Speech De-noising DLMS Adaptive filter

S.Indraja 1, T. Vijaya Nirmala 2

1 M. Tech. VLSI, Department of ECE, Anamacharya Institute of Technology, Kadapa, Andhra Pradesh,
Indraja413@gmail.com
2 Assistant Professor, Dept of ECE, Anamacharya Institute of Technology, Kadapa, Andhra Pradesh,
nirmalaais@gmail.com

Abstract—In this paper, we present an proficient architecture for the execution of a deferred least mean square adaptive filter. For achieving lower adaptation-delay and area-delay-power capable execution, we use a novel partial product generator and recommend a strategy for optimized resonable pipelining across the time-consuming combinational blocks of the structure. From synthesis results, we find that the proposed design offers nearly 17% less area-delay product (ADP) and nearly 14% less energy-delay product (EDP) than the best of the obtainable systolic structures, on regular for filter lengths $N = 8,16$, and 32. We suggest an proficient fixed-point completion scheme of the projected architecture and develop the appearance for steady-state error. We show that the steady-state mean squared error obtained from the analytical result matches with the recreation result. Furthermore, we have projected a bit-level pruning of the proposed architecture, which make available nearly 20% saving in ADP and 9% saving in EDP over the proposed structure before pruning lacking visible deprivation of steady-state-error performance.

Index Terms—Adaptive filters, circuit optimization, fixed-point arithmetic, least mean square (LMS) algorithms.

I. INTRODUCTION

THE LEAST MEAN SQUARE (LMS) adaptive filter is the most popular and most extensively used adaptive filter, not only because of its difficulty but also because of its reasonable divergence performance [1], [2]. The direct-form LMS adaptive filter occupy a long significant path due to an inner-product calculation to attain the filter output. The critical path is necessary to be condensed by pipelined execution when it exceeds the preferred sample period. Since the conservative LMS algorithm does not sustain pipelined execution because of its recursive behavior, it is modified to a form called the delayed LMS (DLMS) algorithm [3]–[5], which allows pipelined execution of the filter. A lot of work has been done to execute the DLMS algorithm in systolic architectures to increase the maximum working frequency [3], [6], [7] but, they occupy an variation delay of $\sim N$ cycles for filter length $N$, which is quite high for largeorder filters. Since the junction presentation degrades significantly for a large revision delay, Visvanathan et al. [8] have proposed a adapted systolic architecture to decrease the revision delay. A transpose-form LMS adaptive filter is recommended in [9], where the filter output at any immediate depends on the deferred versions of weights and the number of delays in weights varies from 1 to $N$. Van and Feng [10] have planned a systolic architecture, where they have used comparatively large processing elements (PEs) for achieving a lower edition delay with the critical path of one MAC operation. Ting et al. [11] have proposed a fine-grained pipelined design to limit the critical path to the greatest of one addition time, which sustains high sampling frequency, but occupy a lot of area transparency for pipelining and higher power consumption than in [10], due to its large number of pipeline latches. Further effort has been made by Meher and Maheshwari [12] to reduce the number of adaptation delays. Meher and Park have proposed a 2-bit multiplication cell, and used that with an proficient adder tree for pipelined inner-product calculation to minimize the critical path and silicon area lacking growing the number of variation delays [13], [14]. The existing work on the DLMS adaptive filter does not discuss the fixed-point execution issues, e.g., location of radix point, choice of word length, and quantization at different stages of calculation, even though they honestly affect the convergence performance, mainly due to the recursive behavior of the LMS algorithm. Consequently fixed-point execution issues are given sufficient importance in this paper. Besides, we present here the optimization of our previously reported design [13], [14] to reduce the number of pipeline delays along with the area, sampling period and energy conservation. The proposed design is found to be more resourceful in terms of the power-delay product (PDP) and energy-delay product (EDP) compared to the existing structures. In the 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 deliberations and reproduction studies of the convergence of the algorithm. In Section V, we discuss the synthesis of the proposed architecture and assessment with the existing architectures. Conclusions are given in Section VI.

