ISSN: 2278 – 909X
International Journal of Advanced Research in Electronics and Communication Engineering (IJARECE)
Volume 6, Issue 8, August 2017

Delay Power Efficient DLMS Adaptive Filter With Low Adaptation Delay Using Kogge Stone Adder

Swapnakumari Peethala, Rajasekhar Turaka

Abstract -In this paper, we introduce an effective architecture for the implementation of a delayed least mean square adaptive filter. For accomplishing lower adaptation delay and delay power efficient architecture, we utilize a novel partial product generator and propose a technique for advanced adjusted pipelining over the time consuming combinational blocks of the structure. From synthesis comes about, we find that the proposed configuration offers less delay and power product

Additionally, we have proposed the Kogge Stone adder is a parallel prefix shape carry look-ahead adder. The Kogge Stone adder takes more area to implement than the Brent Kung adder, yet has a lower fan-out at each stage.

Keywords: adaptive filter, least mean square(LMS) algorithms, ripple carry adder, kogge stone adder

I INTRODUCTION

The least mean square (LMS) adaptive filter is the most famous and most generally utilized adaptive filter, in view of its simplicity as well as in view of its satisfactory convergence performance. The direct-form LMS adaptive filter includes a long critical path because of an inner-product computation to acquire the filter output. The critical path is required to be diminished by pipelined implementation when it transcend the desired sample period. Since the normal LMS algorithm does not bolster pipelined usage on account of its recursive direct, it is changed as per a form called the delayed LMS (DLMS) algorithm, which allow pipelined usage of the filter.

A great deal of work has been done to implement the DLMS algorithm in systolic designs toincreases the maximum usable frequency in any case, they include an adaptation delay of ~N cycles for filter length N, which is high for extensive order filter. Since the convergence performance reduces significantly for a large order delay. A transpose-form LMS adaptive filter is recommended in High speed FPGA-based implementations of delayed LMS filters, where the filter output at any moment depends on the delayed version of weights and the number of delays in An effective systolic architecture for the DLMS adaptive filter and its applications weights change from 1 to N. Van and Feng have proposed systolic architecture, where they have utilized generally large processing elements (PEs) for accomplishing a lower adaptation delay with the critical path of one MAC operation.

Ting et al. have proposed a fine-grained pipelined design to constrain the critical path to the maximum of one addition time, which support high sampling frequency, however includes a considerable measure of area overhead to pipelining furthermore, higher power utilization than in , because of its huge number of pipeline latches. Further exertion has been made by Meherand Maheshwari to diminish the number of adaptation delays. Meher and Park have proposed a 2-bit multiplication cell, and utilized that with a capable adder tree for pipelined inner-product computation to keep the critical path and silicon area without expanding the number of adaptation delays.

The current work on the DLMS adaptive filter having large adaptation delay, in spite of the fact that they specifically influence the convergence performance, especially because of the recursive behavior of the LMS algorithm. Consequently, large adaptation delay issues are with the energy utilization. The proposed design is observed to be moreproficient as far as the power-delay product (PDP) contrasted with the existing structures.

In Section II, we survey the DLMS algorithm, and in Section III, we describe the proposed optimized architecture for its usage. Section IV discuss simulation studies of the convergence of the algorithm. In Section V, we consider the synthesis of the proposed architecture and comparison with the existingarchitectures. Conclusions are given in Section V

II. REVIEW OF DELAYED LMS ALGORITHM

The weights of LMS adaptive filter throughout the nth iteration are updated by the accompanying conditions [2]:

$Wn+1=w_n+\mu\cdot e_n\cdot x_n$(1a)
Where
$e_n = d_n – y_n$
and
$y_n=w^T_n\cdot x_n$(1b)

where the input vector x_n, and the weight vector w_n at the nth iteration are, separately, given by

$x_n = [x_n, x_{n-1}, . . . , x_{n-N+1}]^T$

$w_n = [w_n(0),w_n(1), . . . ,w_n(N − 1)]^T$

d_n is the desired response, y_n is the filter output, and e_n indicates the error computed amid the nth iteration. μ is the step size, and N is the number of weights utilized as a part of the LMS adaptive filter. On account of pipelined design with m pipeline stages, the error e_n winds up plainly accessible after m cycles, where m is known as the "adaptation delay." The DLMS algorithm consequently utilizes the delayed error e_{n-m}, i.e., the error relating to (n - m)th iteration for updating the present weight rather than the current most error. The weight-update condition of DLMS adaptive filter is given by

$w_{n+1} = w_n+ \mu \cdot e_{n-m} \cdot x_{n-m}$.   (2)

The block diagram of the DLMS adaptive filter is appeared in Fig. 1, where the adaptation delay of m cycles adds up to the delay presented by the entire of adaptation filter structure comprising of finite impulse response (FIR) filtering and the weight-update process. It is appeared in that the adaptation delay of traditional LMS can be decomposed into two parts: one part is the delay presented by the pipeline stages in FIR filtering, and the other part is because of the delay associated with pipelining the weight updated process. In light ofsuch decomposition of delay, the DLMS adaptation delay can be actualizeddecomposition of delay, the DLMS adaptation delay can be actualized by a structure appeared in Fig. 2.Expecting that the delay of computation

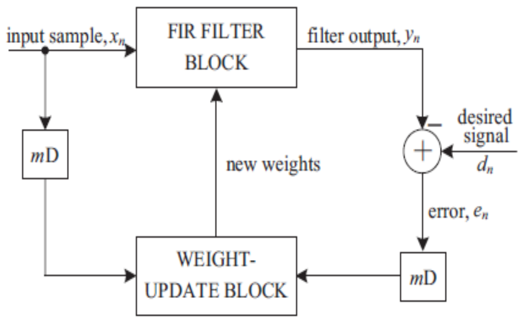


Fig. 1.Construction of the conventional delayed LMS adaptive filter.

