INTERNATIONAL JOURNAL OF RESEARCH
In Advanced Engineering Technologies

Volume 4, Issue 2 JULY 2015

Critical Path Analysis and Low Complexity Implementation of the LMS Adaptive Algorithm

B. SHYAMALA [(1)], N. UMARANI [(2)]

M.TECH SCHOLAR, VLSI SYSTEM DESIGN, SIDDHARTHA INSTITUTE OF TECHNOLOGY AND SCIENCES[(1)]
KORREMULA ROAD, NARAPALLY, GHATKESAR MANDAL, RANGA REDDY DIST. TELANGANA, INDIA
M.TECH, ASSOSCIATE PROFESSOR, HOD, SIDDHARTHA INSTITUTE OF TECHNOLOGY AND SCIENCES [(2)]
KORREMULA ROAD, NARAPALLY, GHATKESAR MANDAL, RANGA REDDY DIST. TELANGANA, INDIA

Abstract - This paper presents a precise analysis is of the critical path of the Least - Mean - Square (LMS) adaptive filter for deriving its architectures for high - speed and low-complexity implementation. It is shown that the direct-form LMS adaptive filter has nearly the same critical path as its transpose-form counterpart, but provides much faster convergence and lower register complexity. From the critical-path evaluation, it is further shown that no pipelining is required for implementing a direct-form LMS adaptive filter for most practical cases, and can be realized with a very small adaptation delay in cases where a very high sampling rate is required. Based on these findings, this paper proposes three structures of the LMS adaptive filter: (i) Design 1 having no adaptation delays, (ii) Design 2 with only one adaptation delay, and (iii) Design 3 with two adaptation delays. For comparison of proposed systems its verilog coded and synthesized in Xilinx.

Index Terms - Adaptive filters, critical-path optimization, least mean square algorithms, and LMS adaptive filter.

1. INTRODUCTION

Filters of some sort are essential to the operation of most electronic circuits. It is therefore in the interest of anyone involved in electronic circuit design to have the ability to develop filter circuits capable of meeting a given set of specifications. In circuit theory, a filter is an electrical network that alters the amplitude and/or phase characteristics of a signal with respect to frequency. Ideally, a filter will not add new frequencies to the input signal, nor will it change the component frequencies of that signal, but it will change the relative amplitudes of the various frequency components and/or their phase relationships. Filters are often used in electronic systems to emphasize signals in certain frequency ranges and reject signals in other frequency ranges. Such a filter has a gain which is dependent on signal frequency.

Filters used for direct filtering can be either Fixed or Adaptive.

1. **Fixed filters** - The design of fixed filters requires a priori knowledge of both the signal and the noise, i.e. if we know the signal and noise beforehand; we can design a filter that passes frequencies contained in the signal and rejects the frequency band occupied by the noise.
2. **Adaptive filters** - Adaptive filters, on the other hand, have the ability to adjust their impulse response to filter out the correlated signal in the input. They require little or no a priori knowledge of the signal and noise characteristics (they require a signal (desired response) that is correlated in some sense to the signal to be estimated). Moreover adaptive filters have the capability of adaptively tracking the signal under non-stationary conditions.

The Least Mean Square (LMS) adaptive filter is the most popular and 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. A lot of work has been done to implement the DLMS algorithm in systolic architectures to increase the maximum usable frequency, but, they involve an adaptation delay of ~N cycles for filter length N, which is quite high for large order filters. Since the convergence performance degrades considerably for a large adaptation delay, Visvanathanet al. has proposed a modified systolic architecture to reduce the adaptation delay. A transpose-form LMS adaptive filter is suggested in, where the filter output at any instant depends on the delayed versions of weights and the number of delays in weights varies from 1 to N.

