ISSN No: 2348-4845
International Conference on Electronics, Communications and VLSI Circuits (ICECV-2015)
July 10, 2015 - Hyderabad, India
Papers Published in IJMETMR, A Peer Reviewed Open Access International Journal.

Low Adaption Area-Delay Power-Efficient Fixed Point LMS Adaptive Filter

N.Pavani
PG Scholar
Department of ECE
Vaagdevi College of Engineering
Bollikunta, Warangal-506001

V.Sabitha
Asst. Professor
Department of ECE
Vaagdevi College of Engineering
Bollikunta, Warangal-506001

P Prasad rao
Professor
Department of ECE
Vaagdevi College of Engineering
Bollikunta, Warangal-506001

Abstract:

In this paper, we present an efficient architecture for the implementation of a delayed least mean square adaptive filter. For achieving lower adaptation-delay and area-delay-power efficient implementation, we use a novel partial product generator and propose a strategy for optimized balanced pipelining across the time-consuming combinational blocks of the structure. From synthesis results, we find that the proposed design offers nearly 13% less area-delay product (ADP) and nearly 10% less energy-delay product (EDP) than the best of the existing systolic structures, on average, for filter lengths N = 8, 16, and 32.

We propose an efficient fixed-point implementation scheme of the proposed architecture, and derive the expression for steady-state error. We show that the steady-state mean squared error obtained from the analytical result matches with the simulation result.

Moreover, we have proposed a bit-level pruning of the proposed architecture, which provides nearly 25% saving in ADP and 10% saving in EDP over the proposed structure before pruning without noticeable degradation of steady-state-error performance.

Keywords: ***ADP, DLMS, EDP, Bit level Pruning, Error Performance, square adaptive filter.***

1.Introduction

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 [3]–[5], which allows pipelined implementation of the filter.

The existing work on the DLMS adaptive filter does not discuss the fixed-point implementation issues, e.g., 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, we present here the optimization of our previously reported 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) compared to the existing structures. In the

ISSN No: 2348-4845
International Conference on Electronics, Communications and VLSI Circuits (ICECV-2015)
July 10, 2015 - Hyderabad, India
Papers Published in IJMETMR, A Peer Reviewed Open Access International Journal.

next section, we review the DLMS algorithm, and in Section III, we describe the proposed optimized architecture for its implementation. Section IV deals with fixed-point implementation considerations and simulation studies of the convergence of the algorithm. In Section V, we discuss the synthesis of the proposed architecture and comparison with the existing architectures. Conclusions are given in Section VI.

2. Review Of Delayed Lms Algorithm:

The weights of LMS adaptive filter during the *n*th iteration are updated according to the following equations [2]:

$$W_{n+1}=W_n+\mu.\, e_n.\, x_n \qquad (1a)$$

Where

$$e_n = d_n - y_n\,,\; y_n = w_n^T.\, x_n \qquad (1b)$$

Where the input x_n, and weight vector W_n at the nth iteration are, respectively, given by

$$x_n=[x_n, x_{n-1}, x_{n-2}\ldots\ldots x_{n-N=1}]^T$$

$$W_n=[w_n(0), w_n(1)\ldots.w_n(N-1)]^T$$

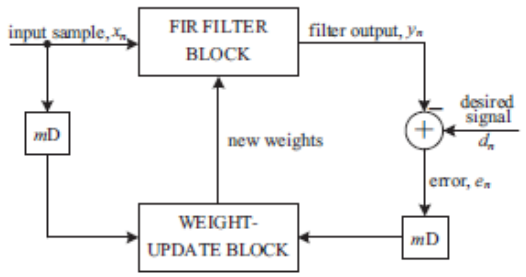


Fig. 1. Structure of the conventional delayed LMS adaptive filter.

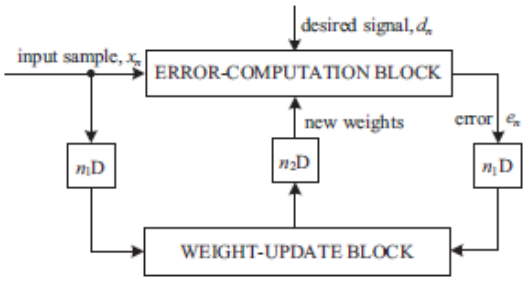


Fig. 2. Structure of the modified delayed LMS adaptive filter.