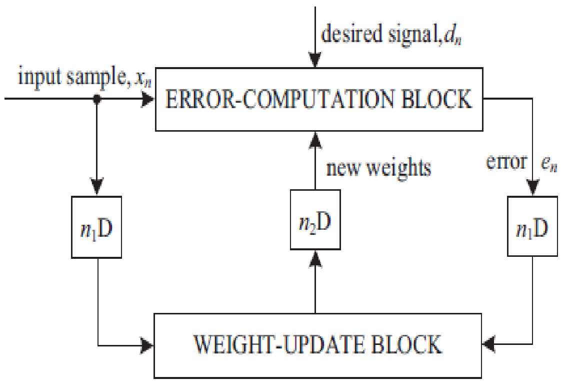


Fig. 2.Construction of the modifieddelayed LMS adaptive filter.

of error is n_1 cycles, the error figured by the structure at the nth cycle is e_{n-n1}, which is utilized with the input samples delayed by n_1 cycles to produce the weight-increase term. The weight-update condition of the modified DLMS algorithm is given by

$w_{n+1}= w_n+\mu\cdot e_{n-n1}\cdot x_{n-n1}$ (2a)

where

$e_{n-n1}=d_{n-n1}-y_{n-n1}$ (2b)

also,

$y_n=w^T_{n-n2}\cdot x_n$ (2c)

We notice that, throughout the weight update, the error with n_1 delays is utilized, while the filtering unit utilizes the weights delayed by cycles. The modified DLMS algorithm decouples calculations of the error-computation and the weight-update block and enablesus to perform ideal pipelining by feed forward cut-set retiming of both these areas independently to limit the number of pipeline stages and adaptation delay. The adaptation filter with various n_1 and n_2 are simulated for a system distinguishing proof issue.

III. PROPOSED ARCHITECTURE

As appeared in Fig. 2, there are two fundamental main blocks in the adaptive filter design: 1) the error computation block and 2) weight update block

In this Section, we talk about the design procedure of the proposed structure to limit the adaptation delay in the error computation block, come after 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 appeared in Fig. 3. It comprises of N number of 2-b partial product generators (PPG) comparing to N multipliers and a group of L/2 binary addertrees, come after by a single shift–add tree. Each sub block is portrayed in detail

1) Structure of PPG: The structure of each PPG is appeared in Fig. 4. It comprises of L/2 number of 2-to-3 decoders and a similar number of AND/OR cells (AOC).1 Each of the 2-to-3 decoders takes a 2-b digit (u_1u_0) as input and produces three output $b_0 = u_0\cdot \bar{u}_1$, $b_1 = \bar{u}_0\cdot u_1$, and $b_2 = u_0 \cdot u_1$, to such an extent that $b_0 = 1$ for (u_1u_0) = 1, $b_1 = 1$ for (u_1u_0) = 2, and $b_2 = 1$ for (u_1u_0) = 3. The decoder output b_0, b_1 and b_2 alongside w, $2w$, and $3w$ are sustained to an AOC, where w, $2w$, and $3w$ are in 2's complement representation and sign-reached out to have (W + 2) bits each. To take care of the indication of the input samples while figuring the partial product be consistent with the most significant digit (MSD), i.e., ($u_{L-1}u_{L-2}$) of the input sample, the AOC (L/2 − 1) is sustained with w, $-2w$, and $-w$ as input since ($u_{L-1}u_{L-2}$) can have four conceivable values 0, 1, −2, and −1.

2) Structure of AOCs: The structure and function of an AOC are represented in Fig. 6. Each AOC comprises of three AND cells and two OR cells. The structure and function of AND cells and OR cells are represented by Fig. 5(b) and (c), individually. Each AND cell takes an n-bit input D and a single bit input b, and comprises of n AND doors. It disperses all the n bits of input D to its n AND gates as one of the inputs. The other input all the n AND gates are encouraged with the single-bit input b. As appeared in Fig. 5(c), each OR cell likewise takes a couple of n-bit input words and has n OR gates. A couple of bits in a similar piece position in B and D is bolstered to the same OR gates. The output of an AOC is w, $2w$, and $3w$ comparing to the decimal values 1, 2, and 3 of the 2-b input (u_1u_0), individually. The decoder alongside the AOC performs a multiplication of input operand w with a 2-b digit (u_1u_0), to such an extent that the PPG of Fig. 5 performs L/2 parallel multiplication of input word w with a 2-b digit to create L/2 partial product of the product word wu.

3) Structure of Adder Tree: Based on, we need have performed the shift-add operation on the partial products of each PPG independently to get the product value and after that included all the N product values to figure the desired inner product. Although, the shift-add operation to acquire the product value expands the word length and subsequently increments the adder size of N − 1 additions of the product values.

