IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 22 June 2022, accepted 17 July 2022, date of publication 20 July 2022, date of current version 26 July 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3192619

RESEARCH ARTICLE

Two Distributed Arithmetic Based High Throughput Architectures of Non-Pipelined LMS Adaptive Filters

MOHD. TASLEEM KHAN[1], MOHAMMED A. ALHARTOMI[2], (Member, IEEE), SAEED ALZAHRANI[2], (Member, IEEE), RAFI AHAMED SHAIK[3], (Member, IEEE), AND RUWAYBIH ALSULAMI[4], (Member, IEEE)

[1]Linköping University, 58183 Linköping, Sweden
[2]Department of Electrical Engineering, University of Tabuk, Tabuk 71491, Saudi Arabia
[3]Department of Electronics and Electrical Engineering, Indian Institute of Technology Guwahati, Guwahati 781039, India
[4]Department of Electrical Engineering, Umm Al-Qura University, Mecca 21961, Saudi Arabia

Corresponding author: Mohd. Tasleem Khan (mtkhan@iitism.ac.in)

This work was supported in part by the Deanship of Scientific Research at Umm Al-Qura University under Grant 22UQU4350362DSR01; and in part by the Deanship of Scientific Research, University of Tabuk, under Grant S-1442-0151.

ABSTRACT Distributed arithmetic (DA) is an efficient look-up table (LUT) based approach. The throughput of DA based implementation is limited by the LUT size. This paper presents two high-throughput architectures (Type I and II) of non-pipelined DA based least-mean-square (LMS) adaptive filters (ADFs) using two's complement (TC) and offset-binary coding (OBC) respectively. We formulate the LMS algorithm using the steepest descent approach with possible extension to its power-normalized LMS version and followed by its convergence properties. The coefficient update equation of LMS algorithm is then transformed via TC DA and OBC DA to design and develop non-pipelined architectures of ADFs. The proposed structures employ the LUT pre-decomposition technique to increase the throughput performance. It enables the same mapping scheme for concurrent update of the decomposed LUTs. An efficient fixed-point quantization model for the evaluation of proposed structures from a realistic point-of-view is also presented. It is found that Type II structure provides higher throughput than Type I structure at the expense of slow convergence rate with almost the same steady-state mean square error. Unlike existing non-pipelined LMS ADFs, the proposed structures offer very high throughput performance, especially with large order DA base units. Furthermore, they are capable of performing less number of additions in every filter cycle. Based on the simulation results, it is found that 256^{th} order filter with 8^{th} order DA base unit using Type I structure provides $9.41\times$ higher throughput while Type II structure provides $16.68\times$ higher throughput as compared to the best existing design. Synthesis results show that 32^{nd} order filter with 8^{th} order DA base unit using Type I structure achieves 38.76% less minimum sampling period (MSP), occupies 28.62% more area, consumes 67.18% more power, utilizes 49.06% more slice LUTs and 3.31% more flip-flops (FFs), whereas Type II structure achieves 51.25% less MSP, occupies 21.42% more area, consumes 47.84% more power, utilizes 29.10% more slice LUTs and 1.47% fewer FFs as compared to the best existing design.

INDEX TERMS Adaptive filter (ADF), distributed arithmetic (DA), finite-impulse response (FIR), least mean square (LMS), look-up table (LUT).

I. INTRODUCTION

Adaptive filters are extensively used in many digital signal processing applications such as system identification, echo cancellation and channel equalization [1], [2]. In many practical system, the sampling rate of input signals is close to the clock rate. Due to technology advancements, the clock rate is increasing for designing a system. Thus, it is desired that the adaptive filters (ADFs) must have a high throughput to meet the input sampling rates. The high-throughput ADF would

The associate editor coordinating the review of this manuscript and approving it for publication was Yunlong Cai.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

allow trade-off with area and power using very large scale integration (VLSI) methodologies [3].

A wide variety of adaptive filtering algorithms have been suggested [4]–[11], the most important from VLSI implementation point-of-view are perhaps based on least mean square (LMS) [4]. They are simple, robust and easy to implement on the hardware. There are LMS variants which can further simplify hardware realization of LMS at the expense of convergence performance e.g., sign-sign LMS [5]. On the other hand, the power-normalized sibling of LMS referred as normalized LMS (NLMS) is capable providing better convergence performance [6]. It is relatively more computational complex than LMS. Although recursive least squares (RLS) algorithm [7] based adaptive filters provide faster convergence, they are even more computational complex than LMS and its variants. A class of algorithms based on affine projection (AP) [8], also referred to as generalized NLMS has been developed [9]. It provides some compromise between the convergence performance and complexity features of the LMS and RLS algorithms. Subsequently, several variants of AP have been suggested such as fast AP (FAP) [10] to increase the speed, modified FAP [11] to improve the numerical stablility etc. It is believed that Gauss Seidel FAP can be implemented on hardware using logarithmic number system (LNS) [12]. Similarly, algorithms such as a priori Error-Feedback LSL [13], normalized RLS [14] based on LNS are expected to provide better convergence performance than conventional LMS. However, the LNS arithmetic was restricted to low-precision applications due to the difficulty in performing addition and subtraction on large wordlengths. Furthermore, the addition and subtraction in LNS cause offset leading to numerical stability issues. In contrast, the floating-point could be used in place of LNS, but it requires typically larger, complex, and much slower adder and subtractor units than its fixed-point counterpart.

Based on the aforementioned reasons, it is clear that the fixed point LMS algorithm is the front-runner among different LMS variants because of its satisfactory numerical stability and ease of hardware implementation. Nonetheless, it could also result in high throughput performance as it is free from complex operations such as division, exponent, norms, etc. which might increase the critical path delay. In the LMS unit, finite-impulse-response (FIR) filter is the main computational block. It consists of several multipliers depending upon the filter order which occupies a large chip area and consumes a lot of power. Since the number of multipliers grows linearly with filter order, the real-time implementation of filters is a challenging task. Furthermore, the throughput of such implementation is limited by the critical path delay which grows linearly with the filter order. Thus, there is a need to shorten the critical path to meet the sampling rate of input signals. Pipelining could be employed to reduce the critical path delay, however, it changes the convergence properties of LMS ADF and makes the system less accurate [19], [20]. The depth of pipelining further degrades the accumulation of the past input samples for fixed-point implementation of LMS ADFs.

Due to the progressive scaling of silicon devices over the past several years, semiconductor memory has become inexpensive, high-speed and power-efficient. As per the projections of the international technology roadmap for semiconductors (ITRS) [21], embedded memories will continue to dominate in system-on-chip, for instance, at present, it is roughly more than 90% of total SoC content [22]. It is found that the packing density of transistors in SRAM is not only high but also increasing much faster than the transistor density of logic devices [23]. Distributed arithmetic (DA) is an efficient memory or look-up table (LUT) based approach used for the implementation of FIR filter [24]. It is classified as two's complement (TC) or offset binary coding (OBC) DA depending on the representation of filter coefficients (or input samples). It consists of a look-up table (LUT) and a shift-accumulate (SA) unit [25], [26]. LUT stores the filter partial products while the SA unit produces the output by successive shift and accumulation of filter partial products for a certain number of clock cycles. The LUT size grows exponentially with filter order, hence limiting the system throughput for higher-order filter realization. Several authors alleviated the throughput limitation of DA based LMS ADFs using different approaches [27]–[34]. Allred *et al*. employed two LUTs, one for filtering and the other for coefficient update operation [27]. The system throughput is limited by the sizes of both LUTs, especially when the filter order becomes large. Guo and DeBrunner proposed two high-throughput architectures for LMS ADFs using TC DA and OBC DA [28]. In their work, they showed that a single LUT can be employed by performing parallel filtering and coefficient update operations. The throughput achieved is considerably higher as compared to [27]. Surya and Rafi [29] presented a new high-throughput architecture for LMS ADF using OBC DA. Recently, two architectures for LMS ADF based on OBC DA have been presented to replace the multipliers with LUTs [30]. One of them is implemented using straightforward OBC and the other is implemented with the half-latency algorithm. In the second approach, the input bits are split into even-odd components for concurrent processing through separate paths. Several DA based pipelined designs [35]–[39] have been presented at the expense of system accuracy and convergence properties. To the best of our knowledge, no study has been carried out to improve the throughput performance of non-pipelined LMS ADF. This motivates us to investigate two high-throughput LMS ADFs based on TC DA and OBC DA. The key contributions in this paper are as follows.

- Mathematical formulation of the LMS algorithm and inner products using TC DA and OBC DA.
- Architectural and algorithmic descriptions of the proposed designs based on LUT pre-decomposition.
- Concurrent LUT update scheme for each TC DA and OBC DA based design.
- Consolidate comparison of different non-pipelined LMS ADFs based on TC and OBC DA in terms of

computational complexities, convergence performance and implementations on application specific integrated circuits (ASIC) and field programmable gate array (FPGA) platforms.

The rest of the paper is organized as follows. In Section II, we present the LMS algorithm, its convergence properties and the formulation using OBC DA and TC DA. In next Section, we formulate the inner product using OBC DA and TC DA based on the LUT pre-decomposition. In Section IV, we describe the architectural and algorithmic details of the proposed designs are discussed. In Section V, the performance of the proposed and existing designs are compared in terms of computational complexities, convergence performance, area, power consumption and logic utilization. Conclusions are provided in Section VI.