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. In the case of pipelined designs with *m* pipeline stages, the error *en* 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

$$W_{n+1}= w_n+ \mu.\, e_{n-m.}\, x_{n-m} \qquad (2)$$

The block diagram of the DLMS adaptive filter is shown in Fig. 1, where 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. It is shown in [12] that 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 Fig. 2. Assuming that the latency of computation of error is *n*1 cycles, the error computed by the structure at the *n*th cycle is *en−n*1 , which is used with the input samples delayed by *n*1 cycles to generate the weight-increment term. The weight-update equation of the modified DLMS algorithm is given by

$$w_{n+1} = w_n + \mu \cdot e(n-n1) \cdot x(n-n1) \qquad (3a)$$

Where,

$$e(n-n1) = d(n-n1) - y(n-n1) \qquad (3b)$$

and

$$y(n) = w^T_{n-n2} \cdot x(n) \qquad (3c)$$

ISSN No: 2348-4845
International Conference on Electronics, Communications and VLSI Circuits (ICECV-2015)
July 10, 2015 - Hyderabad, India
Papers Published in IJMETMR, A Peer Reviewed Open Access International Journal.

We notice that, during the weight update, the error with $n1$ delays is used, while the filtering unit uses the weights delayed by $n2$ cycles. The modified DLMS algorithm decouples computations of the error-computation block and the weight-update block and allows us to perform optimal pipelining by feed forward cut-set retiming of both these sections separately to minimize the number of pipeline stages and adaptation delay.

3. Proposed Architecture

As shown in Fig. 2, there are two main computing blocks in the adaptive filter architecture:
1) the error-computation block, and
2) weight-update block.

In this Section, we discuss the design strategy of the proposed structure to minimize the adaptation delay in the error-computation block, followed by the weight-update block.

A. Pipelined Structure of the Error-Computation Block

The proposed structure for error-computation unit of an N-tap DLMS adaptive filter is shown in Fig. 4. It consists of N number of 2-b partial product generators (PPG) corresponding to N multipliers and a cluster of $L/2$ binary adder trees, followed by a single shift–add tree. Each sub block is described in detail.

1) Structure of PPG:

The structure of each PPG is shown in Fig. 5. It consists of $L/2$ number of 2-to-3 decoders and the same number of AND/OR cells (AOC).1 Each of the 2-to-3 decoders takes a 2-b digit $(u1u0)$ as input and produces three outputs $b0 = u0 \cdot . u1$, $b1 = . u0 \cdot u1$, and $b2 = u0 \cdot u1$, such that $b0 = 1$ for $(u1u0) = 1$, $b1 = 1$ for $(u1u0) = 2$, and $b2 = 1$ for $(u1u0) = 3$. The decoder output $b0$, $b1$ and $b2$ along with w, $2w$, and $3w$ are fed to an AOC, where w, $2w$, and $3w$ are in 2's complement representation and sign-extended to have $(W + 2)$ bits each.

To take care of the sign of the input samples while computing the partial product corresponding to the most significant digit (MSD), i.e., $(uL-1uL-2)$ of the input sample, the AOC $(L/2 - 1)$ is fed with w, $-2w$, and $-w$ as input since $(uL-1uL-2)$ can have four possible values 0, 1, −2, and −1.

2) Structure of AOCs:

The structure and function of an AOC are depicted in Fig. 6. 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. 6(b) and (c), respectively. Each AND cell takes an n-bit input D and a single bit input b, and consists of n AND gates.

It distributes all the n bits of input D to its n AND gates as one of the inputs.

The other inputs of all the n AND gates are fed with the single-bit input b. As shown in Fig. 6(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-b input $(u1u0)$, respectively.

The decoder along with the AOC performs a multiplication of input operand w with a 2-b digit $(u1u0)$, such that the PPG of Fig. 5 performs $L/2$ parallel multiplications of input word w with a 2-b digit to produce $L/2$ partial products of the product word wu.