To keep away from such increment in word size of the adders, we include all the N partial products of a similar place an incentive from all the N PPGs by one adder tree. All the L/2 partial products created by each of the N PPGs are therefore included by (L/2) binary adder trees. The output of the L/2

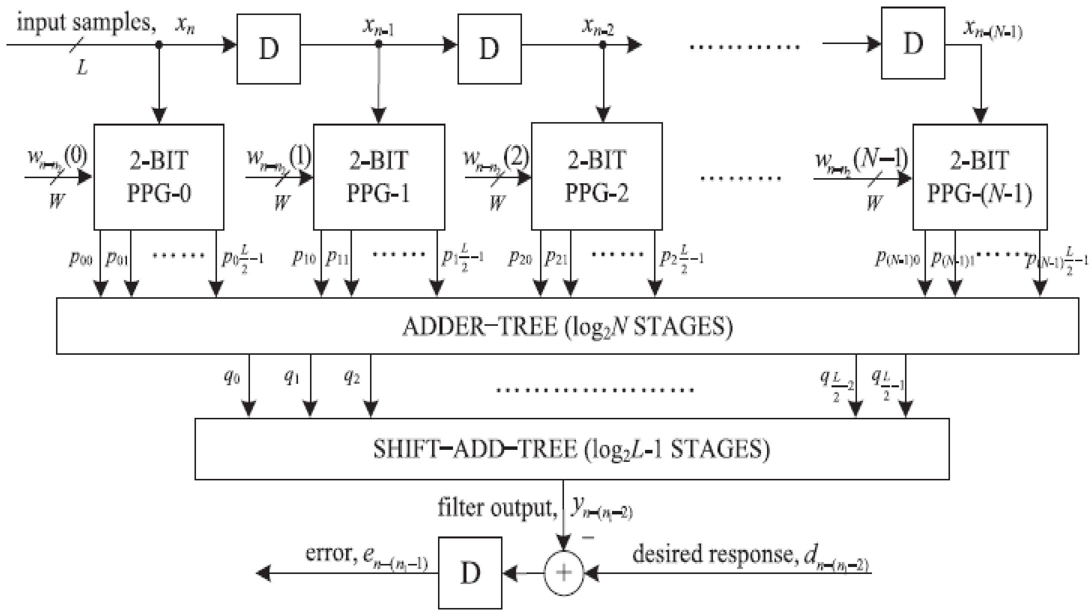

input samples, x_n
L
D
x_{n-1}
x_{n-2}
$x_{n-(N-1)}$
2-BIT PPG-0
2-BIT PPG-1
2-BIT PPG-2
2-BIT PPG-(N-1)
W
ADDER–TREE ($\log_2 N$ STAGES)
SHIFT–ADD–TREE ($\log_2 L$-1 STAGES)
filter output, $y_{n-(n_1-2)}$
error, $e_{n-(n_1-1)}$
desired response, $d_{n-(n_1-2)}$

Fig:3 Proposed structure of the error computation block

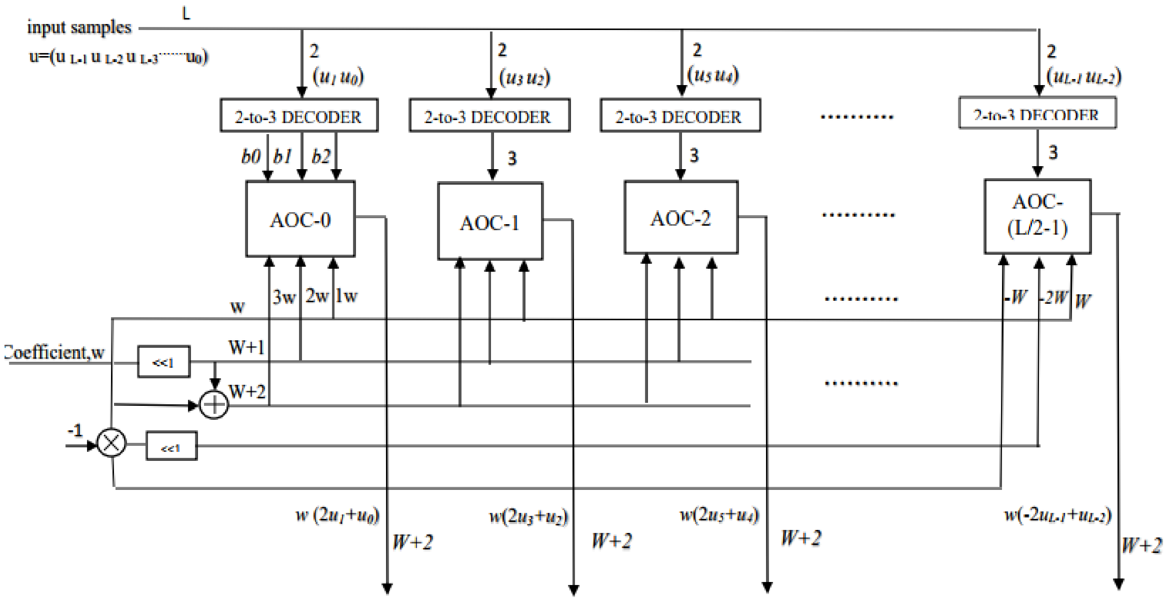
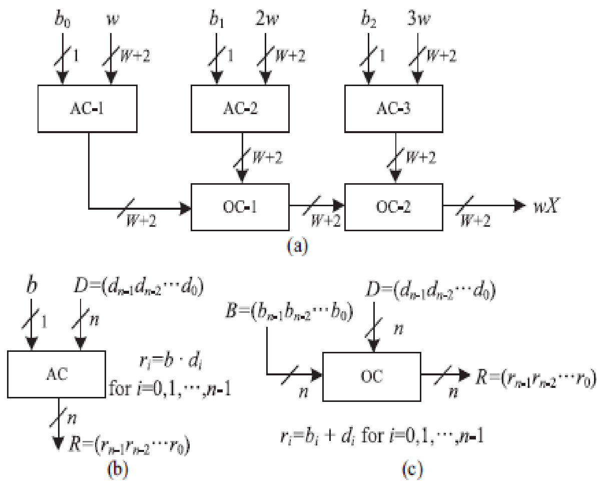


Fig : 4 Proposed structure of the PPG. AOC remains for AND/OR

Adder trees are then included by a shift-add tree as indicated by their place values. Each of the binary adder trees require $\log_2 N$ stages of adders to include N partial products, and the shift–add tree requires $\log_2 L - 1$ stages of adders to include L/2 output of L/2 binary adder trees.