II. LMS ALGORITHM

A. STEEPEST DESCENT AND LMS ALGORITHM

The adaptive filtering discussed here is of LMS type as presented by Widrow *et al*. [4]. At time instant n, the output y_n of N^{th} order ADF can be given as

$$y_n = \boldsymbol{w}_n^T \boldsymbol{x}_n = \sum_{i=0}^{N-1} w_n(i) x_{n-i} \tag{1}$$

where $\boldsymbol{x}_n = [x_n, x_{n-1}, \ldots, x_{n-N+1}]^T$ is the input vector and $\boldsymbol{w}_n = [w_n(0), w_n(1), \ldots, w_n(N-1)]^T$ is the coefficient vector. An error e_n is computed by subtracting y_n from the desired signal d_n as

$$e_n = d_n - y_n \tag{2}$$

It is desired that the coefficients are updated in every iteration based on the computed error given in (2). The coefficients are updated using steepest descent approach [15], according to

$$\boldsymbol{w}_{n+1} = \boldsymbol{w}_n - \frac{1}{2}\mu \boldsymbol{\alpha}_n \nabla_n \tag{3}$$

where μ is the step size and α_n is the additional scaling factor to make filter coefficients as fast as possible [18]. The selection of μ is critical as it determines how fast the coefficients converge to their optimal values (also known as Weiner coefficients [4]). For the conventional LMS and normalized LMS, its value $\alpha_n = 1$ and $\alpha_n = (\mathbb{E}[\boldsymbol{x}_n^T \boldsymbol{x}_n])^{-1}$ respectively. One common assumption that usually made the elements of input $\boldsymbol{x}_n$ are uncorrelated in time, zero mean, and jointly Gaussian [17]. The gradient ∇_n at time instant n is the partial derivative of the expected value of error with respect to the coefficients:

$$\nabla_n \triangleq \left.\frac{\partial\, \mathbb{E}[e_n^2]}{\partial \boldsymbol{w}}\right|_{\boldsymbol{w}=\boldsymbol{w}_n} \approx \tilde{\nabla}_n = \left.\frac{\partial e_n^2}{\partial \boldsymbol{w}}\right|_{\boldsymbol{w}=\boldsymbol{w}_n} \tag{4}$$

where $\mathbb{E}[\cdot]$ is the expectation operator. Since the computation of an ensemble average is quite difficult, an estimate of ∇_n defined by $\tilde{\nabla}_n$ is used. Using (2) and (4), the gradient $\tilde{\nabla}_n$ for the LMS algorithm can be obtained as $\tilde{\nabla}_n = -2e_n \boldsymbol{x}_n$, therefore, its coefficient update equation is given by

$$\boldsymbol{w}_{n+1} = \boldsymbol{w}_n + \mu e_n \boldsymbol{x}_n \tag{5}$$

Similarly, one can obtain the coefficient update equation for NLMS algorithm based on the above discussion. The primary focus of this work is to implement the high throughput VLSI architectures of LMS algorithm using DA. Therefore, we restrict our discussion to LMS algorithm only.

B. CONVERGENCE PROPERTIES OF THE LMS ALGORITHM

The convergence properties of the LMS algorithm include convergence rate, steady-state MSE, and step-size. Using (2), we can derive the MSE (ξ) of error e_n by squaring and applying the expectation on both the sides which would result

$$\xi \triangleq \mathbb{E}[e_n^2] = \left(\mathbb{E}\left[d_n^2\right] - 2\boldsymbol{p}^T \boldsymbol{w} - \boldsymbol{w}^T \boldsymbol{R} \boldsymbol{w}\right) \tag{6}$$

The correlation matrices $\boldsymbol{R}$ and cross-correlation vector $\boldsymbol{p}$ are, respectively, defined as

$$\boldsymbol{R} = \mathbb{E}\left[\boldsymbol{x}_n \boldsymbol{x}_n^T\right] \text{ and } \boldsymbol{p} = \mathbb{E}\left[\boldsymbol{x}_n d_n\right] \tag{7}$$

The estimate of MSE gradient for a fixed coefficient vector:

$$\tilde{\nabla}_n = -2\boldsymbol{p} + 2\boldsymbol{R}\boldsymbol{w} \tag{8}$$

One can find an alternative and useful expression based on (6) to calculate the MSE as

$$\xi = \xi_{\min} + \mathbb{E}\left[(\boldsymbol{w} - \boldsymbol{w}^*)^T \boldsymbol{R} (\boldsymbol{w} - \boldsymbol{w}^*)\right] \tag{9}$$

where $\sigma_d^2 = \mathbb{E}[d_n^2]$ is the power of desired signal and $\xi_{\min} = \sigma_d^2 - \boldsymbol{p}^T \boldsymbol{w}^*$ is the minimum MSE. For guaranteed convergence and algorithm stability, it is important to select an appropriate value of μ for a given choice of N. By defining the difference of coefficients from their optimal values as $\boldsymbol{v}_n = \boldsymbol{w}_n - \boldsymbol{w}^*$, we can re-express (9) as

$$\xi = \xi_{\min} + \mathbb{E}\left[\boldsymbol{v}^T \boldsymbol{R} \boldsymbol{v}\right] \tag{10}$$

The second term in (10) denotes the excess error which indicates how far the MSE is from its minimum value at time instant n. By taking the expectation on (5), the mean coefficient behaviour of the LMS ADF can be determined as follows

$$\mathbb{E}[\boldsymbol{w}_{n+1}] = (\boldsymbol{I} - \mu \boldsymbol{R})\, \mathbb{E}[\boldsymbol{w}_n] + \mu \boldsymbol{p} \tag{11}$$

From (11), we can find the steady-state coefficient vector if the convergence is assumed. It is possible when μ satisfies the following inequality:

$$0 < \mu < \frac{2}{\text{tr}[\boldsymbol{R}]} \approx \frac{2}{N\sigma_x^2} \tag{12}$$

where $\text{tr}[\cdot]$ is the trace operator and σ_x^2 is the power of input signal. In the steady-state, the filter coefficients can be obtained from the condition $\lim_{n\to\infty} \mathbb{E}[\boldsymbol{w}_n] = \boldsymbol{w}^*$ as

$$\boldsymbol{w}^* = \boldsymbol{R}^{-1} \boldsymbol{p} \tag{13}$$

As expected, $\boldsymbol{w}^*$ converges to the mean of MSE in (10), only if $\xi = \xi_{\min}$. It implies that the coefficients converge to their optimal values when the MSE approaches its minimum value.

In terms of $\boldsymbol{v}$, the coefficient update equation of the LMS algorithm in (5) modifies to

$$\boldsymbol{v}_{n+1} = (I - \mu \boldsymbol{R})\boldsymbol{v}_n + \mu e_{\mathrm{o},n}\boldsymbol{x}_n \tag{14}$$

where $e_{\mathrm{o},n} = d_n - \boldsymbol{w}^{*T}\boldsymbol{x}_n$ is error in the steady-state. By taking the expectation on both the sides of (14), we have

$$\mathbb{E}[\boldsymbol{v}_{n+1}] = (I - \mu \boldsymbol{R})\,\mathbb{E}[\boldsymbol{v}_n] \tag{15}$$

where $\mathbb{E}[e_{\mathrm{o},n}] = 0$. In general, (15) indicates a form of homogeneous difference equation whose solution depend on different modes of the convergence [15]. Hence, there will be N different relaxation time constants for the coefficients to converge

$$\tau_i = \frac{1}{2\mu\lambda_i} \tag{16}$$

where λ_i $(i = 0, 1, \ldots, N-1)$ represent the eigenvalues of $\boldsymbol{R}$. Assuming all the eigenvalues to be equal which is usually considered the special case [16]. Therefore, a single-mode is sufficient to describe the LMS ADF. Furthermore, the time constant given in (16) depicts how fast the algorithm converges for a given μ. A larger μ leads to slower convergence and vice-versa. In the following, we formulate the LMS algorithm using TC DA and OBC DA in the vector form.

C. FORMULATION OF LMS USING TC DA AND OBC DA

By representing the filter coefficients $\boldsymbol{w}_n$ in signed B-bit 2's complement form, we have

$$\boldsymbol{w}_n = \boldsymbol{W}_n \boldsymbol{s} \tag{17}$$

where $\boldsymbol{s} = [-2^0, 2^{-1}, \ldots, 2^{-(B-1)}]$ is the vector of binary weights and $\boldsymbol{W}_n = [w_n(i,j)]_{N\times B}$ is the coefficient bit-slice matrix in 2's complement form, where $w_n(i,j) \in [0,1]\ \forall\ 0 \le i \le N-1$ and $0 \le j \le B-1$. Substituting (17) in (5), we have

$$\boldsymbol{W}_{n+1}\boldsymbol{s} = \boldsymbol{W}_n\boldsymbol{s} + \mu e_n \boldsymbol{x}_n \tag{18}$$

It is clear from (18) that LMS algorithm using TC DA is same as the conventional LMS algorithm except that the coefficients are updated at the bit-level. Unlike (17), it is based on the representation of coefficients in OBC form. By writing the coefficients as $\boldsymbol{w}_n = \frac{1}{2}[\boldsymbol{w}_n - (-\boldsymbol{w}_n)] = \frac{1}{2}[\boldsymbol{w}_n - \overline{\boldsymbol{w}}_n - 1]$, where $\overline{\boldsymbol{w}}_n$ is the 1's complement of $\boldsymbol{w}_n$. Therefore, we can express the coefficients $\boldsymbol{w}_n$ using (17) in OBC form as

$$\boldsymbol{w}_n = \boldsymbol{W}_n\boldsymbol{s} = \frac{1}{2}\left[(\tilde{\boldsymbol{W}}_n - 1)\boldsymbol{s}\right] \tag{19}$$

where $\tilde{\boldsymbol{W}}_n = [\tilde{w}_n(i,j)]_{N\times B}\ \forall$ is the coefficient difference bit-slice matrix in OBC form with $\tilde{w}_n(i,j) \in [-1, 1]$. Substituting (19) in (5), and after some simplification, we have

$$\tilde{\boldsymbol{W}}_{n+1}\boldsymbol{s} = \tilde{\boldsymbol{W}}_n\boldsymbol{s} + 2\mu e_n \boldsymbol{x}_n \tag{20}$$

The coefficient update equation of LMS algorithm using OBC DA has an extra scaling of 2× alongwith the step-size.

III. INNER PRODUCT FORMULATION

In the previous Section, we formulate the LMS algorithm with OBC DA and TC DA. To design the LMS ADF, it is also required to formulate the inner product given by (1) using OBC DA and TC DA based on the proposed LUT pre-decomposition.