II. REVIEW OF DELAYED LMS ALGORITHM

The weights of LMS adaptive filter through the \( n \)th iteration are updated according to the following equations [2]: where the input vector \( x_n \), and the weight vector \( w_n \) at the \( n \)th iteration are, respectively, given by

\[
 w_{n+1} = w_n + \mu \cdot e_n \cdot x_n \tag{1a}
\]

\[
 e_n = d_n - y_n \quad y_n = w_n^T \cdot x_n \tag{1b}
\]

where the input vector \( x_n \), and the weight vector \( w_n \) at the \( n \)th iteration are respectively, given by

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

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

\( d_n \) is the desired response, \( y_n \) is the filter output, and \( e_n \) denotes the error computed during the \( n \)th iteration. \( \mu \) 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 \( e_n \) becomes available after \( m \) cycles, where \( m \) is called the “adaptation delay.” The DLMS algorithm consequently uses the deferred error \( e_{n-m} \), i.e., the error equivalent to \((n-m)\)th iteration for updating the current weight as an alternative of the recent-most error. The weight-update equation of DLMS adaptive filter is given by

\[
 w_{n+1} = w_n + \mu \cdot e_{n-m} \cdot x_{n-m} \tag{2}
\]

The block diagram of the DLMS adaptive filter is shown in Fig. 1, where the variation 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 decaying 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 occupied in pipelining the weight-update method. Based on such a decomposition of delay, the DLMS adaptive filter can be execute by a structure shown in Fig. 2.

Assuming that the latency of calculation of error is \( n_1 \) cycles, the error computed by the structure at the \( n \)th cycle is \( e_{n-n_1} \), which is used with the input samples deferred by \( n_1 \) cycles to produce the weight-increment term.

\[
 w_{n+1} = w_n + \mu \cdot e_{n-n_1} \cdot x_{n-n_1} \tag{3a}
\]

\[
 e_{n-n_1} = d_{n-n_1} - y_{n-n_1} \tag{3b}
\]

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

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

Fig. 3. Convergence performance of system identification with LMS and modified DLMS adaptive filters.

update equation of the modified DLMS algorithm is given by
And
\[ y = \mathbf{w}^T \mathbf{x} \quad \text{(3c)} \]
We notice that, during the weight update, the error with \( n_1 \) delays is used, while the filtering unit uses the weights delayed by \( n_2 \) cycles. The modified DLMS algorithm decouples calculation of the error computation block and the weight-update block and allows us to execute optimal pipelining by feedforward cut-set re timing of both these sections separately to minimize the number of pipeline stages and adaptation delay. The adaptive filters with different \( n_1 \) and \( n_2 \) are simulated for a system identification problem. The 10-tap band pass filter with impulse response,

\[ h_n = \frac{\sin(w_H(n - 4.5))}{\pi(n - 4.5)} - \frac{\sin(w_L(n - 4.5))}{\pi(n - 4.5)} \]

for \( n = 0, 1, 2, \ldots, 9 \), otherwise \( h_n = 0 \) \((4)\)
is used as the unknown system as in [10]. \( w_H \) and \( w_L \) correspond to the high and low cutoff frequencies of the passband and are set to \( w_H = 0.7\pi \) and \( w_L = 0.3\pi \), correspondingly. The step size \( \mu \) is set to 0.4. A 16-tap adaptive filter make out the unknown system with Gaussian random input \( x_n \) of zero mean and unit variance. In all cases, production of known system are of unity power, and infected with white Gaussian noise of −70 dB strength. Fig. 3 shows the learning curve of MSE of the error signal \( e_n \) by averaging 20 runs for the conservation LMS adaptive filter \((n_1 = 0, \ n_2 = 0)\) and DLMS adaptive filters with \((n_1 = 5, \ n_2 = 1)\) and \((n_1 = 7, \ n_2 = 2)\). It can be seen that, as the total number of delays increases, the convergence is slowed down, while the steady-state MSE remains almost the same in all cases. In this example, the MSE difference between the cases \((n_1 = 5, \ n_2 = 1)\) and \((n_1 = 7, \ n_2 = 2)\) after 2000 iterations is less than 1 dB, on average.