The addition scheme for the error-computation block for a four-tap filter and input word size L = 8 is appeared in Fig. 8. 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 demonstrated all conceivable locations of pipeline latches by dashed lines, to diminish the critical path to one addition time. The last adder in the shift–add tree add to the maximum delay to the critical path. Based on that perception, we have recognized the pipeline latches that don't contribute fundamentally to the critical path and would bar those with no observable increment of the critical path. The location of pipeline latches for filter lengthsN = 8, 16, and 32 and for input size L = 8. The pipelining is performed by a feed forward cut-set retiming of the error computation block.

3A.Structure of adder: The proposed structure for adder tree and shift-add tree both are utilizing adder. In that adder place we are placing Ripple carry adder and Kogge stone adders. Kogge stone adder has high performance and lower delay compared with Ripple carry adder. The Kogge Stone adder is a parallel prefix frame carry look-ahead adder.



AC-1
AC-2
AC-3
OC-1
OC-2
wX
(a)
AC
OC
$r_i = b \cdot d_i$ for $i=0,1,\cdots,n-1$
$R=(r_{n-1}r_{n-2}\cdots r_0)$
$r_i = b_i + d_i$ for $i=0,1,\cdots,n-1$
(b)
(c)

Fig. 5. Structure and function of AND/OR cell. Binary operators .and + in (b) and (c) are implemented utilizing AND as well as OR gates, separately.

the delay input samples xi come after by the addition with the comparing old weight values w_i. All the N multiplications for the MAC operations are performed by N PPGs, come after by N shift add trees. Each of the PPGs produces L/2 partial product relating to the product of the as of late 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 mistake $(\mu\times e)$ is multiplied with all the N delayed input values in the weight-update block, this sub expression can be shared over every one of the multipliers also. The last output of MAC units constitute the desired updated weights to be utilized as inputs to the error-computation block as well as the weight-updated block for the following iteration.

C. Adaptation Delay

As appeared in Fig. 2, the adaptation delay is divided into n_1 and n_2. The error-computation block produces the delayed error by n_1-1 cycles as appeared in Fig. 4, which is boost to the weight-updated block appeared in Fig. 9 in the wake of scaling by μ; at that point the input is delayed by 1 cycle before the PPG to make the total delay presented by FIR filtering be n1.

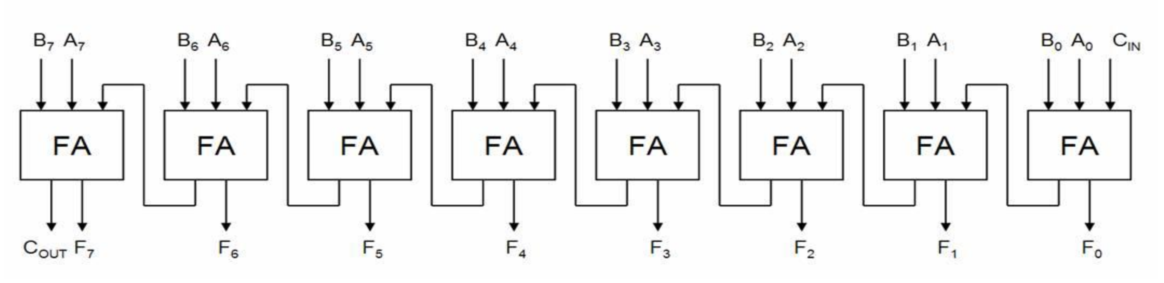


Fig 6: 8bit ripple carry adder

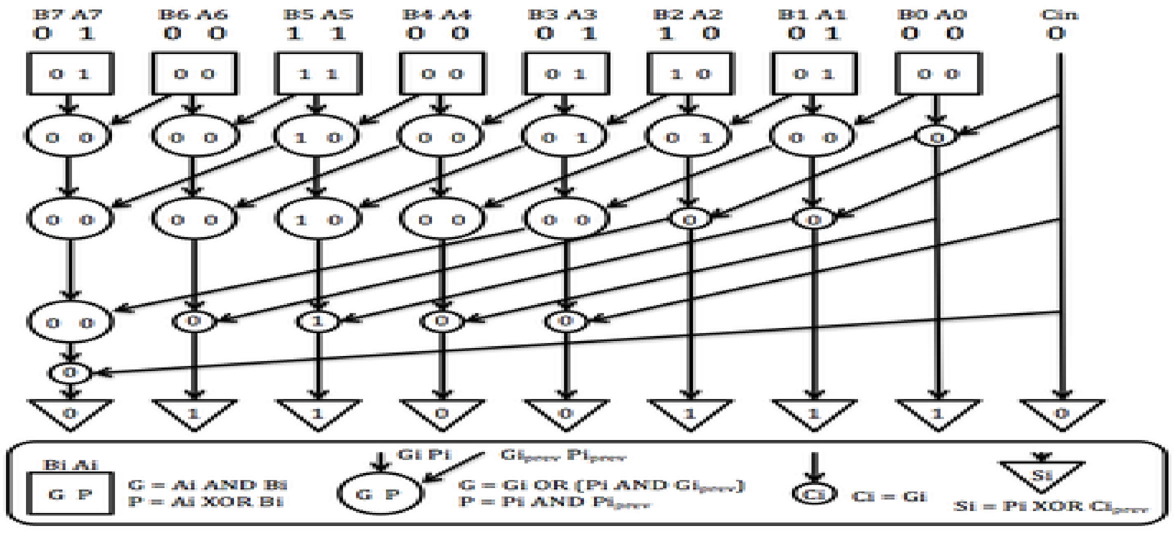


Fig 7: kogge stone parallel prefix adder

B. Pipelined Structure of the Weight-Update Block

The proposed structure for the weight-update block is appeared in Fig.9. It performs N multiply- accumulate operations of the form $(\mu \times e) \times x_i + w_i$ to update N filter weights. The step size μ is taken as a negative power of 2 to understand the multiplication with as of late accessible error just by a shift operation. Each of the MAC units in this way performs the multiplication of the shifted value of error with