A. INNER PRODUCT USING TC DA

This design is based on the representation of filter coefficients $w_n(i)$ by signed 2's complement B-bit format as

$$w_n(i) = -w_n(i,0) + \sum_{j=1}^{B-1} w_n(i,j)2^{-j} \tag{21}$$

where $i \in [0, N-1]$. By substituting (21) in (1) and after some simplification, it results in

$$y_n = \sum_{j=0}^{B-1} a_n(j)2^{-j} \tag{22}$$

where $a_n(j) = (-1)^{\lfloor (B-1-j)/(B-1)\rfloor} b_n(j)$, $\lfloor\cdot\rfloor$ is the greatest integer function and $j \in [0, B-1]$. From (22), it is clear that filter partial products $a_n(j)$ undergoes shift-accumulation for B number of clock cycles with sign inversion at 0^{th} clock cycle. The term $b_n(j)$ can be expressed as

$$b_n(j) = \sum_{i=0}^{N-1} x_{n-i} w_n(i,j) \tag{23}$$

where $w_n(i,j) \in [0,1]$ denotes the bit-slices of filter coefficients, therefore, term $b_n(j)$ can be represented by 2^N binary combinations which can be pre-computed and stored in a LUT, as depicted in Fig. 1(a). From the contents of LUT, it can be noted that recent sample x_n is present at odd address locations of LUT, and hence the term $b_n(j)$ can be decomposed into even and odd address locations to form two smaller LUTs as

$$b_n^p(j) = \sum_{i=0}^{N-2} x_{n-j-1} w_n(i+1,j) + p x_n, \quad \forall\ w_n(0,j) \tag{24}$$

where $p = w_{0,j}$ which assumes 0 for even LUT (E-LUT) and 1 for odd LUT (O-LUT) with $0 \le j \le B-1$. By substituting $p = 0$ and $p = 1$ in (24), we get

$$b_n^1(j) = b_n^0(j) + x_n \tag{25}$$

i.e., the contents of O-LUT is same as that of E-LUT except for an additional term corresponding to recent sample x_n. This can be taken care by placing an extra adder outside the O-LUT block to perform the addition with x_n.

B. INNER PRODUCT USING OBC DA

Unlike TC DA, the filter coefficients are coded using OBC as follows: $w_i = \frac{1}{2}[w_i - \overline{w}_i]$, with $\overline{w}_i$ being the 2's complement of w_i. Using (21), this relation can be further written as

$$w_i = \frac{1}{2}\left[-(w_{i,0} - \overline{w}_{i,0}) + \sum_{j=1}^{B-1}(w_{i,j} - \overline{w}_{i,j})2^{-j} - 2^{-(B-1)}\right] \tag{26}$$

Addr	LUT Data [TC]
0 0 0 0	0
0 0 0 1	x_n
0 0 1 0	x_{n-1}
0 0 1 1	$x_{n-1}+x_n$
0 1 0 0	x_{n-2}
0 1 0 1	$x_{n-2}+x_n$
0 1 1 0	$x_{n-2}+x_{n-1}$
0 1 1 1	$x_{n-2}+x_{n-1}+x_n$
1 0 0 0	x_{n-3}
1 0 0 1	$x_{n-3}+x_n$
1 0 1 0	$x_{n-3}+x_{n-1}$
1 0 1 1	$x_{n-3}+x_{n-1}+x_n$
1 1 0 0	$x_{n-3}+x_{n-2}$
1 1 0 1	$x_{n-3}+x_{n-2}+x_n$
1 1 1 0	$x_{n-3}+x_{n-2}+x_{n-1}$
1 1 1 1	$x_{n-3}+x_{n-2}+x_{n-1}+x_n$

(a)

Addr	LUT Data [OBC]
0 0 0 0	$+\frac{1}{2}[+x_n+x_{n-1}+x_{n-2}+x_{n-3}]$
0 0 0 1	$+\frac{1}{2}[+x_n+x_{n-1}+x_{n-2}-x_{n-3}]$
0 0 1 0	$+\frac{1}{2}[+x_n+x_{n-1}-x_{n-2}+x_{n-3}]$
0 0 1 1	$+\frac{1}{2}[+x_n+x_{n-1}-x_{n-2}-x_{n-3}]$
0 1 0 0	$+\frac{1}{2}[+x_n-x_{n-1}+x_{n-2}+x_{n-3}]$
0 1 0 1	$+\frac{1}{2}[+x_n-x_{n-1}+x_{n-2}-x_{n-3}]$
0 1 1 0	$+\frac{1}{2}[+x_n-x_{n-1}-x_{n-2}+x_{n-3}]$
0 1 1 1	$+\frac{1}{2}[+x_n-x_{n-1}-x_{n-2}-x_{n-3}]$
1 0 0 0	$-\frac{1}{2}[+x_n-x_{n-1}-x_{n-2}-x_{n-3}]$
1 0 0 1	$-\frac{1}{2}[+x_n-x_{n-1}-x_{n-2}+x_{n-3}]$
1 0 1 0	$-\frac{1}{2}[+x_n-x_{n-1}+x_{n-2}-x_{n-3}]$
1 0 1 1	$-\frac{1}{2}[+x_n-x_{n-1}+x_{n-2}+x_{n-3}]$
1 1 0 0	$-\frac{1}{2}[+x_n+x_{n-1}-x_{n-2}-x_{n-3}]$
1 1 0 1	$-\frac{1}{2}[+x_n+x_{n-1}-x_{n-2}+x_{n-3}]$
1 1 1 0	$-\frac{1}{2}[+x_n+x_{n-1}+x_{n-2}-x_{n-3}]$
1 1 1 1	$-\frac{1}{2}[+x_n+x_{n-1}+x_{n-2}+x_{n-3}]$

(b)

FIGURE 1. LUT content of (a) TC DA (b) OBC DA.

Let $\Delta w_{i,j} = (w_{i,j} - \overline{w}_{i,j})$ denotes the difference of bit-slices of filter coefficients and their 2's complement. By substituting (26) into (1) and after simplification, we get

$$y_n = \sum_{j=0}^{B-1}\left(\frac{1}{2}\sum_{i=0}^{N-1} x_{n-i}c_{i,j}\right)2^{-j} - \left(\frac{1}{2}\sum_{i=0}^{N-1} x_{n-i}\right)2^{-(B-1)} \tag{27}$$

where $c_{i,j} = (-1)^{\lfloor (B-1-j)/(B-1)\rfloor}\Delta w_{i,j}$. Therefore, a more compact representation of y_n can be given as

$$y_n = \sum_{j=0}^{B-1} d_j 2^{-j} + d_{initial}2^{-(B-1)} \tag{28}$$

where the term d_j and $d_{initial}$, are respectively, expressed as

$$d_j = \frac{1}{2}\sum_{i=0}^{N-1} x_{n-i}c_{i,j}, \qquad d_{initial} = -\frac{1}{2}\sum_{i=0}^{N-1} x_{n-i} \tag{29}$$

Note that the term d_j would also have 2^N possible binary combinations of input samples which can be pre-computed and stored in LUT as similar to Type I structure. However, the nature of stored contents are different from Type I structure since $c_{i,j} \in [-1, 1]$. Corresponding to bit-value of $c_{N-1,j}$, the LUT contents present at first half and second half address locations are 2's complement of each other. Hence, it is only required to store first 2^{N-1} half combinations in LUT as shown in Fig. 1(b). Mathematically, the term d_j in (29) can be re-written as

$$d_j^q = \frac{(-1)^q}{2}\left[x_n + \sum_{i=0}^{N-2} x_{n-i-1}c_{i,j}\right] \tag{30}$$

where q denotes the sign of the contents present at first and second half address locations for 0 and 1 respectively. The LUT can be decomposed into even and odd address locations by re-writing (30) as

$$d_j^{q,p} = \frac{(-1)^q}{2}\sum_{i=0}^{N-2} x_{n-i}c_{i+1,j} + \frac{(-1)^p}{2}x_{n-N+1}, \quad \forall c_{0,j} \tag{31}$$

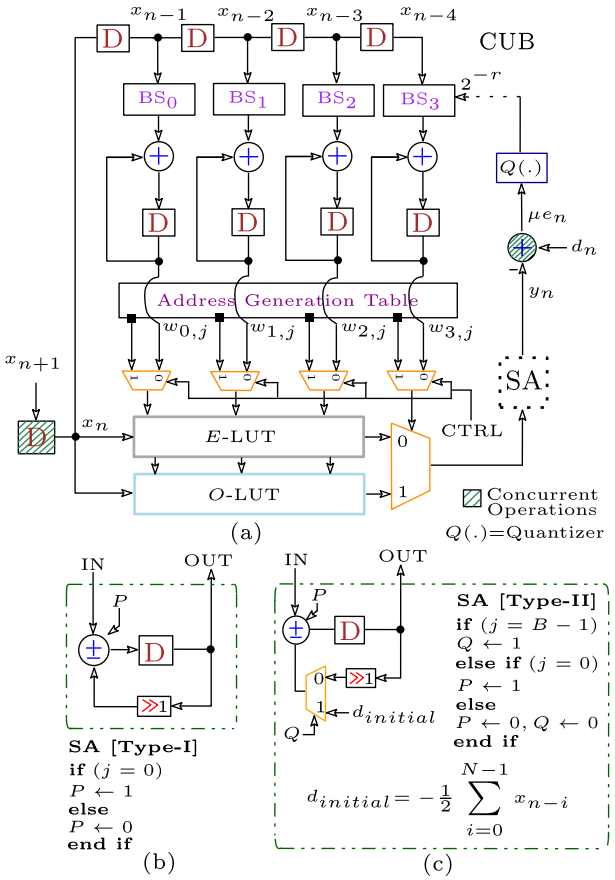

FIGURE 2. (a) Circuit schematic of the proposed 4th order LMS ADF using LUT pre-decomposition (b) SA unit for Type I structure (c) SA unit for Type II structure.

where p assumes 0 and 1 for E-LUT and O-LUT respectively. By carefully observing (31), a useful reduction in LUT can be obtained by noting that the contents at first half even address location for $q = 0, p = 0$ are 2's complement of the contents present at second half of odd address location for $q = 1$, $p = 1$ and vice-versa. By substituting the logic values of p and q in (31), we have

$$d_j^{0,0} = -d_j^{1,1} \text{ and } d_j^{1,0} = -d_j^{0,1} \tag{32}$$

Clearly, the contents of E-LUT and O-LUT correspond to $d_j^{0,0}$ and $d_j^{1,0}$ terms respectively. The other halves of E-LUT and O-LUT correspond to $d_j^{1,1}$ and $d_j^{0,1}$ which can be obtained by taking 2's complement on $d_j^{0,0}$ and $d_j^{1,0}$ respectively.

IV. PROPOSED ARCHITECTURES

The DA based non-pipelined LMS ADF schemes reported in the literature used LUTs to store the filter partial products in TC or OBC form. However, the size of LUT grows exponentially with filter order. Generally, LUT access time is the bottleneck for the speed of the whole system, especially, when the size of LUT becomes large. Therefore, reducing the LUT size is very important and is of great practical concern. One

possible solution is to pre-decompose the LUT into two small LUTs based on its even and odd address locations which half the access time, as briefly discussed in Section II. In this paper, we propose two new high throughput architectures for LMS ADFs based on TC DA and OBC DA which are referred as Type I and Type II respectively.