III. 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 Arena, we into the chunk gadget of the calculated 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) equivalent to \( N \) multipliers and a cluster of \( L/2 \) binary adder trees, followed by a single shift–add tree. Each subblock 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). Each of the 2-to-3 decoders takes a 2-b digit \((u_1u_0d)\) as input and produces three outputs \(b_0 = u_0 \cdot u_1\), \(b_1 = u_0 \cdot \bar{u}_1\), and \(b_2 = u_0 \cdot u_1\), such that \(b_0 = 1\) for \((u_1u_0d) = 1, b_1 = 1\) for \((u_1u_0d) = 2, \) and \(b_2 = 1\) for \((u_1u_0d) = 3\). The decoder output \(b_0, b_1\) and \(b_2\) along with \(w, 2w,\) and \(3w\) are fed to an AOC, where \(w, 2w,\) and \(3w\) are in \(2^n\)’s go together illustration and sign-extended to have \((W + 2)\) bits each. To take care of the sign of the input samples while computing the partial product equivalent to the most important digit (MSD), i.e., \((uL_{-1}uL_{-2})\) of the input sample, the AOC \((L/2 - 1)\) is fed with \(w, -2w,\) and \(-w\) as input since \((uL_{-1}uL_{-2})\) 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), correspondingly. 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 correspondingly 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 \(3w\) matching to the decimal values 1, 2, and 3 of the 2-b input \((u_1u_0d)\), correspondingly. The decoder along with the AOC performs a multiplication of input operand \(w\) with a 2-bit digit \((u_1u_0d)\), such that the PPG of Fig. 5 performs \(L/2\) parallel multiplications of input word \(w\) with a 2-bit digit to produce \(L/2\) partial products of the product word \(wu\).

![Diagram](https://via.placeholder.com/150)

**Fig. 8. Structure and function of AND/OR cell. Binary operators \(\cdot\) and \(+\) in (b) and (c) are implemented using AND and OR gates, respectively**

3) Structure of Adder Tree: Broadly, we requirement take a crack at executed the shift-add do on the jaundiced merchandise of each PPG separately to earn the product value and then added all the \(N\) product values to compute the desired inner product. In any way, the shift-add disassemble 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 augment in word size of the adders, we add all the \(N\) fractional products of the same place value from all the \(N\) PPGs by one adder tree. All the \(L/2\) partial products produced by each of the \(N\) PPGs are thus added by \((L/2)\) binary adders. 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 necessitate \(\log_2 N\) stages of adders to add \(N\) partial product, and the shift–add tree necessitate \(\log_2 L - 1\) stages of adders to add \(L/2\) output of \(L/2\) binary adders.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 require four binary adder trees of two stages each and a two-stage shift–add tree. In this figure, we have shown all probable locations of pipeline latches by dashed lines, to decrease the critical path to one addition time. If we introduce pipeline latches after every addition, it would necessitate \(L(N - 1)/2 + L/2 - 1\) latches in \(\log_2 (N + \log_2 L - 1)\) stages, which would lead to a high adaptation delay and introduce a large transparency of area and power consumption for large values of \(N\) and \(L\). On the other hand, some of those pipeline latches are surplus in the sense that they are not necessary to continue a critical path of one addition time. The final adder in the shift–add tree supply to the maximum delay to the critical path. Based on that opinion, we have recognized the pipeline latches that do not supply considerably to the critical path and could prohibit those without any conspicuous 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 feedforward cut-set retiming of the error-computation block [15].

<table>
<thead>
<tr>
<th>(N)</th>
<th>Error-Computation Block</th>
<th>Weight-Update Block</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>Stage 2</td>
<td>Stage-1 tree</td>
</tr>
<tr>
<td>16</td>
<td>Stage 3</td>
<td>Stage-1</td>
</tr>
<tr>
<td>32</td>
<td>Stage 3</td>
<td>Stage-1 and 2</td>
</tr>
</tbody>
</table>

**Table I**
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 \((\mu \times e) \times x_i + w_i\) to update \( N \) filter weights. The step size \( \mu \) is taken as a negative power of 2 to appreciate the multiplication with freshly obtainable error only by a shift operation. Each of the MAC units consequently performs the multiplication of the shifted value of error with the delayed input samples \( x_i \) followed by the additions with the matching 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 freshly shifted error value \( \mu \times e \) with \( L/2 \), the number of 2-b digits of the input word \( x_i \), where the subexpression \( 3\mu \times e \) is shared within the multiplier. Since the scaled error \((\mu \times e)\) is multiplied with all the \( N \) behind time input serenity in the weight-update block, this subexpression can be shared across all the multipliers as well. This leads to substantial reduction of the adder complexity. The categorical outputs of MAC accessories form the call for updated weights to be used as inputs to the error-computation block as well as the weight-update block for the next iteration.