In Fig.9, the weight-updated block produce w_{n-1-n2}, and the weights are delayed by n_2+1 cycles. Although, it should be this prompts significant reduction of the adder complexity. noticed that the delay by 1 cycle is because of the latch before the PPG, which is involved into the delay of the error computation block, i.e., n_1. Consequently, the delay produced in the weight updated block moves toward becoming n_2. In the event that the location of pipeline latches are chosen as inTable I, $n_1$moves toward becoming 5,where three latches are in the error computation block, one latches is after the subtraction in Fig. 3, and the other latch is before PPG in Fig. 9. Additionally, n_2 is set to 1 from a latch in the shiftadd tree in the weight-updated block.

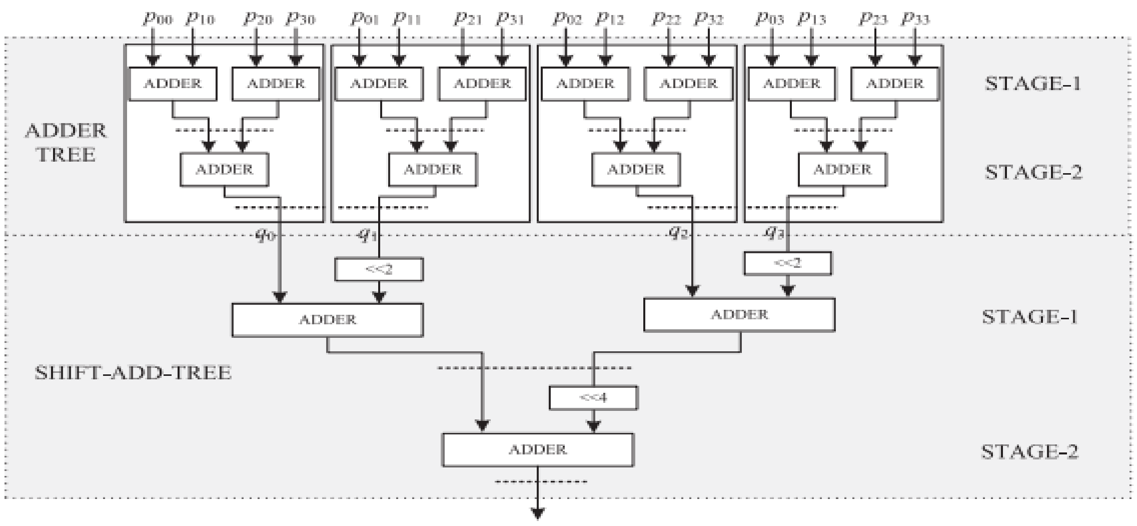

p_{00} p_{10} p_{20} p_{30} p_{01} p_{11} p_{21} p_{31} p_{02} p_{12} p_{22} p_{32} p_{03} p_{13} p_{23} p_{33}
ADDER TREE
ADDER
STAGE-1
STAGE-2
q_0 q_1 q_2 q_3
<<2
<<4
SHIFT-ADD-TREE
STAGE-1
STAGE-2

Fig. 8. Adder construction of the filtering unit for N = 4 and L = 8.