ISSN No: 2348-4845
International Conference on Electronics, Communications and VLSI Circuits (ICECV-2015)
July 10, 2015 - Hyderabad, India
Papers Published in IJMETMR, A Peer Reviewed Open Access International Journal.

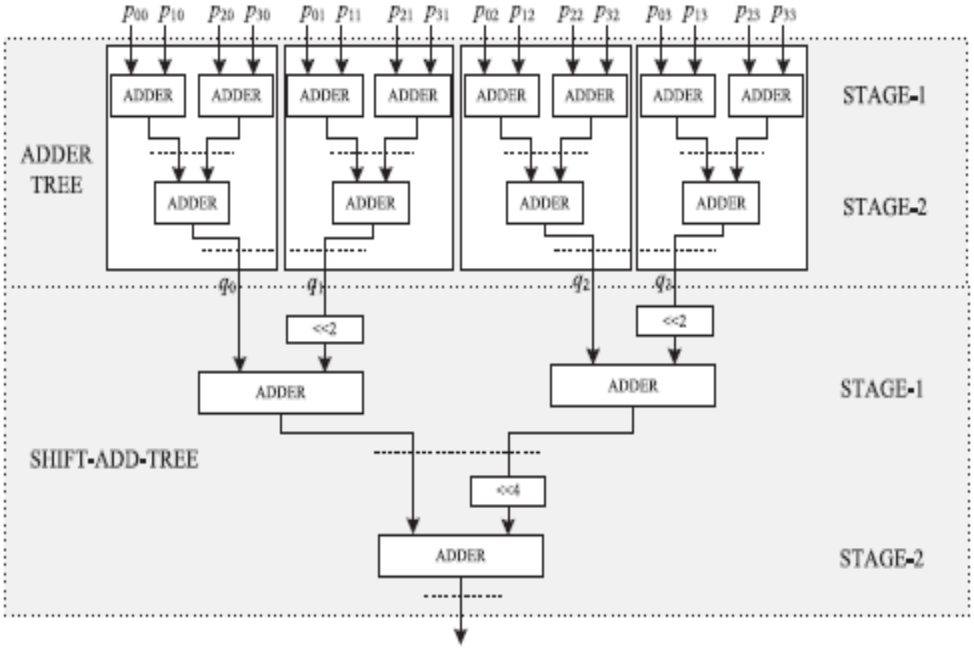


Fig. 7. Adder-structure of the filtering unit for $N = 4$ and $L = 8$.