2. ADAPTATION ALGORITHM

The basic configuration of an adaptive filter, operating in the discrete time domain n, is illustrated in Figure 1. In such a scheme, the input signal is denoted by x(n), the reference signal d(n) represents the desired output signal (that usually includes some noise component), y(n) is the output of the adaptive filter, and the error signal is defined as e(n) = d(n) − y(n). The error signal is used by the adaptation algorithm to update the adaptive filter coefficient vector w(n) according to some performance criterion. In general, the whole adaptation process aims at minimizing some metric of the error signal, forcing the adaptive filter output signal to approximate the reference signal in a statistical sense. There are several adaptation algorithms with different performance criterion. Due to its low complexity and proven robustness, Least Mean Square (LMS) algorithm is used here.

LMS algorithm is a noisy approximation of steepest descent algorithm. It is a gradient type algorithm that updates the coefficient vector by taking a step in the direction of the negative gradient of the objective function.

$$w(n+1) = w(k) - \frac{\mu}{2} \frac{\delta Jw}{\delta w(n)}$$

LMS Algorithm:

For each n

$$\mathbf{w}_{n+1} = \mathbf{w}_n + \mu \cdot e_n \cdot \mathbf{x}_n \quad \text{.....(1)}$$

Where,

$$e_n = d_n - y_n \quad y_n = \mathbf{w}_n^T \cdot \mathbf{x}_n \quad \text{.....(2)}$$

Where, the input vector **x***n*, and the weight vector **w***n* at the *n*th iteration are, respectively, given by

$$\mathbf{x}_n = [x_n, x_{n-1}, \ldots, x_{n-N+1}]^T$$
$$\mathbf{w}_n = [w_n(0), w_n(1), \ldots, w_n(N-1)]^T$$

dn is the desired response, *yn* is the filter output, and *en* denotes the error computed during the *n*th iteration. μ is the step-size, and *N* is the number of weights used in the LMS adaptive filter.

2.1 Implementation of Direct - Form Delayed LMS Algorithm

In the case of pipelined designs with *m* pipeline stages, the error *e(n)* becomes available after *m* cycles, where *m* is called the "adaptation delay." The DLMS algorithm therefore uses the delayed error *en−m*, i.e., the error corresponding to *(n − m)*th iteration for updating the current weight instead of the recent-most error. The weight-update equation of DLMS adaptive filter is given by

$$\mathbf{w}_{n+1} = \mathbf{w}_n + \mu \cdot e_{n-m} \cdot \mathbf{x}_{n-m}. \quad \text{.....(3)}$$

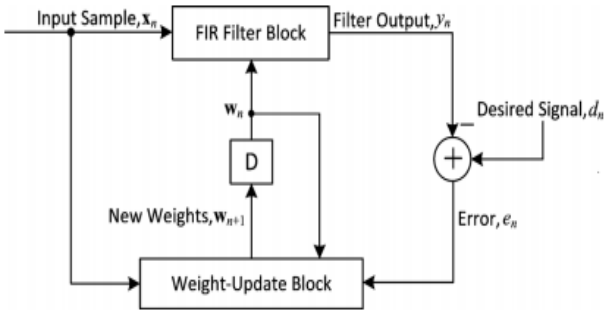


Fig. 1. Structure of the conventional LMS adaptive filter

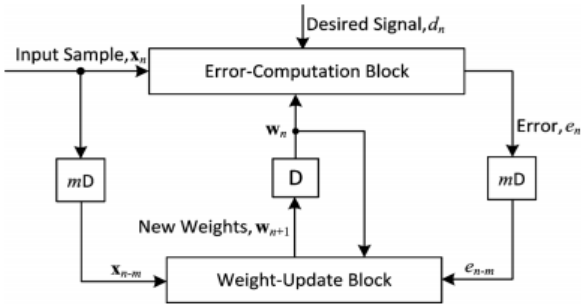


Fig. 2. Generalized block diagram of direct form DLMS adaptive filter

Since all weights are updated concurrently in every cycle to compute the output according to (1), direct-form realization of the FIR filter is a natural candidate for implementation. However, the direct-form LMS adaptive filter is often believed to have a long critical path due to an inner product computation to obtain the filter output. This is mainly based on the assumption that an arithmetic operation starts only after the complete input operand words are available/generated. For example, in the existing literature on implementation of LMS adaptive filters, it is assumed that the addition in a multiply-add operation can proceed only after completion of the multiplication, and with this assumption, the critical path of the multiply-add operation (T_{MA}) becomes (T_{MULT} + T_{ADD}), where T_{MULT} and T_{ADD} are the time required for a multiplication and an addition, respectively. Under such assumption, the critical path of the direct-form LMS adaptive filer (without pipelining) can be estimated as