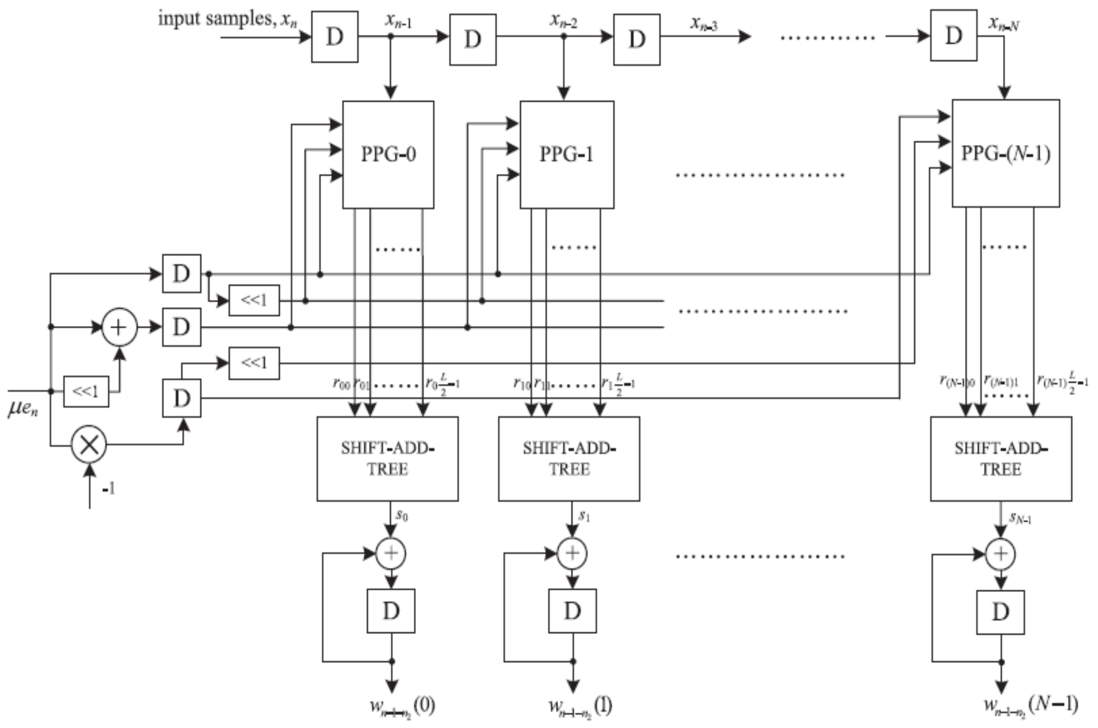


Fig: 9. Proposed structure of the weight-updated block

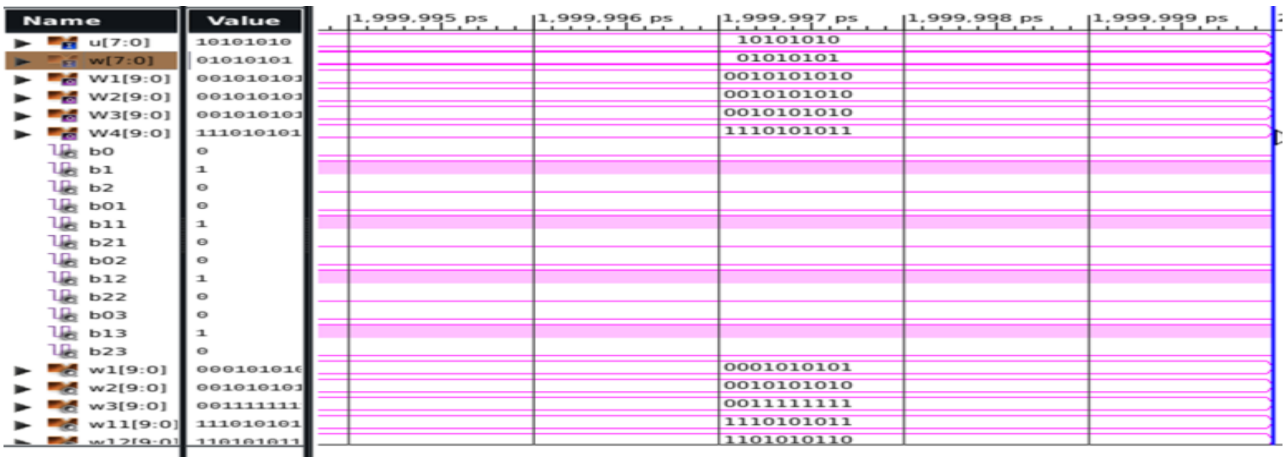
TABLE I

LOCATION PIPELINE LATCHES FOR L=8

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

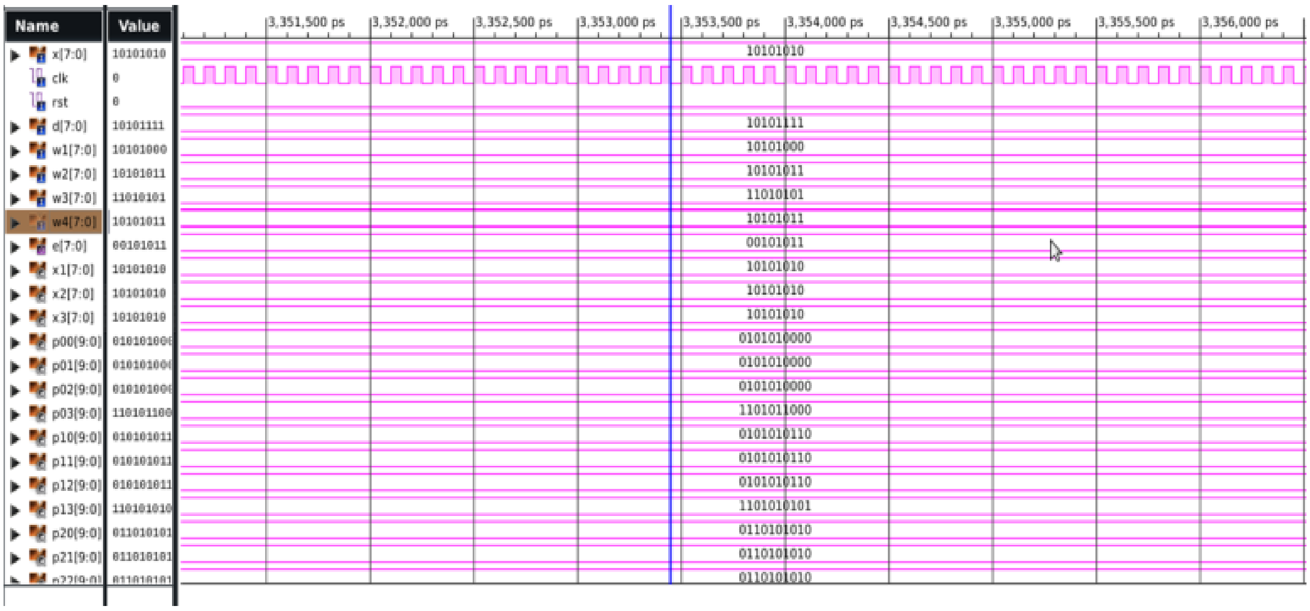
IV. SIMULATION RESULTS

The proposed designs were also implemented on the field programmable gate array (FPGA) platform of Xilinx devices. The number of slices (NOS) and the maximum usable frequency (MUF) using two different devices of Spartan-3A (XC3SD1800A-4FG676) and Virtex-4 (XC4VSX35-10FF668) are listed in Table II. The proposed design-II, after the pruning, offers nearly 11.86% less slice-delay product, which is calculated as the average NOS/MUF, for N = 8, 16, 32, and two devices.