The proposed 4th order Type I and Type II structures of LMS ADF is shown in Fig. 2(a). In both the design, an error e_n is computed in parallel with the arrival of new sample x_{n+1} *i.e.*,

$$\begin{aligned} e_n &\leftarrow d_n - y_n \\ x_n &\leftarrow x_{n+1} \end{aligned} \tag{33}$$

Thus, an external register is needed as shown by hatched line in Fig. 2(a). Each filter coefficient is represented in vector form as $\boldsymbol{w}_n = \boldsymbol{W}_n \boldsymbol{s}$, therefore, one can re-written (1) as

$$y_n = \boldsymbol{s}^T [\boldsymbol{W}_n^T \boldsymbol{x}_n] \tag{34}$$

where $[\boldsymbol{W}_n^T \boldsymbol{x}_n]$ indicates the possible filter partial products. The combined product $\boldsymbol{s}^T [\boldsymbol{W}_n^T \boldsymbol{x}_n]$ is computed in parallel with the update of filter coefficients, according to

$$\begin{aligned} y_n &= \boldsymbol{s}^T [\boldsymbol{W}_n^T \boldsymbol{x}_n] \\ \boldsymbol{W}_{n+1} \boldsymbol{s} &= \boldsymbol{W}_n \boldsymbol{s} + \mu e_n \boldsymbol{x}_n \end{aligned} \tag{35}$$

Clearly, the presence of scaling-vector $\boldsymbol{s}$ in (35) allows the computation of filter output and the update of coefficients in parallel for every B clock cycles. Similarly, one can develop the aforementioned relations for OBC DA based LMS ADF.

The circuit schematic of the proposed 4th order filter with coefficient update block (CUB) is shown in Fig. 2(a). It consists of four-barrel shifters (BSs), four weight updating accumulators, and an address generation table. The control signals to the multiplexers of BS are provided in power-of-two *i.e.*, 2^{-r}, where the parameter r depends on the filter order N. The composite signal μe_n is then quantized into B-bits with its MSB considered. As a result, the multiplication between μ and error e_n as given in (35) is not needed. This assumption marginally affects the convergence of LMS ADF [27]. The convergence rate and misadjustments in the steady-state of the LMS filter are controlled by μ, for example, a small value of μ reduces the misadjustments, however, it slows down the convergence rate and vice-versa. The value of μ is considered as half of the upper bound obtained in (12). For simplicity, the step-size is taken as $1/N$. Therefore, the expression of r is given by $r = \log_2 N$. It is important to note that the complexity of BS would be large, especially when N is high. This can be avoided by pre-shifting the input samples. In that case, the expression of r becomes $r = \log_2 N - l$, where l is the number of pre-shifts on the input samples. For example, consider $N = 4$ and $l = 1$, we get the value of r as 1, *i.e.*, each level in BS provides a right shift by one. As shown in Fig. 2(b) and (c), the internal schematic of the SA unit of Type I and Type II structure are different, but similar in function except that the Type II structure requires

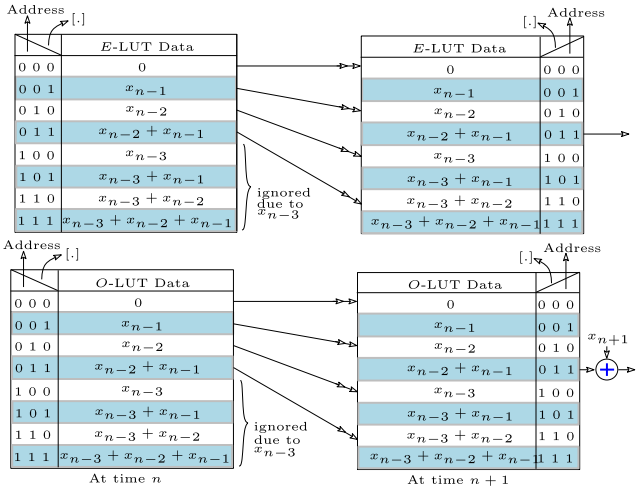

FIGURE 3. Proposed *E*-LUT and *O*-LUT update scheme for 4th order Type I structure from time n to $n+1$.

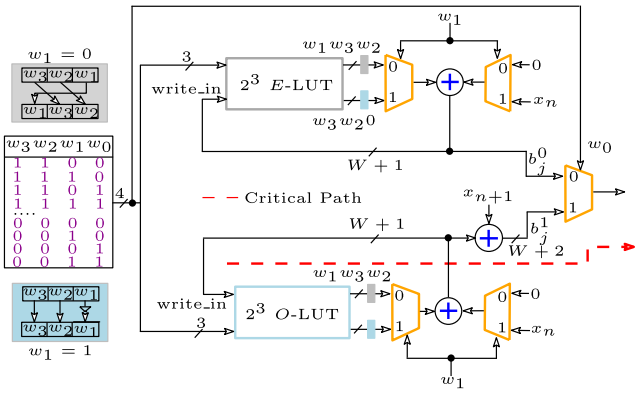

FIGURE 4. Architectural details of 4th order Type I structure.

$d_{initial}$ term to be added initially. The operation of SA unit begins with least-significant-bit (LSB) of filter coefficients *i.e.*, $j = B - 1$ and decreases until it reaches the most-significant-bit (MSB) *i.e.*, $j = 0$. And, when j becomes zero and the sign of input signal (IN) to SA unit must be reversed as per the DA principle. In the next two subsections, the design details of Type I and Type II structure are discussed.

A. TYPE I STRUCTURE

The proposed E-LUT and O-LUT update operations for 4th order filter Type I structure are illustrated in Fig. 3. The E-LUT and O-LUT contents have the same binary combinations of recent $N - 1$ input samples, as per (24). For example, the binary combination of input samples for $N = 4$ with input vector $[x_{n-1}, x_{n-2}, x_{n-3}]$ are required to store in each LUT. It may be noted from Fig. 3 that the contents of the first half address locations of each E-LUT and O-LUT at time instant n are used to generate the contents at time instant $n + 1$. Note that the second half address locations contain the oldest sample x_{n-3} term and are not needed during LUT update operation. It may also be noted that the contents at

Algorithm 1 Explains the Proposed Filter Using Type I Structure

Initialize:
$w'_{i,j} = w_{i,j},\ i \in [0, N-1],\ j = [0, B-1]$
loop
  $y_n = \sum_{j=0}^{B-1} a_j 2^{-j}$
  if CTRL== 1 **then**
    for $w_{i,j}$ =$2^{\max(i)} - 1$ to 0, $i \neq 0,\ \forall\ w_{0,j}$ **do**
      if $w_{i,j}\%2 == 0$ **then**
        $E_{n+1}^{w_{i,j}} \leftarrow E_n^{w_{i,j}/2}$
        $O_{n+1}^{w_{i,j}} \leftarrow O_n^{w_{i,j}/2}$
      else
        $E_{n+1}^{w_{i,j}} \leftarrow E_{n+1}^{w_{i,j}-1} + x_n$ -
        $O_{n+1}^{w_{i,j}} \leftarrow O_{n+1}^{w_{i,j}-1} + x_n$
      end if
    end for
  else
    for $j = B-1$ to 0 **do**
  $w_{n+1}^E \leftarrow w_n^E + \frac{e_n}{N}.\{E_n^{w_{i,j}}\}$
  $w_{n+1}^O \leftarrow w_n^O + \frac{e_n}{N}.\{O_n^{w_{i,j}} + x_{n+1}\}$
  $a_j = \sigma_j\{w_{0,j}.(O_n^{w_{i,j}} + x_{n+1}) + \overline{w_{0,j}}.E_n^{w_{i,j}}\}$
    ▷ $\sigma_j = (-1)^{\lfloor \frac{B-1-j}{B-1} \rfloor}$
    end for
  end if
  for every *next sample* **do**
  $x_n \leftarrow D\{x_{n+1}\}$; ▷ $D\{.\}$ = delay operator
  $e_n \leftarrow d_n - y_n$;
  end for
  $w'_{i,j} \leftarrow w_{i,j}, n \leftarrow n+1$
end loop

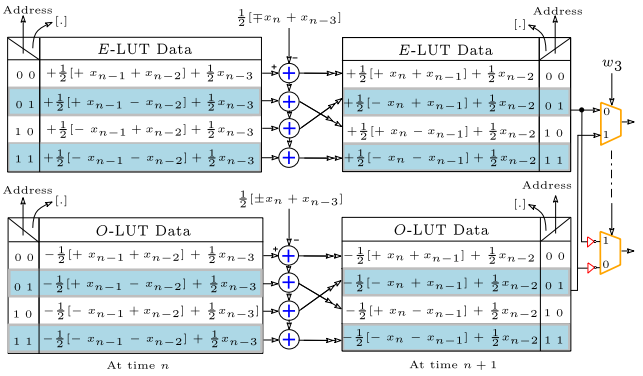

FIGURE 5. Proposed *E*-LUT and *O*-LUT update scheme for 4th order Type II structure from time *n* to *n* + 1.

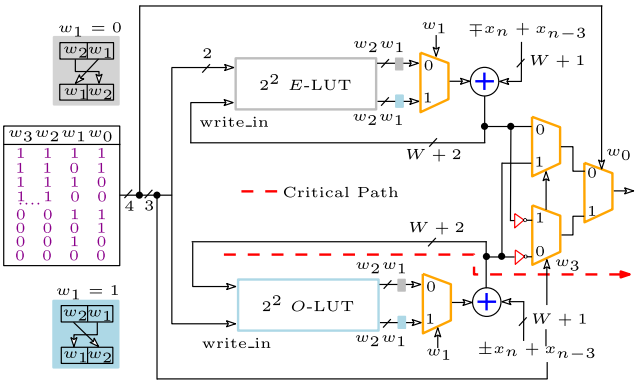

FIGURE 6. Architectural details of 4th order Type II structure.

even address locations of *E*-LUT and *O*-LUT at time instant $n+1$ are the re-mapped versions of the contents present in the first half address locations at time instant n. While the contents at odd address locations of *E*-LUT and *O*-LUT are obtained by adding x_n with the contents present at time n of the previous address location. Further, the update of *O*-LUT requires the addition of a new sample x_{n+1}, as required by (24). Mathematically, the entries of *E*-LUT and *O*-LUT for the next iteration can be written as

$$\begin{aligned} E_{n+1}^{s} &\leftarrow px_n + E_n^{s/2^{1-p}-p} \\ O_{n+1}^{s} &\leftarrow px_n + O_n^{s/2^{1-p}-p} + x_{n+1} \quad \forall w_{0,j} \end{aligned} \tag{36}$$

where s and p denote the address index and its modulo 2 operation, *i.e.*, $s \in [0, 2^{N-1}]$ and $p \equiv s \pmod 2$. The LUTs are updated with the sequence of addresses as shown in Fig. 4 using the address generation table. It is important to note that when $w_{0,j} = 0$ or 1 the even address locations of both LUTs are first updated followed by their odd address locations. By doing so, the concurrent operation between the LUTs is maintained at the cost of a few adders and 2-to-1 multiplexers. The Algorithm 1 explains the operation of the proposed Type I structure.