$$T = 2\,T_{MULT} + (N+1)\,T_{ADD}$$

A generalized block diagram of direct-form DLMS adaptive filter is shown in Fig.2.It consists of an error-computation block (shown in Fig. 3) and a weight-update block (shown in Fig. 4). The number of delays shown in Fig. 2 corresponds to the pipeline delays introduced due to pipelining of the error-computation block.

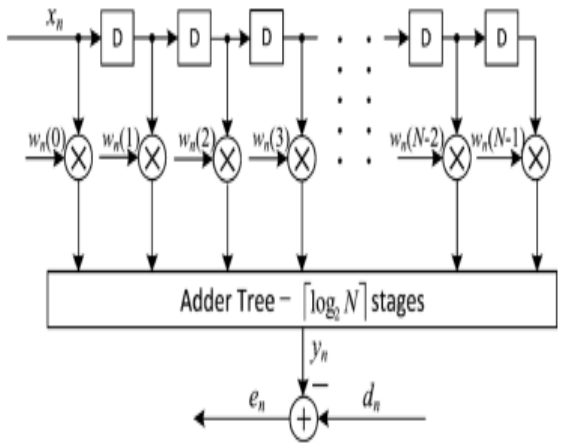


Fig. 3. Error - computation block of Fig .2

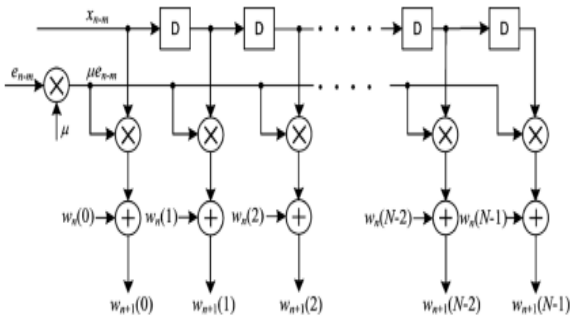


Fig. 4 Weight – update block of Fig. 3

2.2 Implementation of Transpose-Form Delayed LMS Algorithm

The transpose-form FIR structure cannot be used to implement the LMS algorithm given by (1), since the filter output at any instant of time has contributions from filter weights updated at different iterations, where the adaptation delay of the weights could vary from1to .It could, however, be implemented by a different set of equations as follows:

$$y_n = \sum_{k=0}^{N-1} x_{n-k} w_{n-k}(k)$$

$$w_{n+1}(k) = w_n(k) + \mu e_n x_{n-k}$$

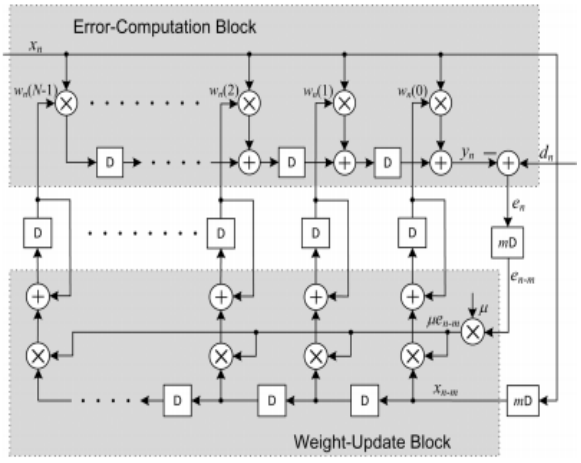


Fig 5: Structure of transpose-form DLMS adaptive filter

If additional delays are introduced in the error computation at any instant, then the weights are required to be updated according to the following equation