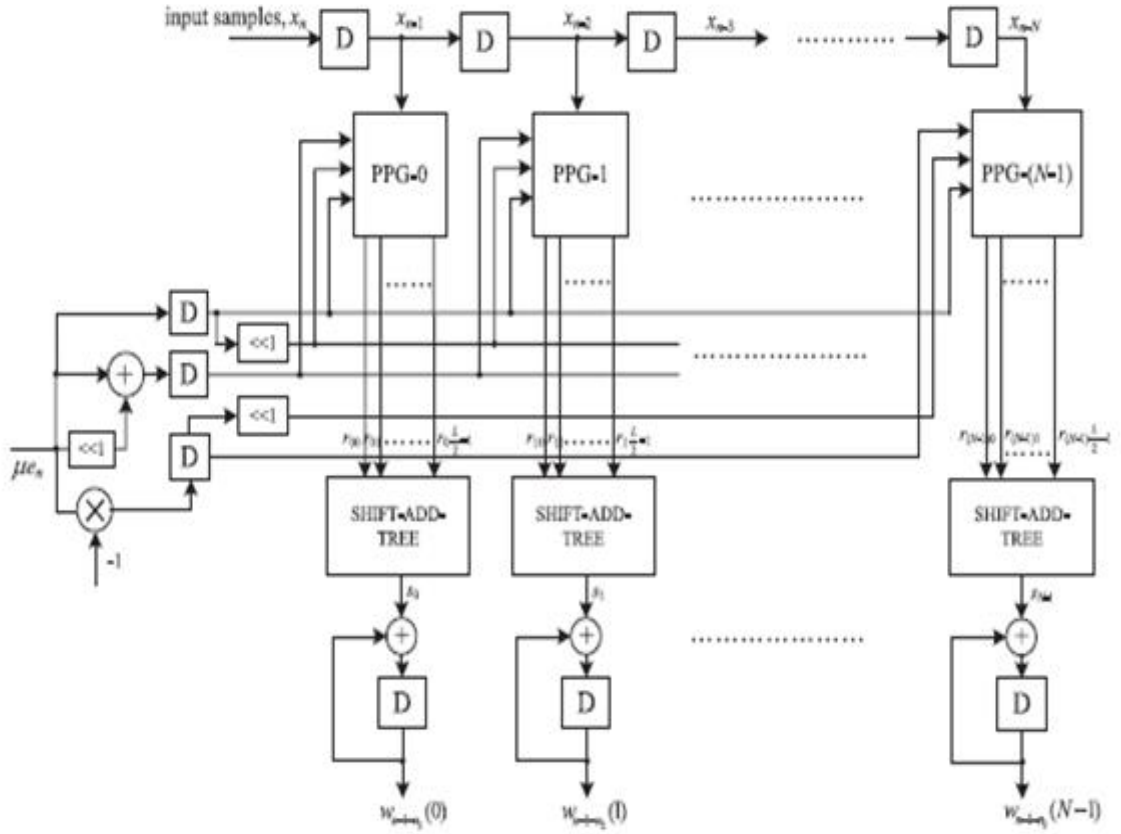


Fig. 8. Proposed structure of the weight-update block.

ISSN No: 2348-4845
International Conference on Electronics, Communications and VLSI Circuits (ICECV-2015)
July 10, 2015 - Hyderabad, India
Papers Published in IJMETMR, A Peer Reviewed Open Access International Journal.

3) Structure of Adder Tree:

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* product 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. All the *L*/2 partial products generated by each of the *N* PPGs are thus added by *(L/2)* binary adder trees.

The outputs of the *L*/2 adder trees are then added by a shift-add tree according to their place values. Each of the binary adder trees require log2 *N* stages of adders to add *N* partial product, and the shift–add tree requires log2 $L - 1$ stages of adders to add *L*/2 output of *L*/2 binary adder trees.2 The addition scheme for the error-computation block for a four-tap filter and input word size $L = 8$ is shown in Fig. 7.

For $N = 4$ and $L = 8$, the adder network requires four binary adder trees of two stages each and a two-stage shift–add tree. In this figure, we have shown all possible locations of pipeline latches by dashed 2When *L* is not a power of 2, log2 *L* should be replaced by _log2 *L*_

TABLE I

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

lines, to reduce the critical path to one addition time. If we introduce pipeline latches after every addition, it would require $L(N - 1)/2 + L/2 - 1$ latches in log2 N + log2 $L - 1$ stages, which would lead to a high adaptation delay and introduce a large overhead of area and power consumption for large values of *N* and *L*. On the other hand, some of those pipeline latches are redundant in the sense that they are not required to maintain a critical path of one addition time. The final adder in the shift–add tree contributes to the maximum delay to the critical path. Based on that observation, we have identified the pipeline latches that do not contribute significantly to the critical path and could exclude those without any noticeable increase of the critical path. The location of pipeline latches for filter lengths $N = 8$, 16, and 32 and for input size $L = 8$ are shown in Table I. The pipelining is performed by a feed forward cut-set retiming of the error-computation block.

B. Pipelined Structure of the Weight-Update Block: The proposed structure for the weight-update block is shown in Fig. 8. It performs *N* multiply-accumulate operations of the form $(\mu \times e) \times xi + wi$ to update *N* filter weights. The step size μ is taken as a negative power 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 *xi* followed by the additions with the corresponding old weight values *wi* . 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 partialproducts.

ISSN No: 2348-4845
International Conference on Electronics, Communications and VLSI Circuits (ICECV-2015)
July 10, 2015 - Hyderabad, India
Papers Published in IJMETMR, A Peer Reviewed Open Access International Journal.