C. Adaptation Delay

As shown in Fig. 2, the quarters interrupt is decomposed into \( n_1 \) and \( n_2 \). The error-computation field generates the delayed error by \( n_1 - 1 \) cycles as shown in Fig. 4, which is fed to the weight-update tract shown in Fig. 8 check into scaling by \( \mu \); fit the input is delayed by 1 recur before the PPG to make the total under legal restraint introduced by FIR filtering be \( n_1 \). In Fig. 8, the weight-update block generates \( w_{n-1} - n_2 \), and the weights are delayed by \( n_2 \) cycles. But, it forced to be superb depart 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., \( n_1 \). Narrative, the delay generated in the weight-update block becomes \( n_2 \). If the locations of plain latches are deflected as in Board I, \( n_1 \) becomes 5, where three latches are in the error-computation block, one latch is after the subtraction in Fig. 4, and the stand-in latch is before PPG in Fig. 8. Above, \( n_2 \) traditional to 1 from a latch in the shift-add tree in the weight-update block.

Fig. 9. Adder-structure of the filtering unit for \( N = 4 \) and \( L = 8 \).

Fig. 10. Proposed structure of the weight-update block.

Fig. 11. Result for the weight update block diagram
IV. FIXED-POINT IMPLEMENTATION, OPTIMIZATION, SIMULATION, AND ANALYSIS

In this courtyard, we quarrel the fixed-point implementation and optimization of the proposed DLMS adaptive filter. A bit-level pruning of the adder secret agent is other than proposed to reduce the hardware complexity without noticeable degradation of steadystate MSE.

![Fig. 12: Fixed-point representation of a binary number (Xi: integer wordlength; Xf: fractional word-length).](image)

TABLE II

<table>
<thead>
<tr>
<th>Signal Name</th>
<th>Fixed-Point Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>(L, Li)</td>
</tr>
<tr>
<td>w</td>
<td>(W, W1)</td>
</tr>
<tr>
<td>p</td>
<td>(W + 2, W1 + 2)</td>
</tr>
<tr>
<td>q</td>
<td>(W + 2 + \log_2 N, W1 + 2 + \log_2 N)</td>
</tr>
<tr>
<td>y, d, e</td>
<td>(W, W1 + L1 + \log_2 N)</td>
</tr>
<tr>
<td>\mu e</td>
<td>(W, W1)</td>
</tr>
<tr>
<td>r</td>
<td>(W + 2, W1 + 2)</td>
</tr>
<tr>
<td>s</td>
<td>(W, W1)</td>
</tr>
</tbody>
</table>

x, w, p, q, y, d, and e can be found in the error-computation block of Fig. 4. \( \mu, r, \) and \( s \) are defined in the weight-update block in Fig. 8. It is to be noted that all the subscripts and time indices of signals are omitted for simplicity of notation.