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

3. CRITICAL-PATH ANALYSIS OF LMS ADAPTIVE FILTER

The critical path of the LMS adaptive filter of Fig. 1 for direct implementation is given by

$$T = C_{error} + C_{update}$$

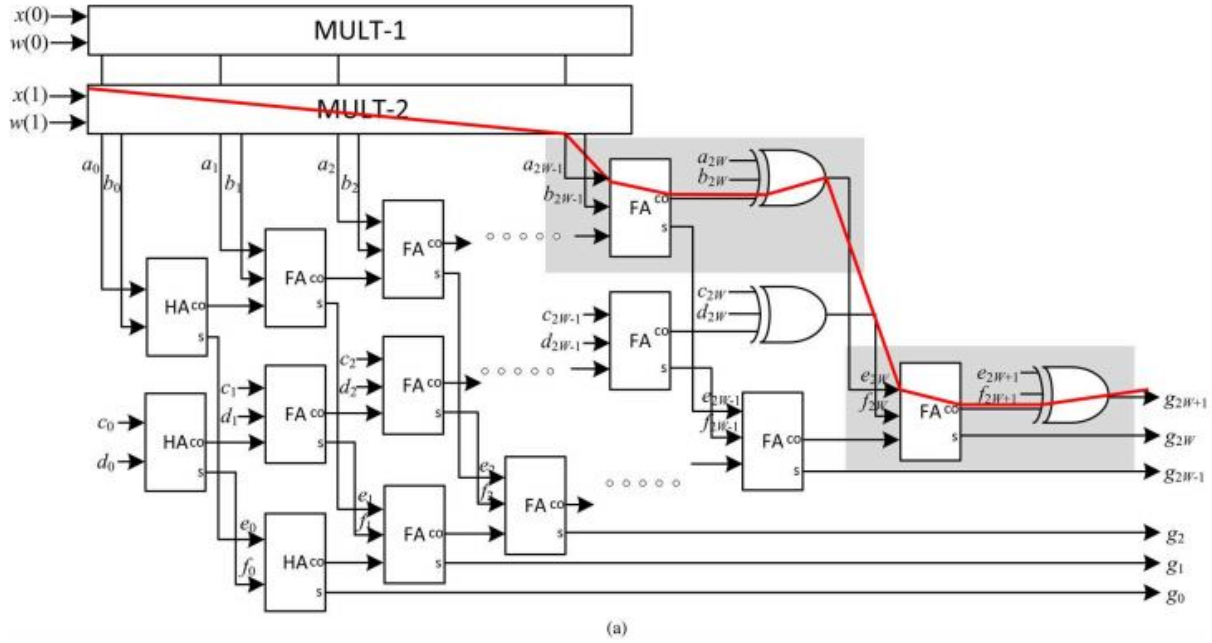
3.1 Critical Path of Direct Form

Let us consider the implementation of an inner product (0)w(0)+x(1)w(1)+x(2)w(2)+x(3)w(3)of length 4 to find out critical path of direct form of adaptive filter. All computation of inner product is done concurrently. Computations of the first-level adders (ADD-1 and ADD-2) are completed in time $T_{MULT} + T_{FAS} + T_{FAC}$, where T_{FAS} is the delay due to the 3-input XOR operation for the addition of the last bits and $T_{FAC} = T_{AND} + T_{XOR}$ are the propagation delays of AND and XOR operations, respectively.