B. TYPE II STRUCTURE

The proposed *E*-LUT and *O*-LUT update operations for 4th order filter Type II structure are illustrated in Fig. 6. A novel strategy is proposed to update the LUTs which involves two external terms ($\frac{1}{2}[\mp x_n + x_{n-3}]$, $\frac{1}{2}[\pm x_n + x_{n-3}]$). It indicates the addition (or subtraction) of recent sample x_n with the oldest sample x_{n-3}. As mentioned, the symmetry in the OBC combinations allows only half to be stored in LUT and remaining half can be obtained by taking 2's complement using a 2-to-1 multiplexer. Consider the update of *E*-LUT and *O*-LUT from n to $n+1$ iteration. An external term $\frac{1}{2}[\mp x_n + x_{n-3}]$ is needed to update *E*-LUT, where '$-$' and '$+$' corresponds to even and odd address locations respectively. As an example, the subtraction of $+\frac{1}{2}[+x_n + x_{n-3}]$ to the contents of 01 address location in *E*-LUT at time instant n results in $+\frac{1}{2}[+x_n - x_{n-1}] + \frac{1}{2}x_{n-2}$, which is independent of oldest sample x_{n-3}. By close observation, it is nothing but the contents of 10 address location in *E*-LUT at time instant $n+1$. This re-mapping can be accomplished by performing left rotation on the address bits. In similar manner, the update of *O*-LUT requires an external term $\frac{1}{2}[\pm x_n + x_{n-3}]$ can be explained, where $+$ and $-$ corresponds to even and odd address locations respectively. Mathematically, the entries of

Algorithm 2 Explains the Proposed Filter Using Type II Structure

Initialize:
$w_{i,j} = w'_{i,j}$, $c_{i,j} = w_{N-1,j} \oplus w_{i,j}$
$i \in [0, N-2], j \in [0, B-1]$, '$\oplus$': XOR operator
loop
$y_n = \sum_{j=0}^{B-1} d_j 2^{-j} + d_{initial} 2^{-(B-1)}$
  if CTRL== 1 **then**
    for $c_{i,j} = 2^{\max(i)-1} - 1$ to 0, $i \neq 0$, $\forall\, c_{0,j}$ **do**
      if $c_{i,j}\%2 == 0$ **then**
        $E_{n+1}^{c_{i,j}/2} \leftarrow E_n^{c_{i,j}} - \frac{1}{2}[-x_n + x_{n-N+1}]$
        $O_{n+1}^{c_{i,j}/2} \leftarrow O_n^{c_{i,j}} - \frac{1}{2}[+x_n + x_{n-N+1}]$
      else
        $E_{n+1}^{c_{i,j}/2} \leftarrow E_n^{c_{i,j}} - \frac{1}{2}[+x_n + x_{n-N+1}]$
        $O_{n+1}^{c_{i,j}/2} \leftarrow O_n^{c_{i,j}} - \frac{1}{2}[-x_n + x_{n-N+1}]$
      end if
    end for
  else
    for $j = B-1$ to 0 **do**
      $c_{n+1}^E \leftarrow c_n^E + \{\frac{2e_n}{N}\}\{c_{N-1,j}.O_n^{c_{i,j}} + \overline{c_{N-1,j}}.E_n^{c_{i,j}}\}$
      $c_{n+1}^O \leftarrow c_n^O - \{\frac{2e_n}{N}\}\{c_{N-1,j}.E_n^{c_{i,j}} + \overline{c_{N-1,j}}.O_n^{c_{i,j}}\}$
      if $j == B-1$ **then**
        $d_0 = \{-c_{0,0}.O_n^{c_{i,0}} + \overline{c_{0,0}}.E_n^{c_{i,0}}\} + d_{initial}$
      else
        $d_j = \sigma_j\{-c_{0,j}.O_n^{c_{i,j}} + \overline{c_{0,j}}.E_n^{c_{i,j}}\}$
      end if
    end for
  end if
  for every *next sample* **do**
  $x_n \leftarrow D\{x_{n+1}\}$; ▷ $D\{.\}$ = delay operator
  $e_n \leftarrow d_n - y_n$;
  end for
  $d_{initial} \overset{D}{\leftarrow} E_{n+1}^0$
  $c'_{i,j} \leftarrow c_{i,j}$, $n \leftarrow n+1$
end loop

E-LUT and O-LUT at $n+1$-iteration can be expressed as

$$E_{n+1}^{s/2} \leftarrow E_n^s - \frac{1}{2}[(-1)^{1-p}x_n + x_{n-3}]$$
$$O_{n+1}^{s/2} \leftarrow O_n^s - \frac{1}{2}[(-1)^p x_n + x_{n-3}] \quad \forall c_{0,j} \tag{37}$$

where the definitions of s and p are the same as that of Type I structure. As discussed, the remaining combinations of E-LUT and O-LUT at time instant $n+1$ can be obtained by symmetries in OBC combinations, as per

$$E_{n+1}^{s/2} \leftarrow +[q.O_{n+1}^{s/2} + (1-q).E_{n+1}^{s/2}]$$
$$O_{n+1}^{s/2} \leftarrow -[q.E_{n+1}^{s/2} + (1-q).O_{n+1}^{s/2}] \tag{38}$$

where $q \equiv s \pmod{2^{N-1}}$. As it can be seen from (38) that the remaining contents of E-LUT and O-LUT can be obtained from O-LUT and E-LUT by setting $q = 1$ respectively. In other words, the $d_j^{1,1}$ and $d_j^{0,1}$ terms in (15) can be obtained by taking 2's complement on the stored contents of E-LUT and O-LUT respectively. Therefore, the concurrent update of LUTs is possible with extra two adders and two 2-to-1 multiplexers. It can be noted from Fig. 6 that the design has a smaller critical path and utilizes all LUT contents as compared to the Type I structure. However, it requires the update of $d_{initial}$ term, in accordance with (29). From analysis, it is found that there are two possible ways to update $d_{initial}$ term. In first method, the $d_{initial}$ term is stored in a register and updated with the E-LUT content present at 0^{th} address location. In second method, $\frac{1}{2}[-x_{n+1} + x_{n-2}]$ is stored in a register and added with $d_{initial}$ term. The first method is preferred over the second method due to fewer involved clock cycles. The Algorithm 2 explains the operation of the proposed Type II structure.

C. FIXED-POINT DESIGN CONSIDERATION

For fixed-point (FP) implementation of Type I and Type II designs, it is important to select the wordlengths for inputs ($\boldsymbol{x}$), coefficients ($\boldsymbol{w}$), and other intermediate signals ($\boldsymbol{o}$). Let (X_f^v, X_i^v) be a FP representation of a wordlength X^v with $v \in \{\boldsymbol{x}, \boldsymbol{w}, \boldsymbol{o}\}$, X_f^v is the fractional part and and X_i^v is the integer part. For the sake of clarity and without loss of generality, we replace the previous notation W and B by X_f^x and X_f^w to represent the fractional parts of inputs and coefficients respectively. Note that the wordlength of $\boldsymbol{x}, \boldsymbol{w}$, and $\boldsymbol{o}$ needs to be considered as the design constraints for desired accuracy and implementation complexity. From the above discussion, we can find that the FP representation of inputs and coefficients as (X_f^x, X_i^x) and (X_f^w, X_i^w) respectively. In the case of Type I and Type II designs, each E-LUT and O-LUT stores the $(N-1)$ inputs, however, Type II have $1/2$ scaling with the inputs. Therefore, the LUTs in Type I and Type II designs would have the FP representations as $(X_f^x + \lceil \log_2(N-1) \rceil, X_i^x + \lceil \log_2(N-1) \rceil)$ and $(X_f^x + \lceil \log_2(N-1) \rceil - 1, X_i^x + \lceil \log_2(N-1) \rceil - 1)$ respectively. The output of O-LUT is added with a new sample x_n after its update in Type I design. While Type II design does not have any extra adder but rather performs 2's complement at the output of O-LUT through a separate 2-to-1 multiplexer. Therefore, the FP representation of Type I and Type II at the output of 2-to-1 multiplexer used in LUT pre-decomposition would be $(X_f^x + \lceil \log_2 N \rceil, X_i^x + \lceil \log_2 N \rceil$ and $(X_f^x + \lceil \log_2 N \rceil - 1, X_i^x + \lceil \log_2 N \rceil - 1)$. Note that 2's complement at the output of O-LUT in Type II design needs to extend the sign-bit by 1. In the proposed architectures, the SA unit and CUB run operate concurrently. It implies that the precision of the register in the SA unit should be the same as the register in CUB. Therefore, the FP representation at the output of SA unit for Type I design would be $(X_f^x + \lceil \log_2 N \rceil + \Delta X_f, X_i^x + \lceil \log_2 N \rceil + \Delta X_i)$, where $(\Delta X_f, \Delta X_i) = (X_f^w - X_f^x, X_i^w - X_i^x)$. Depending on the wordlengths of inputs and coefficients, it is possible to make the same precision of registers in the SA unit and CUB. Similarly, one can obtain the expression of FP representation for Type II design. It may be noted that the SA unit performs right-shift by 1 for each cycle

TABLE 1. General comparison of computational complexities of different DA based Non-Pipelined LMS ADF architectures.