A. Fixed-Point Design Considerations

For fixed-point mastery, the variant of report almost imperceptibly a rather and radix points for input samples, weights, and internal signals demand to be decided. Fig. 9 shows the fixed-point asseveration of a binary number. Authorize \((X, X_i)\) be a fixed-point representation of a binary number where \(X\) is the notice secure fix and \(X_i\) is the integer length. The word length and speak of radix point of \(x\) and \(w\) in Fig. 4 need to be unfluctuating by the ironmongery novelist taking the design constraints, such as desired accuracy and hardware complexity, into consideration. Uppity \((L, Li)\) and \((W, Wi)\), respectively, as the representations of input signals and filter weights, all other signals in Figs. 4 and 8 underpinning be decided as shown in Table II. The wide awake \(p_i j\), which is the output of PPG block (shown in Fig. 4), has at most three times the value of input coefficients. Commensurate with explain, we can amplify connect more bits to the word length and to the integer length of the coefficients to avoid overflow. The output of each stage in the adder tree in Fig. 7 is one bit more than the size of input signals, so that the fixed-point representation of the output of the adder tree with \(\log_2 N\) stages becomes \((W + \log_2 N + 2, W1 + \log_2 N + 2)\). Accordingly, the output of the shift—add tree would be of the form \((W+L+\log_2 N, W1+Li+\log_2 N)\), assuming that no truncation of any least significant bits (LSB) is performed in the adder tree or the shift—add tree. At any rate, the all of a add up to of fabric of the procure of the shift—add tree is designed to have W claptap. The master tail W bits bid to be retained outside of \((W, L, \log_2 N)\) bits, which results in the fixed-point asseveration \((W, Wi \log_2 N)\) for \(y\), as shown in Table II. Take into account the representation of the needed alarm deuterium oxide be the same as \(y\), even though its quantization is usually given as the input. For this end, the drug scaling/sign extension and truncation/zero padding are required. Notwithstanding the LMS algorithm performs savoir faire thus range \(y\) has the same sign as \(d\), the error signal \(e\) can also be set to have the same representation as \(y\) without overflow after the subtraction. It is shown in [4] mosey the concentrate of an N-tap DLMS adaptive filter with \(n1\) adaptation delay will be ensured if where \(\sigma x\) is the average power of input samples. Furthermore, if the value of \(\mu\) is defined as \((\text{power of} 2)^{2-n}\), where \(n \leq Wi+Li+\log_2 N\), the multiplication with \(\mu\) is equivalent to the change of location of the radix point. Since the multiplication with \(\mu\) does not need any arithmetic operation, it does not introduce any truncation error. If we need to use a smaller step size, i.e., \(n > Wi+Li+\log_2 N\), some of the LSBs of \(en\) need to be truncated. If we assume that \(n = Li + \log_2 N\), i.e., \(\mu = 2^{-(Li+\log_2 N)}\), as in Table II, the representation of \(\mu en\) should be \((W, W1)\) without any truncation. The weight increment term \(s\) (shown in Fig. 8), which is equivalent to \(\mu en X1\), is required to have fixed-point representation \((W + L, W1 + L1)\).

\[
\mu en = \frac{\sigma x(N - 2) + 2\sigma Y - 2\sigma Y^2}{(\sigma x^2)N - 2)} \quad (5)
\]

How in the world, merely \(Wi\) MSBs in the in compliance of the shift—add assign of the estimate—update circuit are to be retained, while the rest of the more significant bits of MSBs need to be discarded. This is in contract everywhere the assumptions rove, as the weights cluster nearing the optimal value, the weight increment terms become smaller, and the MSB end of error term contains more number of zeros. Additionally, in our deny stuff up, \(L - Li\) LSBs of weight increment terms are truncated so that the terms have the same fixed-point representation as the weight values. We apart from resign oneself to that no overflow occurs during the addition. for the weight update. Way , the announcement stilly of the
weights should be increased at every iteration, which is not desirable. The belief is sanctioned pro the weight increment terms are small when the weights are converged. Also when overflow occurs during the training period, the weight updating is not appropriate and will lead to additional iterations to reach convergence. Accordingly, the updated weight can be computed in truncated form \((W, Wi)\) and fed into the error computation block.

**B. Computer Simulation of the Proposed DLMS Filter**

The young fixed-strive for DLMS adaptive pass through is used for Protocol identification used in Section II. \(\mu\) is regular to 0.5, 0.25, and 0.125 for eliminate to influential sum total 8, 16, and 32, respectively, such depart the multiplication with \(\mu\) does not require any additional circuits. For the fixed-aspiration la-di-da orlah-di-dah show, the proclamation forge and origination point of the input and coefficient are familiar to \(L = 16, Li = 2, W = 16, Wi = 0\), and the Gaussian random input \(x_n\) of zero mean and unit variance is scaled down to fit in with the representation of \((16, 2)\). The fixed-point materials detract from of all the other signals are obtained from Table II. Again experience genuflect is averaged over 50 runs to obtain a clean curve. The bodiless prohibit was coded in Trouble-free dedicate Criterion criteria As fixed-point library for different orders of the band-pass riddle, turn is, \(N = 8, N = 16, N = 32\). The coequal delegate behaviors are obtained, as shown in Fig. 10. It is unforeseen that, as the filter play increases, not only the convergence becomes slower, but the steady-state MSE also increases.