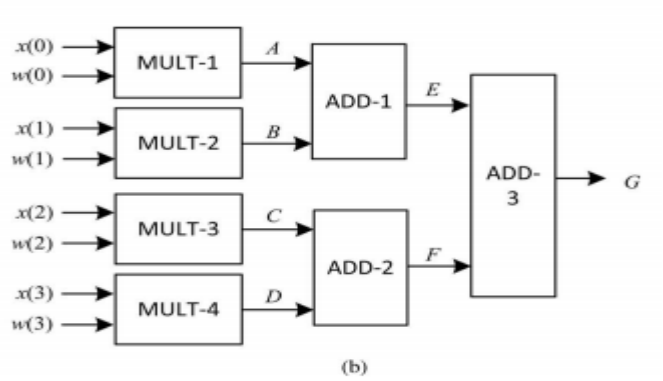
3.2 Critical Path of Transpose Form

In the error-computation block of the transpose form LMS adaptive filter, we can see that all multiplications are performed simultaneously, which involves time. After multiplications, the results are transferred through preceding registers to be added with another product word in the next cycle. Since the addition operation starts as soon as the first bit of the product word is available (as in the direct-form LMS), the critical path of the error-computation block is

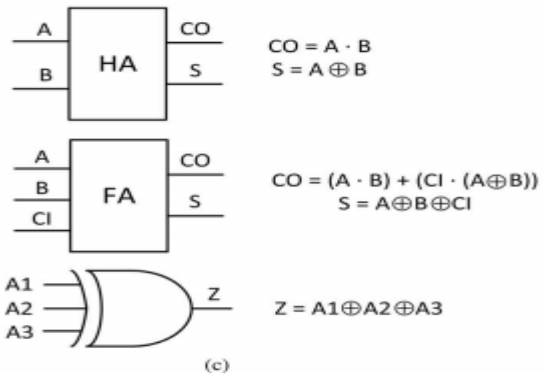
$$C_{\text{ERROR}} = T_{\text{MULT}} + \lceil \log_2(N+1) \rceil \Delta.$$



(a)



(b)



(c)

Fig 6: Critical path of an inner product computation. (a) Detailed block diagram to show critical path of inner product computation of length 4 (b) B lock diagram of inner product computation of x(0)w(0)+x(1)w(1)+x(2)w(2)+x(3)w(3)+x(4)w(4) (c) HA, FA, and 3-input XOR gate.

4. PROPOSED DESIGN

We find that the direct-form FIR structure not only is the natural candidate for implementation of the LMS algorithm in its original form, but also provides better convergence speed with the same residual MSE. It also involves less register complexity and nearly the same critical path as the transpose-form structure. Therefore, we have preferred to design a low-complexity direct-form structure for implementation of the LMS adaptive filter.

4.1 Zero Adaptation Delay

As shown in Fig. 2, there are two main computing blocks in the direct-form LMS adaptive filter, namely, i) the error-computation block (shown in Fig. 3) and ii) the weight-update block (shown in Fig. 4). It can be observed in Figs. 3 and 4 that most of the area-intensive components are common in the error-computation and weight-update blocks: the multipliers, weight registers, and tapped-delay line. The adder tree and subtractor in Fig. 4 and the adders for weight updating in Fig. 5, which constitute only a small part of the circuit, are different in these two computing blocks. For the zero-adaptation-delay implementation, the computation of both these blocks is required to be performed in the same cycle. Moreover, since the structure is of the 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, while the same registers could be used for both these phases if error computation is performed in the first half cycle , while w eight update is performed in the second - half cycle .

4.2 One Adaptation Delay

The proposed structure for a one-adaptation-delay LMS adaptive filter consists of one error computation unit as shown in Fig. 3 and one weight-update unit as shown in Fig.4. A pipeline latch is introduced after computation of μe_n . The multiplication with requires only a hardwired shift, since is assumed to be a power of 2 fraction. So there is no register overhead in pipelining. Also, the registers in the tapped delay line and filter weights can be shared by the error-computation unit and weight-updating unit. The critical path of this structure is the same as C_{error} given by