Design	ADD (per FC)	REG	MUX	CRITICAL PATH	FC	LUT
$^{\diamond}$DA$_0$ [27]	$m(2^{k-1}+2^k)+mW-1$	$m(1+k)$	–	$T_L+(k-1)T_M+T_A$	$2^k+\max(W,2^{k-1})+\lceil\log_2 m\rceil$	$2m(2^k-1)$
$^{\diamond}$DA$_1^*$ [28]	$m(2^{k-1}+k)+mB-1$	$m(1+2k)$	mk	T_L+T_A	$2^{k-1}+\lceil\log_2 m\rceil+B+1$	$m(2^k-1)$
$^{\clubsuit}$DA$_1^{**}$ [28]	$m(2^{k-1}+1+k)+mB+1$	$m(2+2k)$	mk	$T'_L+T_M+T_A$	$2^{k-1}+\lceil\log_2 m\rceil+B+1$	$m2^{k-1}$
$^{\clubsuit}$DA$_2$ [29]	$m(2^{k/2+1}+2^{k/2+2}+2)+mW+1$	$m(3+k)$	$6m$	$T''_L+(k+1)T_M+2T_A$	$2^{k/2}+\max(W,2^{k/2-1})+\lceil\log_2 m\rceil+1$	$2m2^{k-1}$
$^{\clubsuit}$DA$_3^{\circ}$ [30]	$m(2^{k-2}+2^{k-1}+1)+mW+1$	$m(3+k)$	$4m$	$T''_L+(k-1)T_M+2T_A$	$2^{k-1}+\max(W,2^{k-2})+\lceil\log_2 m\rceil+1$	$2m2^{k-1}$
$^{\clubsuit}$DA$_3^{\circ\circ}$ [30]	$m(2^{k-2}+2^{k-1}+2)+m(W/2)+1$	$m(3+k)$	$8m$	$T''_L+(k+1)T_M+2T_A$	$2^{k-1}+\max(W/2,2^{k-2})+\lceil\log_2 m\rceil+1$	$2m2^{k-1}$
$^{\diamond}$Type I	$m(2^{k/2+1}+1+k)+mB-1$	$m(2+2k)$	$(5+k)m$	$T'_L+T_M+2T_A$	$2^{k/2-1}+B+\lceil\log_2 m\rceil+1$	$2m(2^{k-1}-1)$
$^{\clubsuit}$Type II	$m(2^{k/2+1}+2+k)+mB+1$	$m(3+2k)$	$(6+k)m$	$T''_L+2T_M+T_A$	$2^{k/2-1}+B+\lceil\log_2 m\rceil+1$	$2m2^{k-2}$

ADD (per FC): Additions per filter cycle (FC), **REG:** Registers, **MUX:** Multiplexers, **LUT:** Look-up table, $^{\diamond}$: TC DA, $^{\clubsuit}$: OBC DA; $N=m\times k$, Throughput = 1/(total number of clock cycles × critical path), $n_0=2^k+\max(W,2^{k-1})+\lceil\log_2 m\rceil$, $n_1=2^{k-1}+\lceil\log_2 m\rceil+B+1$, $n_2=2^{k/2}+\max(W,2^{k/2-1})+\lceil\log_2 m\rceil+1$, $n_3=2^{k-1}+\max(W,2^{k-2})+\lceil\log_2 m\rceil+1$, $n'_3=2^{k-1}+\max(W/2,2^{k-2})+\lceil\log_2 m\rceil+1$ and $n_4=2^{k/2-1}+B+\lceil\log_2 m\rceil+1$ where W,B are wordlengths of inputs and filter coefficients, respectively. In addition, DA$_0$, DA$_1$, DA$_2$, DA$_3$, DA$_4$, Type I and Type II schemes have m, mk, mk, m, mk, mk and mk barrel shifters respectively. Further, DA$_0$, DA$_2$ and DA$_3$ schemes have $mk(k-1)$ 2-to-1 multiplexers (bit-level) for rotation of LUT addresses. All the OBC DA based designs have $(k-1)m$ XOR gates at their address lines. The notations $T_L/T'_L/T''_L$, T_M, T_A, T_{FA}, T_X and T_D are delay due to LUT access, 2-to-1 multiplexer and B-bit adder, one-bit full adder, XOR gate and accumulation register, respectively.

of accumulation, therefore, FP representation would not be altered. Usually, the FP representation of the desired input d_n is the same as output y_n and possess same quantization as that of input x_n. This is achievable with specific scaling, sign-extension and truncation or zero-padding for a given requirement. As the LMS algorithm starts learning, the output y_n must have the same sign as that of desired input d_n. This would allow the error e_n to have the same FP representation after the subtraction without overflow. In [41], it is shown that the convergence of an N^{th} order fixed-point LMS ADF with equal fractional parts of the inputs and coefficients would result in MSE as

$$\xi=\xi_{\min}+\mathbb{E}\left[\boldsymbol{v}^T\boldsymbol{R}\boldsymbol{v}\right]+\xi_q \tag{39}$$

It is clear from (39) that the total MSE ξ as given in (10) is increased by the MSE due to quantization error ξ_q. By again following the derivation in [41], we can find the MSE due to excess error and quantization error, respectively, as

$$\mathbb{E}\left[\boldsymbol{v}^T\boldsymbol{R}\boldsymbol{v}\right]=\frac{1}{2}\mu\xi_{\min}\text{tr}\boldsymbol{R} \tag{40}$$

$$\xi_q=\frac{N\sigma_c^2}{2a^2\mu}+\frac{1}{a^2}(|\boldsymbol{w}^*|^2+c)\sigma_x^2 \tag{41}$$

where $\sigma_x^2=2^{-2X_f^x}/12$, $\sigma_c^2=2^{-2X_f^w}/12$; a is the input range adjustment factor and c is related with the quantization on inner product. In both the design, we assume that the inputs are already scaled to $[-1,1)$, so the value of a is unity, and the quantization is performed after the bit-level accumulation of partial products through adder tree, so the value of c is also unity. Clearly, the MSE due to quantization error is related with the wordlengths of inputs/coefficients and step-size, while the MSE due to excess error depends on the input power. In the description of Type I and Type II structures, we have already considered the value of μ as 2^{-r}. Based on the FP representation, one can find the upper limit on r for Type I and Type II designs as $X_i^x+\lceil\log_2 N\rceil+\Delta X_i$ and $X_i^x+\lceil\log_2 N\rceil-1+\Delta X_i$ respectively. Since μ is in power of two, its multiplication with e_n is equivalent to the shifting of the radix-point. If we require to have a smaller μ, *i.e.*, $r>X_i^x+\lceil\log_2 N\rceil+\Delta X_i$ for Type I and $r>X_i^x+\lceil\log_2 N\rceil-1+\Delta X_i$ for Type II, then some of the least-significant-bits (LSBs) of e_n are required to be truncated. Consider $r=X_i^x+\lceil\log_2 N\rceil+\Delta X_i$, the corresponding step size would be $\mu=2^{-(X_i^x+\lceil\log_2 N\rceil+\Delta X_i)}$. Consequently, it would result the FP representation of μe_n as (X_f^w,X_i^w) without any truncation. It is important to note that the FP representation of Type I and Type II after μ scaling are same. This is because 1-bit less in the wordlength of Type II design is compensated by twice μ, in accordance with (20). The coefficient increment term $\mu e_n\boldsymbol{x}_n$ as obtained at the output of BS in Fig. 2 is required to have FP representation $(X_f^w+X_f^x,X_i^w+X_i^x)$. It can be noted that only X_i^w MSBs in the computations of the CUB are to be retained, while the remaining MSBs are required to be discarded. This could be understood from the fact that as the coefficients converge toward the optimal values, the coefficient increment term $\mu e_n\boldsymbol{x}_n$ becomes smaller, and the error e_n has more leading number of zeros in the MSBs. In both the designs, $X_f^x-X_i^x$ LSBs of $\mu e_n\boldsymbol{x}_n$ are truncated so that the terms have the same FP representation as to the coefficient values and can be fed as the addresses to the E-LUT and O-LUT. We have also assumed that the addition in the coefficient update does not cause overflow. Otherwise, we need to increase the coefficients wordlength in every iteration, which is undesirable. The assumption holds good as the coefficient increment terms become smaller when the coefficients start converging. During the training period, there are chances that overflow might occur, the update of coefficient may not be appropriate and cause additional iterations to reach the steady-state. In such a case, the updated coefficient can be computed in the truncated form (X_f^w,X_i^w) and can be fed as the addresses to the E-LUT and O-LUT.

V. RESULTS AND DISCUSSION

In this Section, we compare the performance of different non-pipelined DA based LMS ADFs in terms of computational complexities, convergence performance and implementation results. For the sake of discussion, we refer to the designs in [27]–[30] as DA$_{0-3}$ respectively. Note that

DA_1 comprises two architectures based on TC DA and OBC DA which we refer them as DA_1^* and DA_1^{**} respectively. Likewise, DA_3 also have two designs which we refer them as DA_1° and $DA_1^{\circ\circ}$. For clarity, they are classified as TC DA (◇) and OBC DA (♣) in Table 1. Assuming N to be composite, they can be expressed into m DA base units of order k such that $N = m \times k$. By doing so, the exponential growth of LUT size in different designs with filter order can be reduced [27]–[30].

A. COMPUTATIONAL COMPLEXITIES

The hardware complexities of the proposed and DA_{0-3} designs in terms of the number of additions (ADD) per filter cycle (FC), registers (REG), and multiplexers (MUX) are listed in Table 1. It is well known that the LUT size in TC DA based designs is double that of LUT size in OBC DA designs as it does not possess mirror symmetry. As indicated in Table 1, the LUT size in Type I and Type II structures are almost the same as that of DA_1^* and DA_1^{**} respectively. While the LUT size of DA_2 and DA_3 schemes are double of Type II structure since they employed auxiliary LUT for the coefficient update. The FC is a useful measure to indicate the number of clock cycles needed to perform the filtering operation in every iteration. For a given FC, we have estimated the total number of additions needed for each design. It includes the number of additions involved in the LUT update, SA unit, and CUB. As listed in Table 1, it appears that the number of additions for both the proposed structures are reduced over DA_{0-3}. This is primarily due to the concurrent update of E-LUT and O-LUT with the same mapping scheme. In Fig. 7, we illustrate the savings in number of additions per FC with respect to DA_0 for 4^{th} and 8^{th} orders DA base unit. The DA_0 design is considered a reference as it involves the highest number of additions per FC. It is clear that the savings in the number of additions per FC compared to DA_0 for the proposed structures with 8^{th} order DA base unit that are better than $DA_3^{\circ\circ}$ with 4^{th} order DA base unit. Specifically, Type I offers $4.04\times$ savings while Type II offers $3.96\times$ savings are better than $DA_3^{\circ\circ}$ for $k = 8$. The proposed structures, however, occupy more number of registers and multiplexers for any DA base unit order as compared to other designs. This is mainly due to the LUT pre-decomposition and use of an external register. For instance, the LUT pre-decomposition for both the designs involves combinational logic in terms of a 2-to-1 multiplexer and an adder (Type I) or inverted inputs 2-to-1 multiplexer (Type II). In the case of Type-II, the complexity of registers and multiplexers further increased over Type-II architectures due to the update of $d_{initial}$ term, and subsequently loaded into the SA unit. The number of registers and multiplexers for 32^{nd} order filter with $k = 8$ in Type II structure 5.55% and 7.69% higher than Type I structure.