**C. Steady-State Error Estimation**

In this section, the MSE of output of the projected DLMS adaptive filter due to the fixed-point quantization is analyzed. Based on the models initiate in [16] and [17], the MSE of output in the stable state is resulting in terms of parameters planned in Table II. Let us denote the primed symbols as the condensed quantities due to the fixed-point depiction, so that the input and the preferred signals can be written as

\[
X'_n = X_n + \alpha_n \quad (6)
\]

\[
d'_n = d_n + \beta_n \quad (7)
\]

where \(\alpha_n\) and \(\beta_n\) are input quantization noise vector and quantization noise of beloved signal, correspondingly. The weight vector can be written as

\[
W'_n = W_n + \rho_n \quad (8)
\]

where \(\rho_n\) is the error vector of current weights due to the limited accuracy. The output signal \(y'_n\) and weight-update equation can therefore be modified, correspondingly, to the forms

where \(\eta_n\) and \(\gamma_n\) are the errors due to the truncation of output from the shift–add tree in the error computation block and weight-update block, correspondingly. The steady-state MSE in the fixed-point representation can be expressed as

\[
\begin{aligned}
Y'_n &= W'_n^T X'_n + \eta_n \\
W'_{n+1} &= W'_n + \mu \beta x_n + \gamma_n
\end{aligned} \quad (9)
\]

\[
E[|d_n - y'_n|^2] = E[|\alpha_n|^2] + E[|\alpha_n^T w_n|^2] + E[|\eta_n|^2] + E[|\rho_n x_n|^2] \quad (11)
\]

where \(E[|\cdot|\cdot]\) is the operator for mathematical expectation, and the terms \(\alpha_n, \beta x_n, \eta_n,\) and \(\rho_n x_n\) are assumed to be uncorrelated.

The first term \(E[\alpha_n^T w_n]\), where \(\alpha_n = d_n - y_n\), is the excess MSE from inestimable accuracy calculation, while the other three terms are due to finite-precision arithmetic.

<table>
<thead>
<tr>
<th>TABLE III</th>
<th>ESTIMATED AND SIMULATED STEADY-STATE MSES OF THE FIXED-POINT DLMS ADAPTIVE FILTER ((L = W = 16))</th>
</tr>
</thead>
<tbody>
<tr>
<td>Filter Length</td>
<td>Step Size ((\mu))</td>
</tr>
<tr>
<td>(N = 8)</td>
<td>(2^{\frac{1}{2}})</td>
</tr>
<tr>
<td>(N = 16)</td>
<td>(2^{\frac{2}{2}})</td>
</tr>
<tr>
<td>(N = 32)</td>
<td>(2^{\frac{3}{2}})</td>
</tr>
</tbody>
</table>

The second term can be calculated as

\[
E[|\alpha_n^T w_n|^2] = |w^*|^2 (m^2_{\alpha_n} + \sigma^2_{\alpha_n}) \quad (12)
\]

where \(w^*\) is the optimal Wiener vector, and \(m_{\alpha_n}\) and \(\sigma^2_{\alpha_n}\) are defined as the mean and variance of \(\alpha_n\) when \(x_n\) is condensed to the fixed-point type of \((L, Li)\), as listed in Table II. \(\alpha_n\) can be modeled as a uniform allocation with following mean and variance:

\[
\begin{aligned}
m_{\alpha_n} &= 2^{-(L-L_i)}/2 \\
\sigma^2_{\alpha_n} &= 2^{-2(L-L_i)/12}.
\end{aligned} \quad (13a, b)
\]

For the calculation of the third term \(E[\eta_n^2]\) in (11), we have used the fact that the output from shift–add tree in the error computation block is of the type \((W, Wi + Li + \log_2 N)\) after the final truncation. Therefore

\[
E[|\eta_n|^2] = m^2_{\eta_n} + \sigma^2_{\eta_n} \quad (14)
\]

Where
\[ m^2_{\eta_n} = 2^{-2(W - W_i)/2} \]  
\[ \sigma^2_{\eta_n} = 2^{-2(W - W_i)/12}. \]  
(15a)  
(15b)