$$T = T_{\mathrm{MULT}} + (\lceil \log_2 N \rceil + 1)\Delta.$$

4.3 Two Adaptation Delays

The proposed structure for a two-adaptation-delay LMS adaptive filter is shown in Fig. 8, which consists of three pipeline stages, where the first stage ends after the first level of the adder tree in the error-computation unit, and the rest of the error-computation block comprises the next pipeline stage. The weight-update block comprises the third pipeline stage. The two-adaptation-delay structure involves additional registers over the one-adaptation-delay structure. The critical path of this structure is the same as either that of the weight-update unit C_{update} or the second pipeline stage, given by

$$T = \max\{T_{\mathrm{MULT}} + \Delta, T_{\mathrm{AB}}\}$$

Where T_{AB} refers to the adder-tree delay of stages to add N/2 words along with the time required for subtraction in the error computation.

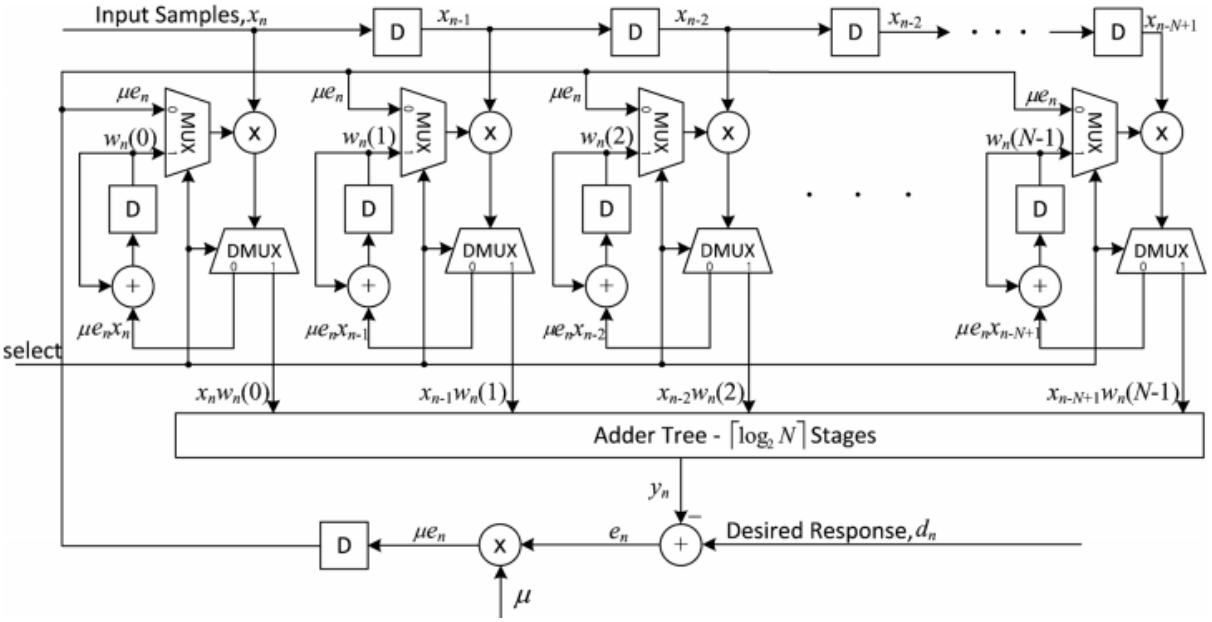


Fig .7. Proposed structure for zero - adaptation – delay time - multiplexed direct – form LMS adaptive filter