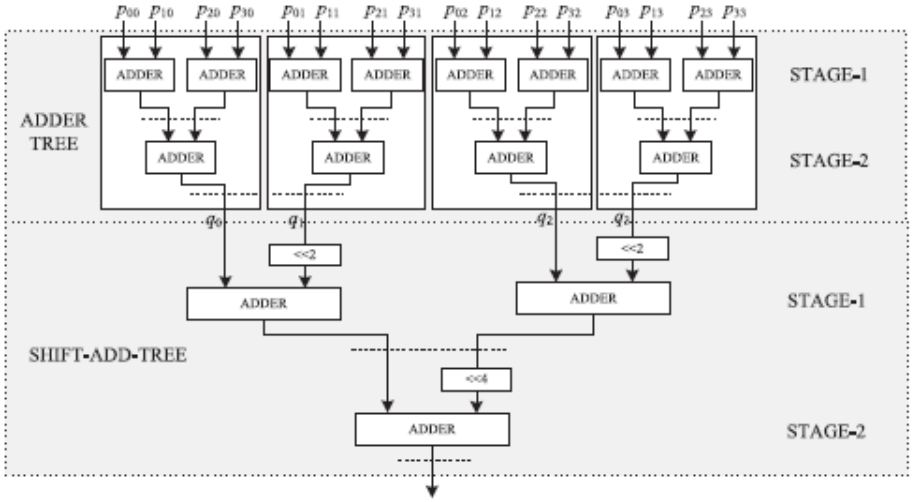


Fig. 7. Adder-structure of the filtering unit for $N = 4$ and $L = 8$.

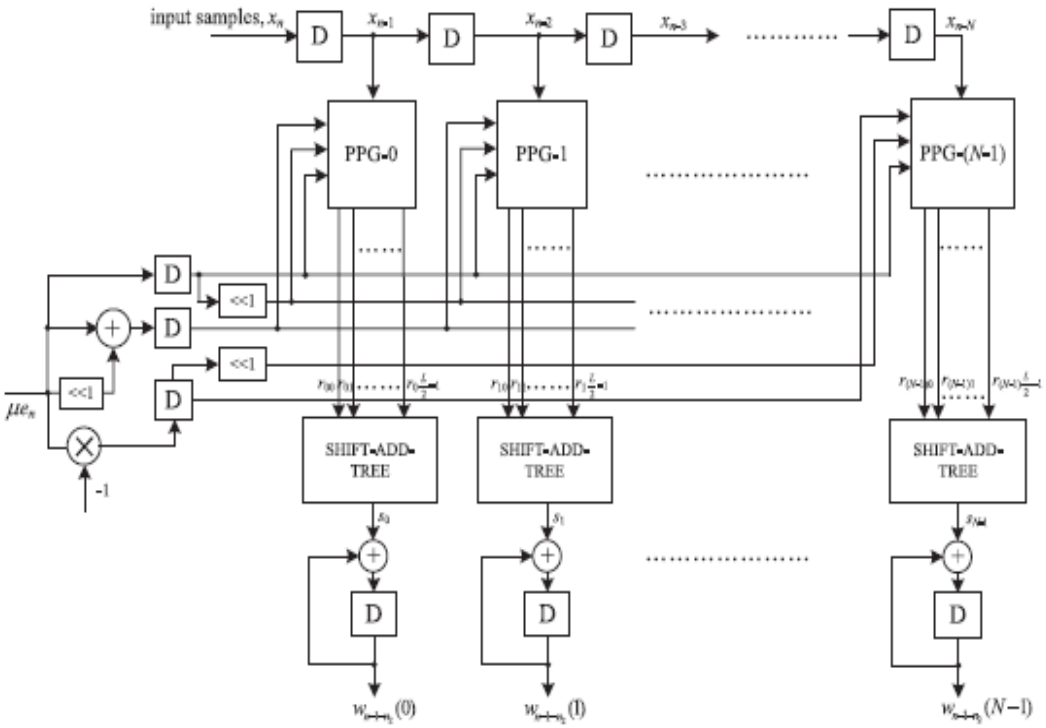


Fig. 8. Proposed structure of the weight-update block.