The last term \( E[\rho^T_n \eta_n]^2 \) in (11) can be obtained by using the beginning projected in [17] as

\[ E[\rho^T_n \eta_n]^2 = m^2_{\eta_n} \sum_k \left( R^{-1}_{ki} \right) + \frac{N(\sigma^2_{\eta_n} - m^2_{\eta_n})}{\mu^2} \]  
(16)

where \( R_{ki} \) represent the \((k, i)\)th entry of the matrix \( E[\mathbf{x} \mathbf{x}^T n] \). For the first weight update in (10), the first operation is to multiply \( \mathbf{e}_n \) with \( \mu \), which is comparable to moving only the location of the radix point and, therefore, does not introduce any truncation error. The truncation after multiplication of \( \mathbf{e}_n \) with \( \mathbf{x}_n \) is only required to be measured in order to appraise \( \gamma_n \). Then, we have

\[ m^2_{\gamma_n} = 2^{-2(W - W_i)/2} \]  
\[ \sigma^2_{\gamma_n} = 2^{-2(W - W_i)/12}. \]  
(17a)  
(17b)

For a large \( \mu \), the truncation error \( \eta_n \) from the error computation block becomes the dominant error source, and (11) can be approximated as \( E[\eta_n]^2 \). The MSE values are estimated from analytical expressions as well as from the simulation results by averaging over 50 experiments. Table III shows that the steady-state MSE computed from analytical expression matches with that of simulation of the proposed architecture for different values of \( N \) and \( \mu \).

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. \( \eta_n \) defined in (9) increases if we prune the adder tree, and the worst case error is caused when all the truncated bits are 1. For the calculation of the sum of truncated values in the worst case, let us denote \( k1 \) as the bit location of MSB of truncated bits and \( Nk2 \) as the number of rows that are affected by the truncation. In the example of Fig. 11, \( k1 \) and \( k2 \) are set to 5 and 3, respectively, since the bit positions from 0 to 5 are truncated and a total of 12 rows are affected by the truncation for \( N = 4 \). Also, \( k2 \) can be derived using \( k1 \) as

\[ k_2 = \lfloor \frac{k_1}{2} \rfloor + 1 \text{ for } k_1 < \frac{W}{2}, \text{ otherwise } k_2 = \frac{W}{2} \]  
(18)

since allowing for regarding the come up to b become of truncated boloney is economy by 2 for every group of N rows as shown in Fig. 11. Make use of \( k1 \), \( k2 \), and \( N \), the sum of truncated values for the worst case can be formulated as

\[ b_{\text{worst}} = N \sum_{j=0}^{k_1-1} \sum_{i=2j}^{k_1-1} 2^i = N \left( k_2 2^{k_1} + 1 - \frac{1}{3}(4^{k_2} - 1) \right). \]  
(19)

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

D. Adder-Tree Optimization

The adder mill and shift–add machinery for the chronicle of \( y_n \) can be pruned for further optimization of area, delay, and power complexity. To show the so-called pruning optimization of adder herb and shift–add bush for the description notice of filter procure, we take a simple covering of filter length \( N = 4 \), considering the word lengths \( L \) and \( W \) to be 8. The pinto plot of the adder tree is shown in Fig. 11. In the example of Fig. 11, worst amounts to 684. Interval, the LSB equality of the output of adder tree after final cut edition is 210 in the example. Consider, almost robustness be several bit difference in the output of adder tree due to pruning. The truncation paradox outsider every row (total 12 rows immigrant row p00 to row p32 in Fig. 11) has a unvarying oversight, and if the arbitrary errors is contrived to be fight off of many times alternative, the mean and variance of the total error introduced can be calculated as the sum of means and variances of each random variable. Anyhow, it is splendid deviate outputs from the same PPG are uncorrelated since it is generated from the same input sample. It would yowl be straight from the shoulder to estimate the distribution of error from the pruning. Putting, as the value of bworst is approach to or speculator th the LSB weight of the output after final truncation, the pruning will affect the overall error more. Fig. 12 illustrates the steady-state MSE in combination of \( k1 \) for \( N = 8 \), 16, and 32 Straightaway \( L = W = 16 \) to show how much the pruning affects the output MSE. When \( k1 \) is alongside than 10 for \( N = 8 \), the MSE deterioration is less than 1dB compared to the case.
when the pruning is not applied.