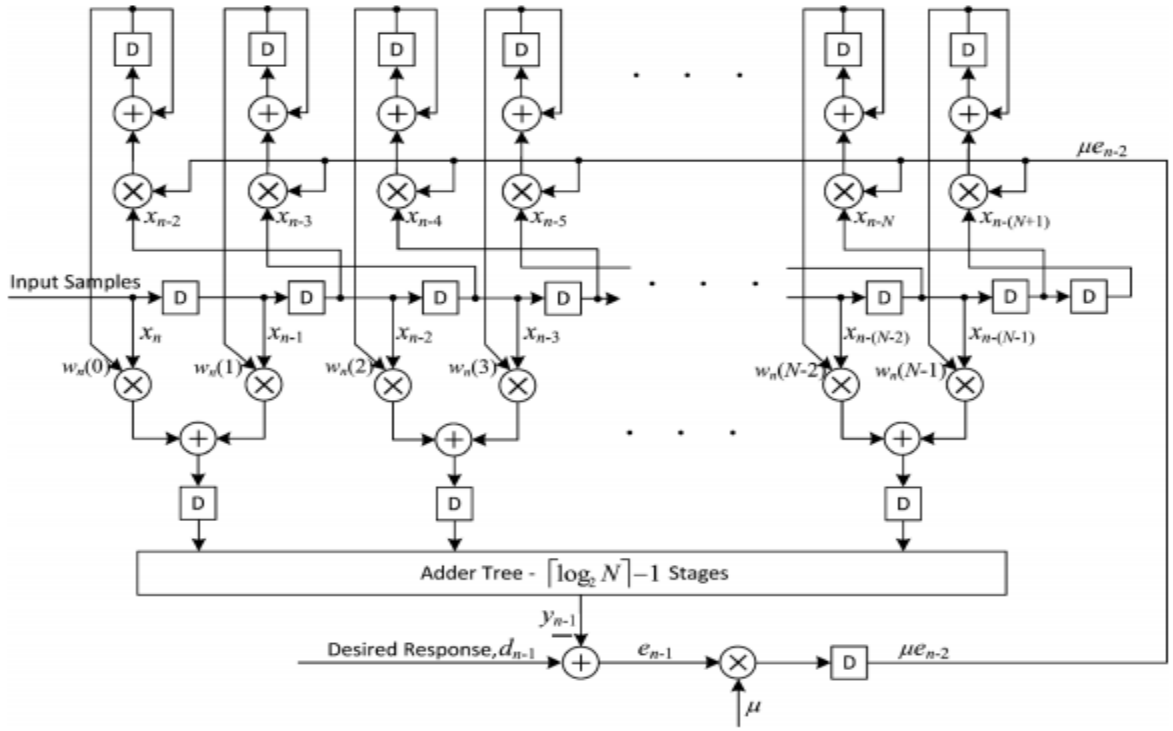


Fig. 8. Proposed structure for two-adaptation-delay direct-form LMS adaptive filter

5. SIMULATION RESULT

Low complexity adaptive filter is verilog coded and simulated on Xilinx to check the desired functionality .The critical path of direct form of FIR filter is also analyzed. The filter structured in Verilog is synthesized on Xilinx ISE.Fig 9.shows simulation result of conventional adaptive filter and fig.10 shows simulation result of proposed structure.