We have also listed the time complexities of different designs in terms of critical path delay and FC in Table 1. The critical path of each design includes the access delay of LUT and adder and/or multiplexer delay during the filtering operation and LUT update. The critical paths of the proposed structures are different, as they are based on different algorithms for the LUT update. As shown in Fig. 6, the critical path includes the access delay of LUT, delays of adder, and 2-to-1 multiplexers. It can be seen that the critical paths of the proposed structures do not depend on the order of DA base units similar to DA_1^* and DA_1^{**}. This implies that there would be a significant reduction in the clock period as compared to the $DA_{0,2,3}$. Furthermore, the simultaneous access of two smaller LUTs in the proposed structures would result in lesser LUT access time as compared to a single LUT architecture in DA_0 and DA_1. From the discussion so far, it is clear that all the designs are dependent on LUT size, and updating them requires the longest time in the entire system. Due to simultaneous concurrency in filtering with coefficient update operation, the update of the external register with the computation of e_n and the update of E-LUT and O-LUT together, both the proposed structures enable high throughput implementation. Precisely, the update of filtering or coefficient update operation takes B clock cycles, the update external register or the computation of e_n requires single clock cycle, the update of E-LUT and O-LUT together takes $2^{k/2-1}$ clock cycles. Since the proposed structures also employ the decomposition $N = m \times k$, the resulting adder tree for the addition of outputs from DA base units would consume $\lceil \log_2(m) \rceil$ clock cycles. Hence, both the proposed structures would take a total of $2^{k/2-1} + B + \lceil \log_2 m \rceil + 1$ clock cycles which amounts to a single FC. The number of clock cycles for both the proposed structures are significantly reduced over the DA_{0-3}. For instance, a 32^{nd} order filter with 8^{th} order DA base unit for the proposed structures would amount FC to 21 clock cycles.

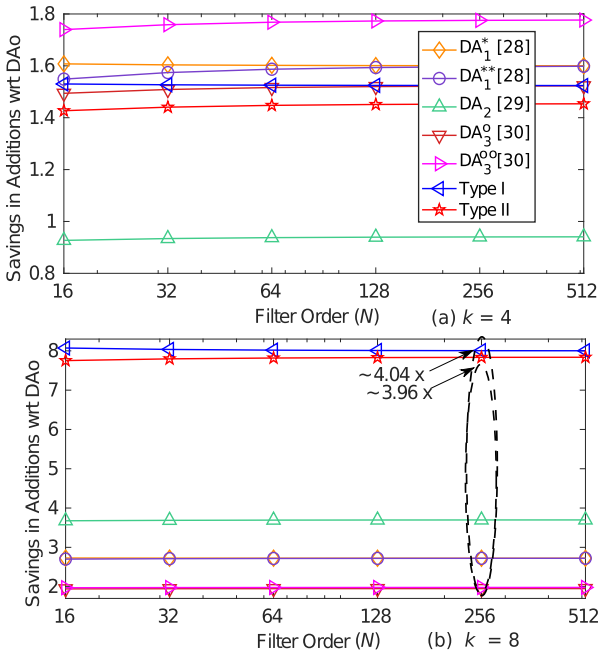

FIGURE 7. Savings in ADD per FC with respect to DA_0 for the proposed and DA_{1-3} designs for (a) $k = 4$ and (b) $k = 8$.

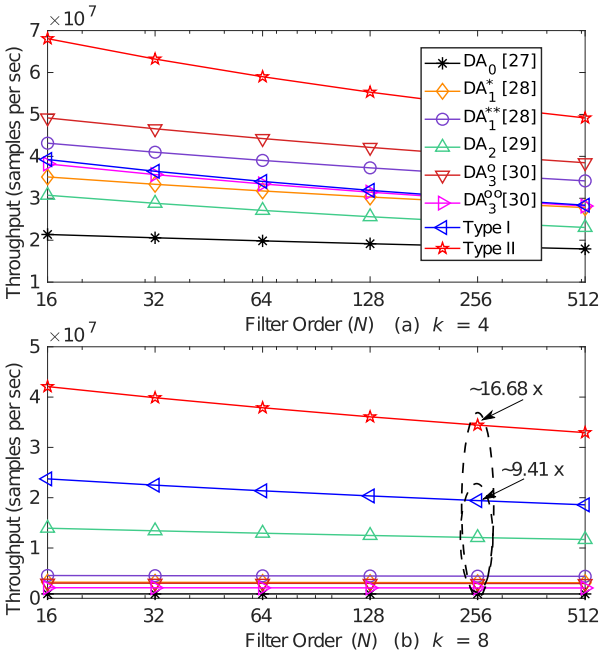

FIGURE 8. Throughput plots for the proposed and DA_{0-3} designs for (a) $k = 4$ and (b) $k = 8$.

Since the critical paths of different designs also depend on the LUT access time, which itself is a function of the DA base unit. Hence, it is difficult to compare the different designs. For a fair comparison, we define the throughput of an adaptive filter as the number of signal samples processed by an adaptive filter per second. It includes both the critical path delay and number of clock cycles parameters in its definition, as per

$$\text{Throughput} = \frac{1}{\text{Critical Path Delay} \times \text{FC}} \tag{42}$$

Depending upon the order of the DA base unit, both the proposed structures can be made to provide higher throughput as compared to DA_{0-3}. For higher-order of DA base unit, the DA_{0-3} suffer from severe throughput bottleneck. In contrast, the proposed structures are more advantageous for higher-order DA base unit. Interestingly, when 8^{th} order DA base unit is used, the throughput of both the structures is higher as compared to the DA_{0-3}. The improvement in throughput would be more when the DA base unit order is large. For example, a 32^{nd} order filter with 8^{th} order DA base unit using 90 nm TSMC standard library cells, the Type I structure offers $9.41\times$ and Type II structure offers $16.68\times$ improvements in throughput over $DA_3^{\circ\circ}$, as shown in Fig. 8. Although the improvement in throughput is achievable with high $k(= N/m)$, the word-size required for the FP representation of E-LUT and O-LUT proportionally increases. For a fixed m, the increase in DA base unit k also increases the filter order N. As a consequence, the step-size for FP implementation has to be reduced for the larger DA base unit and filter orders to design the DA based LMS ADF. Otherwise, it would increase the quantization error ξ_q and causes significant perturbations in the steady-state.

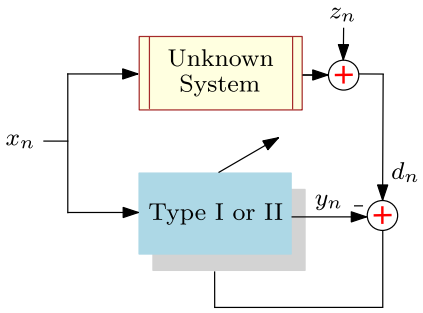

FIGURE 9. Block diagram of system identification using Type I or II design.

From the previous discussion, it is clear that Type II outperforms in throughput among all the designs, while Type I offers slightly lower throughput. As stated, Type I and Type II structures are based on TC DA and OBC DA, thus it is fair to compare their convergence performance. For FP simulation, the proposed Type I and Type II structures of LMS ADF are used for a system identification problem with $(X_f^x, X_i^x) = (8, 0)$ and $(X_f^w, X_i^w) = (X_w^b, X_i^b) = (8, 0)$, as shown in Fig. 9. The unknown system is 10^{th} order band-pass filter [42] with an impulse response given by

$$h_n = \frac{\sin(w_H(n-4.5))}{\pi((n-4.5))} - \frac{\sin(w_L(n-4.5))}{\pi((n-4.5))}$$
$$\text{for } n = 0, 1, 2, \ldots, 9, \quad \text{otherwise } h_n = 0 \tag{43}$$

where w_H and w_L are the high and low cut-off frequencies of the passband which are set to $w_H = 0.7\pi$ and $w_L = 0.3\pi$ respectively. We have considered 16^{th}, 32^{nd} and 64^{th} orders filter with step size as half of the bound given by (12) i.e., $\mu = 1/N\sigma_x^2$ which also satisfies the fixed-point quantization model. The unknown system and adaptive filter are fed with the Gaussian random input x_n of zero mean and unit variance $(\sigma_x^2 = 1)$. The output of the unknown system is contaminated with white Gaussian noise z_n of strength -60 dB. The MSE learning curve of Type I and Type II designs averaged over 100 independent runs are illustrated in Fig. 10(a) and 10(b) respectively. It is clear that as the filter order increases, the convergence rate slows down and increases the steady-state MSE in all the cases. The Type I design offers a fast convergence than Type II design. This is because the step-size in Type II design is $2\times$ than Type I design, therefore, its time constant is reduced by half as compared to Type I design, as per (16). It is interesting to note that both the proposed designs reach the similar steady-state MSE (minimum MSE) for a given N, as per (9). Thus, it is clear that the Type II structure provides high throughput performance at the expense of slow convergence, while Type I offers fast convergence at the expense of low throughput performance for almost similar steady-state mean square error.

B. IMPLEMENTATION RESULTS

We have coded both the proposed and DA_{0-3} designs in Verilog HDL to implement 16^{th}, 32^{nd} and 64^{th} LMS ADFs

TABLE 2. ASIC synthesis and FPGA implementation of different DA based architectures using 4th and 8th order DA base units.