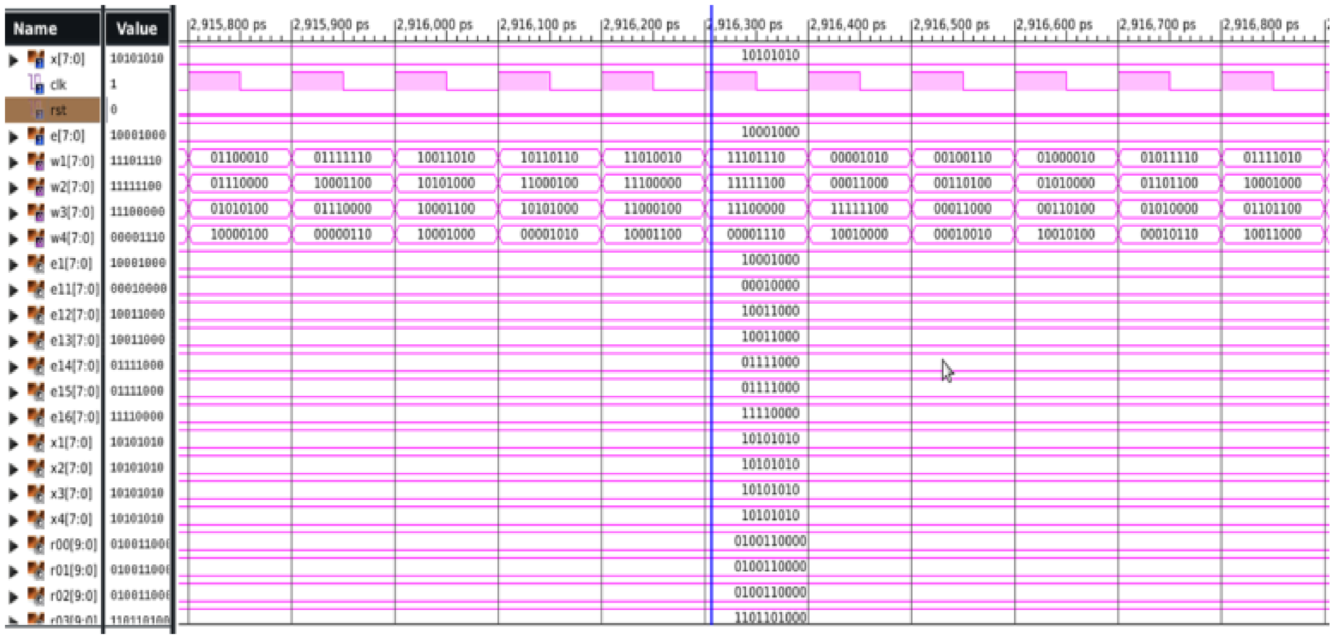
Name	Value
u[7:0]	10101010
w[7:0]	01010101
W1[9:0]	001010101
W2[9:0]	001010101
W3[9:0]	001010101
W4[9:0]	111010101
b0	0
b1	1
b2	0
b01	0
b11	1
b21	0
b02	0
b12	1
b22	0
b03	0
b13	1
b23	0
w1[9:0]	000101010
w2[9:0]	001010101
w3[9:0]	001111111
w11[9:0]	111010101

Simulation waveform:1 for PPG structure



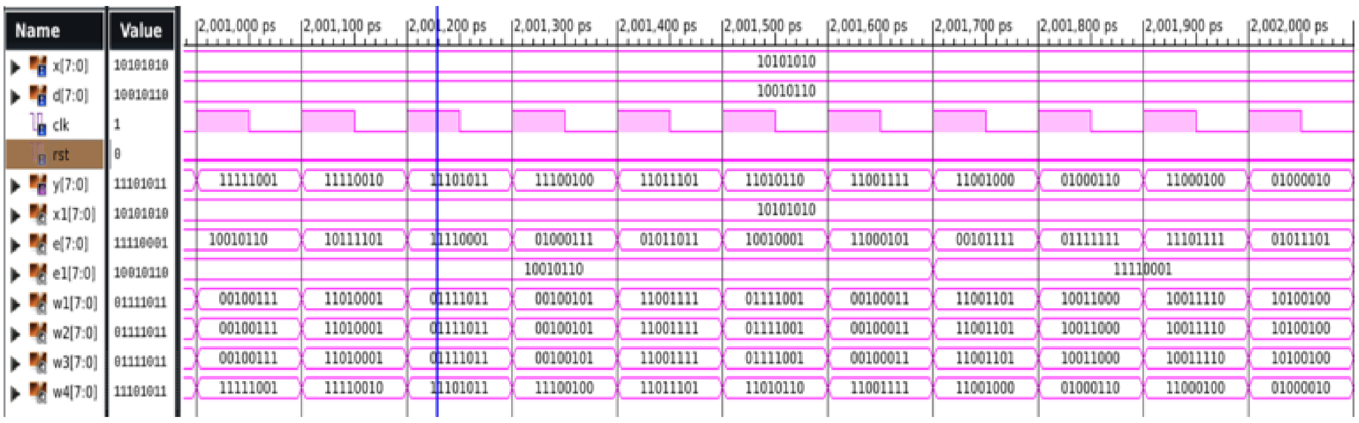
Name	Value
x[7:0]	10101010
clk	0
rst	0
d[7:0]	10101111
w1[7:0]	10101000
w2[7:0]	10101011
w3[7:0]	11010101
w4[7:0]	10101011
e[7:0]	00101011
x1[7:0]	10101010
x2[7:0]	10101010
x3[7:0]	10101010
p00[9:0]	010101000
p01[9:0]	010101000
p02[9:0]	010101000
p03[9:0]	110101100
p10[9:0]	010101011
p11[9:0]	010101011
p12[9:0]	010101011
p13[9:0]	110101010
p20[9:0]	011010101
p21[9:0]	011010101