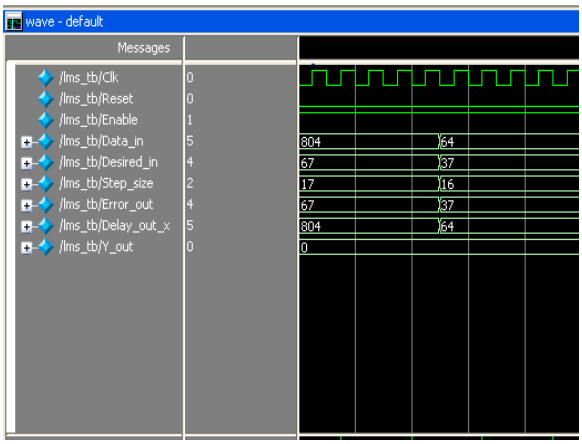


Fig 9: Simulation result of conventional adaptive filter

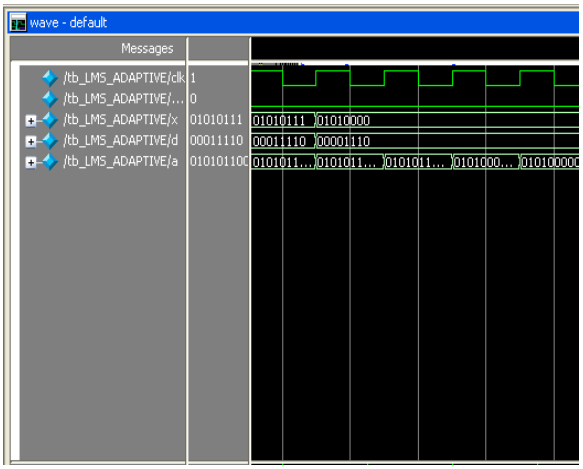


Fig 10: Simulation result of proposed structure

CONCLUSION

Low complexity LMS adaptive filter is proposed and critical path is also analyzed using direct form and transposed form of FIR filter .It is found that critical path of direct and transposed form is same but direct form is having less number of registers means low complexity than transposed .For direct form three systems are proposed .They are following:1)zero adaptation delay 2)one adaptation delay 3) two adaptation delay .It is found that for conventional adaptive filter delay is 19.732 ns and for proposed system delay is 4.182 ns.

REFERENCES

[1] B. Widrow and S. D. Stearns, *Adaptive Signal Processing*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1985.

[2] S. Haykin and B. Widrow, *Least-Mean-Square Adaptive Filters*. Hoboken, NJ, USA: Wiley-Interscience, 2003.

[3] 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.

[4] 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.

[5] L. D. Van and W. S. Feng, "An efficient systolic architecture for the DLMS adaptive filter and its applications," *IEEE Trans. Circuits Syst.II, Analog Digit. Signal Process.*, vol. 48, no. 4, pp. 359–366, Apr.2001.

[6] 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.

[7] E. Mahfuz, C. Wang, and M. O. Ahmad, "A high-throughput DLMS adaptive algorithm," in*Proc.IEEEInt.Symp.CircuitsSyst.*, May 2005, pp. 3753–3756.

[8] P. K. Meher and S. Y. Park, "Low adaptation-delay LMS adaptive filter Part-II: An optimized architecture," in *Proc. IEEE Int. Midwest Symp.Circuits Syst.*, Aug. 2011.

[9] P. K. Meher and S. Y. Park, "Area-delay-power efficient fixed-point LMS adaptive filter with low adaptation-delay," *Trans. Very Large Scale Integer. (VLSI) Signal Process.* [Online]. Available: http://ieeexplore.ieee.org

[10] M. Z. U. Rahman, R. A. Shaik, and D. V. R. K. Reddy, "Adaptive noise removal in the ECG using the block LMS algorithm," in *Proc.*

IEEE Int. Conf. Adaptive Sci. Technol. , Jan. 2009, pp. 380–383.

[11] B. Widrow, J. R. Glover, Jr., J. M. McCool, J. Kaunitz, C. S. Williams ,R. H. Hearn, J. R. Zeidler, E. Dong, Jr., and R. C. Goodlin, "Adaptive noise cancelling: Principles and applications," *Proc. IEEE*, vol. 63, no. 12, pp. 1692–1716, Dec. 1975.

[12] W. A. Harrison, J. S. Lim, and E. Singer, "A new application of adaptive noise cancellation," *IEEE Trans. Acoust., Speech, Signal Process.* , vol. 34, no. 1, pp. 21–27, Feb. 1986.

[13] S. Coleri, M. Ergen, A. Puri, and A. Bahai, "A study of channel estimation in OFDM systems," in *Proc. IEEE Veh. Technol. Conf.*, 2002, pp. 894–898.

[14] J. C. Patra, R. N. Pal, R. Baliarsingh, and G. Panda, "Nonlinear channel equalization for QAM signal constellation using artificial neural networks," *IEEE Trans. Syst., Man, Cybern. B, Cybern.* , vol. 29, no. 2, pp. 262–271, Apr. 1999.

[15] D. Xu and J. Chiu, "Design of a high-order FIR digital filtering and variable gain ranging seismic data acquisition system," in *Proc. IEEE Southeastcon*, Apr. 1993.