11] Pramod Kumar Meher, Sang Yoon Park, "Area-Delay-Power Efficient Fixed-Point LMS Adaptive Filter With Low Adaptation Delay" IEEE trans on VLSI systems, Vol. 22, No. 2, Feb 2014