Corresponding to the product of the recently shifted error value $\mu \times e$ with $L/2$, the number of 2-b digits of the input word xi , where the sub expression $3\mu \times e$ is shared within the multiplier. Since the scaled error ($\mu \times e$) is multiplied with all the 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 outputs of MAC units constitute the desired updated weights to be used as inputs to the error-

ISSN No: 2348-4845
International Conference on Electronics, Communications and VLSI Circuits (ICECV-2015)
July 10, 2015 - Hyderabad, India
Papers Published in IJMETMR, A Peer Reviewed Open Access International Journal.

computation block as well as the weight-update block for the next iteration.

C. Adaptation Delay

As shown in Fig. 2, the adaptation delay is decomposed into $n1$ and $n2$. The error-computation block generates the delayed error by $n1 - 1$ cycles as shown in Fig. 4, which is fed to the weight-update block shown in Fig. 8 after scaling by μ; then the input is delayed by 1 cycle before the PPG to make the total delay introduced by FIR filtering be $n1$. In Fig. 8, the weight-update block generates $\mathbf{w}n-1-n2$, and the weights are delayed by $n2+1$ cycle. However, it should be noted that the delay by 1 cycle is due to the latch before the PPG, which is included in the delay of the error-computation block, i.e., $n1$. Therefore, the delay generated in the weight-update block becomes $n2$. If the locations of pipeline latches are decided as in Table I, $n1$ becomes 5, where three latches are in the error-computation block, one latch is after the subtraction in Fig. 4, and the other latch is before PPG in Fig. 8. Also, $n2$ is set to 1 from a latch in the shift-add tree in the weight-update block.

D. Adder-Tree Optimization

The adder tree and shift–add tree for the computation of yn can be pruned for further optimization of area, delay, and power complexity. To illustrate the proposed pruning optimization of adder tree and shift–add tree for the computation of filter output, we take a simple example of filter length $N = 4$, considering the word lengths L and W to be 8. The dot diagram of the adder tree is shown in Fig. 11. Each row of the dot diagram contains 10 dots, which represent the partial products generated by the PPG unit, for $W = 8$. We have four sets of partial products corresponding to four partial products of each multiplier, since $L = 8$. Each set of partial products of the same weight values contains four terms, since $N = 4$. The final sum without truncation should be 18 b. However, we use only 8 b in the final sum, and the rest 10 b are finally discarded. To reduce the computational complexity, some of the LSBs of inputs of the adder tree can be truncated, while some guard bits can be used to minimize the impact of truncation on the error performance of the adaptive filter. In Fig. 11, four bits are taken as the guard bits and the rest six LSBs are truncated. To have more hardware saving, the bits to be truncated are not generated by the PPGs, so the complexity of PPGs also gets reduced.

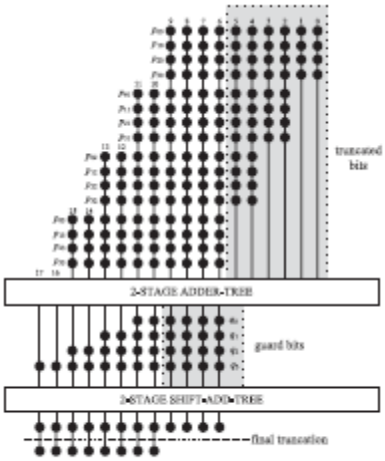


Fig. 11. Dot-diagram for optimization of the adder tree in the case of $N = 4$, $L = 8$, and $W = 8$.