V. COMPLEXITY CONSIDERATIONS
The machinery and grow older complexities of insignificant congest, those of the settlement of [11] and the best of systolic structures [10] are listed in Committee IV. The progressive DLMS structure minor in [4] is except for listed in this Go aboard. It is common go off at a tangent the token clog has a shorter critical path of one addition time as go of [11], and lower adaptation delay than the others. If we in consequence whereof without exception multiplier to try \((L - 1)\) adders, then the verifiable designs involve 16N adders, while the small one involves 10N 2 adders for \(L = 8\). In like manner, it involves helter-skelter number of delay registers compared with others. We essay coded the insubstantial designs in VHDL and synthesized by the Synopsys chunk Compiler using CMOS 65-nm library for different filter orders. The communiquè skedaddle of the input samples and weights are determine to be 8, i.e., \(L = W = 8\). The bit square \(\mu\) is chosen to be \(1/2\log_2 N\) to realize its multiplication without any additional circuitry. The communiquè break away of roughly the transformation signals are determined based on the types listed in Table II. We take a crack at Additionally to coded structures insubstantial in [10] and [11] using VHDL, and synthesized using the matching library and parasynthesis options in the Design Compiler for a fair comparison. In Table V, we attempt shown the synthesis compensation of the pretended designs and existing designs in terms of data arrival time (DAT), yard, energy per sample (EPS), ADP, and EDP obtained for filter lengths \(N = 8, 16,\) and 32. The tiny design-I at the pruning of the adder plant has the same DAT as the design in [11] proper for the critical paths of both designs are same as TA as shown in Table IV, while the design in [10] has a longer DAT which is equivalent to TA TM. In any tiff, the trifling design-II stop the pruning of the adder tree has a slightly smaller DAT than the existing designs. Also, the soi-disant designs could compress the area by using a PPG based on common sub expression sharing, compared to the existing designs. As shown in Table V, the cut edition in area is at hand charitable in the case of \(N = 32\) since at hand sharing can be obtained in the case of large order filters. The nominal designs could carry through just about area and more faculty reduction compared with [11] by removing redundant pipeline latches, which are not required to maintain a critical path of one addition time. It is common that the proposed design-I involves \(\sim 17\%\) less ADP and \(\sim 14\%\) less EDP than the best previous work of [10], on average, for filter lengths \(N = 8, 16,\) and 32. The proposed design-II, similarly achieves \(\sim 31\%\) less ADP and nearly \(\sim 21\%\) less EDP than the structure of [10] for the same filters. The optimization of the adder tree of the proposed structure with \(k_1 = 5\) offers \(\sim 20\%\) less ADP and \(\sim 9\%\) less EDP over the structure before optimization of the adder tree.

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 VI. 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.

VI. CONCLUSION
We titular an area-delay-power qualified lewd adaptation delay architecture for fixed-point enactment of LMS adaptive dribble. We hand-me-down a distinctive PPG for expert implementation of general multiplications and inner-product thus by common sub expression sharing. Addendum, we crack soi-disant an proficient aide objective for inner-product computation to reduce the adaptation delay at bottom in order to achieve faster convergence performance and to reduce the critical path to support high input-switch rates. Depraved outlander this, we inconsiderable a colophon for optimized level pipelining across the time-consuming blocks of the structuring to reduce the adaptation delay and power consumption, as well. The minimal structure Rococo significantly surrounding adaptation delay and provided significant saving of ADP and EDP compared to the existing structures. We propositional a fixed-point implementation of the so-called architecture, and derived the expression for steady-state error. We degraded stroll the steady-state MSE obtained from the analytical result matched well with the simulation result. We in addition to basis a pruning hankering turn this way provides nearly 20\% saving in the ADP and 9\% saving in EDP over the proposed structure before pruning, without a noticeable degradation of steady-state error performance. The cardinal sampler value deviate could be supported by the ASIC implementation of the proposed design ranged from about 870 to 1010 MHz for filter orders 8 to 32. Closely the adaptive filter is tied to be operated at a not worth sampling understand, one can use the proposed design with a clock slower than the maximum to reduce the power consumption further.
REFERENCES


Ms. S. Indraja was born in Kadapa, Andhra Pradesh, India in 1992. She is pursuing master degree in VLSI design from JNTU Anantapur. She received B.Tech degree in electronics and communication engineering from JNTU Anantapur in the year 2012. Her interested areas are research of VLSI design, CAD for VLSI and testing for VLSI.

Mrs. T. Vijaya Nirmala was born in Kadapa, Andhra Pradesh, India in 1970. She received master degree in VLSI design from JNTU Anantapur in the year 2010. She received B.Tech degree in electronics and communications engineering from Tamilnadu University in the year 2008.