Simulation waveform:2 for Error-Computation Block structure

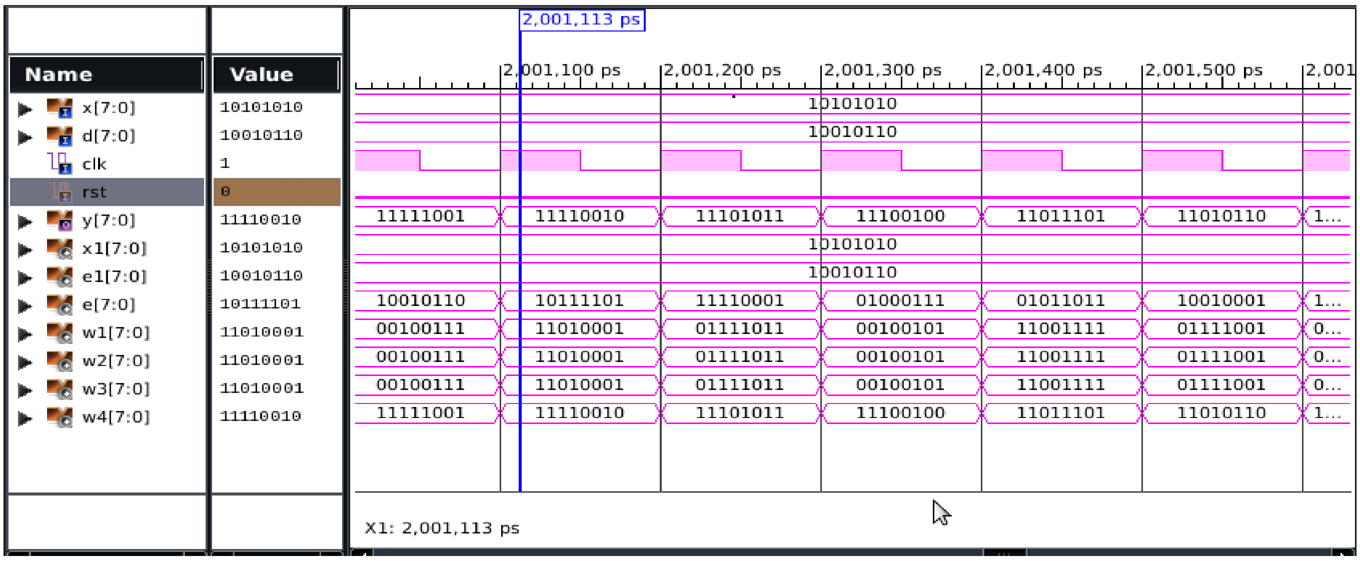


Name	Value
x[7:0]	10101010
clk	1
rst	0
e[7:0]	10001000
w1[7:0]	11101110
w2[7:0]	11111100
w3[7:0]	11100000
w4[7:0]	00001110
e1[7:0]	10001000
e11[7:0]	00010000
e12[7:0]	10011000
e13[7:0]	10011000
e14[7:0]	01111000
e15[7:0]	01111000
e16[7:0]	11110000
x1[7:0]	10101010
x2[7:0]	10101010
x3[7:0]	10101010
x4[7:0]	10101010
r00[9:0]	010011000
r01[9:0]	010011000
r02[9:0]	010011000

Simulation waveform:3 for Weight-Updated Block structure



Simulation waveform :4 for Adaptive filter using ripple carry adder



Simulation waveform :5 for Adaptive filter using kogge stone adder

TABLE II

FPGA IMPLEMENTATION OF PROPOSED DESIGNS FOR L=8 AND N=8

	Ripple carry adder	Kogge stone adder
NOS LUTs	1809	1034
NOS Registers	116	102
Delay	8.200ns	5.706ns

NOS stands for the number of slices

V. Conclusion:

We present a delay-power effective low adaptation delay design of LMS adaptive filter. We utilized a novel PPG for effective design of general multiplication and inner product computation by regular sub expression sharing. In addition, we have proposed kogge stone adder structure for effective addition scheme for inner-product computation to reduce the adaptation delay and to reduce the critical path to help high input sampling rates. Beside this, we proposed a methodology for advanced balanced pipelining over the time consuming blocks of the structure to decrease the adaptation delay and power utilization, too. The proposed structure included the less adaptation delay comparing with the existing structures.

REFERENCES

[1] 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.

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.
[8] Simon hykin, Editor, "adaptive filter theory", International edition, (1996).
[9] [1] B. Widrow and S. D. Stearns, Adaptive Signal Processing. EnglewoodCliffs, NJ, USA: Prentice-Hall, 1985.
[10] S. Haykin and B. Widrow, Least-Mean-Square Adaptive Filters. Hoboken, NJ, USA: Wiley, 2003.

Authors

P. Swapnakumari, Studying M.Tech, Final year in VLSI &ES Specialization, ECEDepartment,VelagapudiRamakrishna SiddharthaEngineeringCollege,Kanuru,Vijayawada-7,AffiliatedtoJawaharlalNehru Technological University Kakinada, Andhra Pradesh.

T. Rajasekhar,AssistantProfessor,ECEDepartment,VelagapudiRamakrishnaSiddhartha Engineering College,Kanuru,Vijayawada-7,AffiliatedtoJawaharlalNehruTechnologicalUniversity – Kakinada, AndhraPradesh.