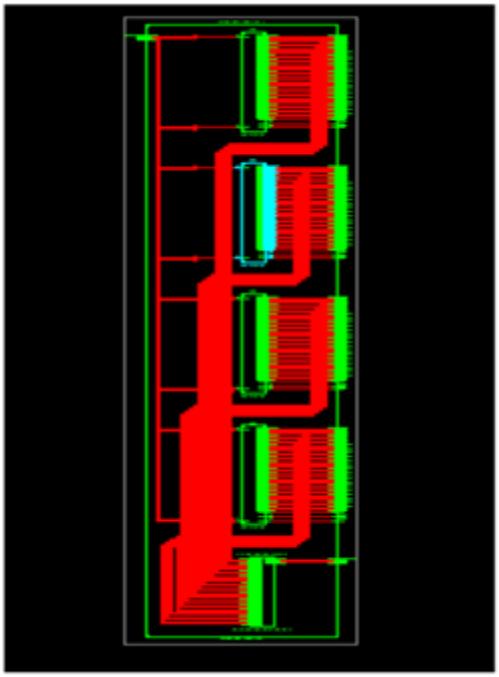
4. Perfomance Results

This section evaluates the performance of the proposed modified least mean square (LMS) algorithm and shows the simulation results. The first result declares about the output of LMS adaptive filter with delay. It is having some delay in the output of Least Mean Square adaptive filter. And the second result declares about the output of LMS adaptive filter without delay. After the clock input has given the output of the adaptive filter is achieved without delay. The ModelSIM is the tool used here to check the performance of LMS adaptive filter. It is a complete HDL simulation environment that enables to verify the source code

ISSN No: 2348-4845
International Conference on Electronics, Communications and VLSI Circuits (ICECV-2015)
July 10, 2015 - Hyderabad, India
Papers Published in IJMETMR, A Peer Reviewed Open Access International Journal.

and functional and timing models using test bench.

Schematic Diagram of The Project



Output Waveforms

OUTPUT OF LMS ADAPTIVE FILTER WITH DELAY

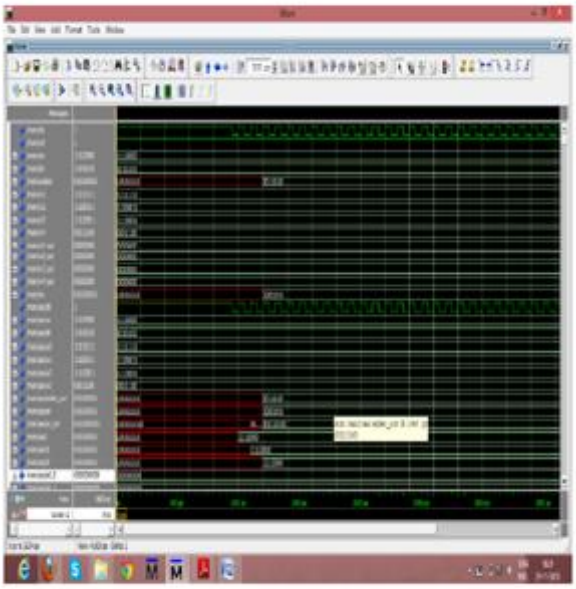


Fig 6: Output with delay

OUTPUT OF LMS ADAPTIVE FILTER WITHOUT DELAY

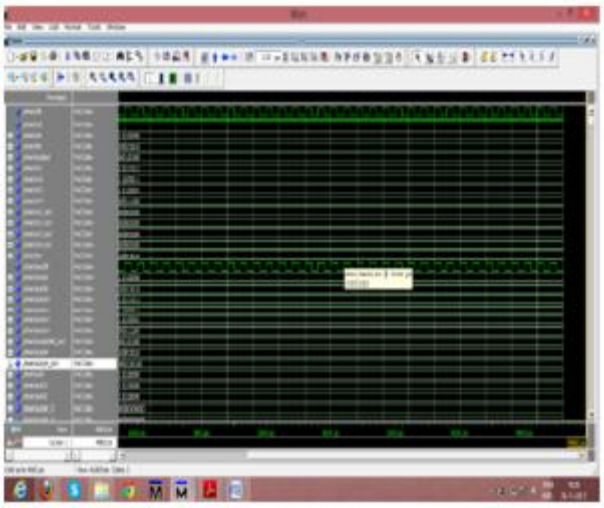
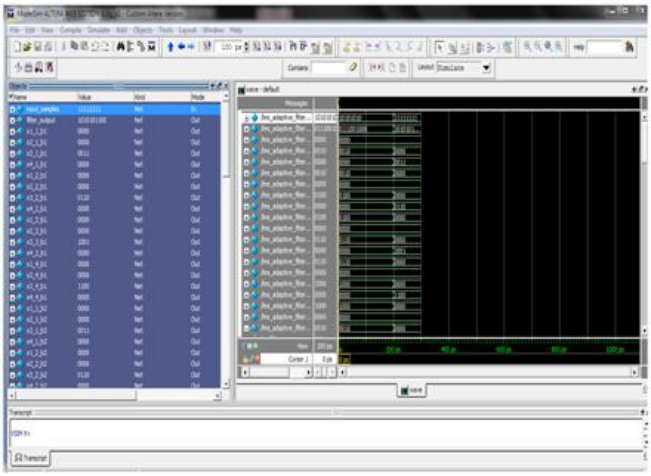


Fig 7: Output without delay

Output of the project



5. Conclusion

We proposed an area–delay-power efficient low adaptation delay architecture for fixed-point implementation of LMS adaptive filter. We used a novel PPG for efficient implementation of general multiplications and inner-product computation by common sub expression sharing. Besides, we have proposed an efficient addition scheme for inner-product computation 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.

ISSN No: 2348-4845
International Conference on Electronics, Communications and VLSI Circuits (ICECV-2015)
July 10, 2015 - Hyderabad, India
Papers Published in IJMETMR, A Peer Reviewed Open Access International Journal.

Aside from this, we proposed a strategy for optimized balanced pipelining across the time-consuming blocks of the structure to reduce the adaptation delay and power consumption, as well. The proposed structure involved significantly less adaptation delay and provided significant saving of ADP and EDP compared to the existing structures.

We proposed a fixed-point implementation of the proposed architecture, and derived the expression for steady-state error. We found that the steady-state MSE obtained from the analytical result matched well with the simulation result. We also discussed a pruning scheme that provides nearly 25% saving in the ADP and 10% saving in EDP over the proposed structure before pruning, without a noticeable degradation of steady-state error performance. The highest sampling rate that could be supported by the ASIC implementation of the proposed design ranged from about 870 to 1010 MHz for filter orders 8 to 32. When the adaptive filter is required to be operated at a lower sampling rate, one can use the proposed design with a clock slower than the maximum usable frequency and a lower operating voltage to reduce the power consumption further.

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, 2003.

[3] 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.

[4] 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.

[5] G. Long, F. Ling, and J. G. Proakis, "Corrections to 'The LMS algorithm with delayed coefficient adaptation'," *IEEE Trans. Signal Process.*, vol. 40, no. 1, pp. 230–232, Jan. 1992.

[6] H. Herzberg and R. Haimi-Cohen, "A systolic 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.

[7] 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.

[8] S. Ramanathan and V. Visvanathan, "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.

[9] 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.

[10] 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 Digital Signal Process.*, vol. 48, no. 4, pp. 359–366, Apr. 2001.

[11] 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.

[12] 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.

[13] 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.

[14] 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, pp. 1–4.

[15] K. K. Parhi, *VLSI Digital Signal Procesing Systems: Design and Implementation*. New York, USA: Wiley, 1999.

[16] C. Caraiscos and B. Liu, "A roundoff error analysis of the LMS adaptive algorithm," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 32, no. 1, pp. 34–41, Feb. 1984.

[17] R. Rocher, D. Menard, O. Sentieys, and P. Scalart, "Accuracy evaluation of fixed-point LMS algorithm," in *Proc. IEEE Int. Conf. Acoust., Speech,*

AUTHOR 1:-

N.Pavani completed her B tech in Khammam Institute of Technology and science and pursuing M-Tech in Vaagdevi College of Engineering.

AUTHOR 2:-

Ms. V. Sabitha is working as Asst. prof in Dept of ECE Vaagdevi College of Engineering Bollikuntla, Warangal.

AUTHOR 3:-

Professor Mr.P Prasad Rao, Head of the Department of ECE Vaagdevi College of Engineering Bollikuntla, Warangal.