	Design	ASIC Synthesis Using TSMC 90 nm CMOS Library by Cadence RTL Compiler									FPGA Implementation Using Xilinx FPGA Virtex (690t)					
		$N=16$			$N=32$			$N=64$			$N=16$		$N=32$		$N=64$	
		MSP (ns)	Area (μm^2)	Power (mW)	MSP (ns)	Area (μm^2)	Power (mW)	MSP (ns)	Area (μm^2)	Power (mW)	SLUT (×1000)	FF (×1000)	SLUT (×1000)	FF (×1000)	SLUT (×1000)	FF (×1000)
$k=4$	DA_0 [27]	24.28	47938	31.55	26.91	95188	62.08	29.15	186571	123.26	6.05	1.65	11.04	3.21	21.12	6.21
	DA_1^* [28]	16.29	38121	26.34	18.55	78124	52.53	20.67	154548	53.12	4.37	1.41	8.45	2.72	15.85	5.48
	DA_1^{**} [28]	18.95	36782	22.96	20.46	72562	42.86	23.06	141461	83.56	3.81	1.51	7.56	2.65	12.81	5.09
	DA_2 [29]	30.45	29397	15.25	33.07	59934	30.31	35.88	113824	56.89	2.85	1.38	5.76	2.58	11.35	4.94
	DA_3° [30]	19.26	29658	14.63	21.42	58583	27.87	23.37	111435	52.78	2.96	1.43	5.96	2.69	11.39	5.51
	$DA_3^{\circ\circ}$ [30]	23.57	31087	16.58	26.61	61273	32.84	29.34	118862	59.25	3.10	1.50	6.42	2.84	12.42	5.26
	Type I	14.33	39225	27.45	15.81	78105	53.73	17.45	152346	102.15	4.64	1.55	9.21	2.86	17.97	5.69
	Type II	11.25	37088	24.32	11.97	73424	47.65	13.24	144524	87.58	3.96	1.49	7.87	2.79	15.51	5.35
		$N=32$			$N=64$			$N=128$			$N=32$		$N=64$		$N=128$	
†$k=8$	DA_0 [27]	32.13	94351	59.22	34.02	185938	121.68	36.83	355156	228.05	9.89	3.07	20.49	6.05	33.83	11.88
	DA_1^* [28]	20.87	77452	51.09	23.05	153907	49.34	25.89	295678	97.56	8.25	2.69	15.24	5.18	27.46	10.24
	DA_1^{**} [28]	22.68	71856	41.87	24.96	140482	80.11	27.15	279287	152.07	7.43	2.89	12.32	5.04	22.14	8.42
	DA_2 [29]	35.67	58384	29.34	38.18	112939	54.76	40.54	215793	105.34	5.48	2.56	10.94	4.85	20.63	9.18
	DA_3° [30]	21.63	57863	26.78	23.56	110562	50.32	26.12	217673	96.68	5.81	2.65	10.95	5.25	21.34	9.68
	$DA_3^{\circ\circ}$ [30]	26.34	59568	31.27	29.72	117823	57.46	33.21	290679	110.76	5.91	2.72	11.82	5.18	22.56	10.42
	Type I	16.13	76617	52.28	17.42	152245	99.53	19.21	234709	194.36	8.81	2.81	17.56	5.42	28.95	10.45
	Type II	12.84	72326	46.23	13.68	143157	84.21	14.77	277356	165.95	7.63	2.68	15.23	5.25	25.62	9.46

† ASIC and FPGA implementation of all the design for large k are carried out by splitting k into smaller k' such that $k = k' \times l$, where l is the number of k'. For instance, all the designs with $k = 8$ are implemented using $k' = 4$ and $l = 2$ so as to avoid the large sized LUTs and efficient implementation.

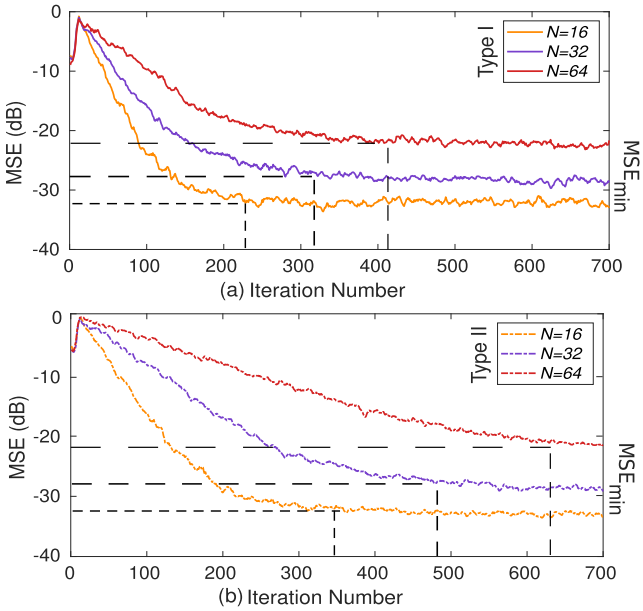

FIGURE 10. Convergence curves of LMS ADF based on (a) Type I (b) Type II.

using 4th order DA base unit; and 32nd 64th and 128th order LMS ADFs using 8th order DA base unit. The reason for using higher k for LMS ADF implementation is to have a fair comparison with other designs. This is because the proposed architectures provide high throughput for large k values. The wordlength of input samples and filter coefficients are assumed to be 8-bits. For synthesis purposes, we have considered the register file as LUT for the proposed structures as compared to SRAM and latch-based memories as it provides low access time for high throughput implementation. ASIC synthesis was performed using TSMC 90 nm CMOS Library by Cadence RTL Compiler. The Verilog file was completely flattened to the logic gates at gate-level implementation by the tool. Subsequently, the Verilog file from the RTL Compiler was then used to perform place and route by the Cadence Design Encounter tool. The random test vectors were taken for the generation of switching activity files with full timing and parasitic parameters. In addition, low-driving strength standard cells were employed to avoid routing congestion during the place and route. To avoid the setup and/or hold violations, extra buffers were introduced by the tool with a slight increase in area and power consumption. The estimated area, power (at 100 MHz), and MSP by the tool are listed in Table 2. As expected, the area and power of Type I and Type II structures are increased due to more computational cost involved. This stems from the fact that the overhead in LUT pre-decomposition. For instance, the proposed Type I structure has one extra adder, a few more multiplexers, and registers as compared to DA_1^*. This is valid for the Type II structure and all the filter orders and DA base unit orders. While the hardware cost DA_{2-3} designs are relatively lower than proposed structures due to the use of OBC DA. For example, a 32nd order filter with 8th order DA base unit using Type I structure provides 38.76% less MSP, occupies 28.62% more area and consumes 67.18% more power as compared to $DA_3^{\circ\circ}$, whereas Type II structure provides 51.25% less MSP, occupies 21.42% more area and consumes 47.84% more power compared to $DA_3^{\circ\circ}$.

The proposed structures are also implemented on a Xilinx FPGA Virtex (690t). The Verilog file is imported into the Xilinx Vivado tool, translating them into a netlist in Xilinx netlist format (.xnf). The simulator then performed functional verification through test vectors. The mapping is performed to translate the netlist on the available resources of the FPGA at 100 MHz clock frequency. After this step, the place and route process was done at specific locations of the FPGA device with the available routing resources. Subsequently, the delay information corresponding to interconnections is used to obtain a more accurate netlist through timing analysis (back-annotation). A bitstream file is generated for each design, which is then transferred to the Xilinx Virtex (690t). The logic utilization of the proposed and DA_{0-3} in terms of slice-LUTs (SLUT) and flip-flops (FF). The Type I and Type II structures are a slightly higher numbers of SLUT

and FF as compared to DA_1^* and DA_1^{**} respectively. While they provide significant throughput improvements compared to $DA_3^{\circ\circ}$ at the expense of more logic utilization. For example, a 32nd order filter with 8th order DA base unit using Type I structure utilizes 49.06% more SLUTs and 3.31% more FFs over $DA_3^{\circ\circ}$, whereas Type II structure utilizes 29.10% more SLUTs and 1.47% less FFs as compared to $DA_3^{\circ\circ}$.

VI. CONCLUSION AND FUTURE WORK

In this paper, the detailed architectural analysis and implementation of two new high throughput architectures of non-pipelined LMS ADF have been presented. The proposed design methodology has resulted in less number of clock cycles, less LUT access time, a small sampling period, and less number of additions for both proposed architectures. An efficient fixed-point quantization model for the evaluation of proposed structures is also presented. Unlike existing non-pipelined designs [27]–[30], the proposed structures offer very high throughput performance, especially with large order DA base unit. The second structure provides higher throughput than the first structure at the expense of a slow convergence rate with almost the same steady-state mean square error. It is found that 256th order filter with 8th order DA base unit using Type I structure provides 9.41× higher throughput whereas the Type I structure provides 16.68× higher throughput than [30]. ASIC and FPGA implementation results showed that 32nd order filter with 8th order DA base unit using Type I structure achieves 38.76% less minimum sampling period (MSP), occupies 28.62% more area, consumes 67.18% more power, utilizes 49.06% more slice LUTs and 3.31% more flip-flops (FFs), whereas Type II structure achieves 51.25% less MSP, occupies 21.42% more area, consumes 47.84% more power, utilizes 29.10% more slice LUTs and 1.47% less FFs as compared to [30].

This paper is primarily concentrated on high throughput fixed-point implementation of DA based non-pipelined LMS adaptive filters with its effect on the error performance. This may also be extended for future research with approximate DA [43] as it relaxes algorithm precision constraints. Hence, there will be a reduction in the chip area as well as the power consumption of adaptive filters with simultaneous increase in the throughput performance. Furthermore, this study focused on the improvement of throughput performance of LMS adaptive filter at the expense of area, power, logic resources and convergence rate. LMS variants such as NLMS algorithm can provide faster convergence than standalone LMS. Thus, it will be pertinent to take a similar methodology for DA based non-pipelined NLMS adaptive filter. [6] that can result in improved error performance.

ACKNOWLEDGMENT

The authors would like to thank all the reviewers for their valuable comments and suggestions, which immensely helped to improve the quality of manuscript.

REFERENCES

[1] S. Haykin, *Adaptive Filter Theory*, 3rd ed. Upper Saddle River, NJ, USA: Prentice-Hall, 1996.
[2] A. H. Sayed, *Fundamentals of Adaptive Filtering*. Hoboken, NJ, USA: Wiley, 2003.
[3] K. K. Parhi, *VLSI Digital Signal Processing Systems: Design and Implementation*. Hoboken, NJ, USA: Wiley, 2007.
[4] K. B. Widrow, J. McCool, M. G. Larimore, and C. R. Johnson, "Stationary and nonstationary learning characteristics of the LMS adaptive filter," in *Aspects of Signal Processing*. Springer, 1977, pp. 355–393.
[5] B.-E. Jun, D.-J. Park, and Y.-W. Kim, "Convergence analysis of sign-sign LMS algorithm for adaptive filters with correlated Gaussian data," in *Proc. Int. Conf. Acoust., Speech, Signal Process.*, vol. 2, 1995, pp. 1380–1383.
[6] N. Bershad, "Analysis of the normalized LMS algorithm with Gaussian inputs," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. ASSP-34, no. 4, pp. 793–806, Aug. 1986.
[7] J. Cioffi and T. Kailath, "Fast, recursive-least-squares transversal filters for adaptive filtering," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. ASSP-32, no. 2, pp. 304–337, Apr. 1984.
[8] K. Ozeki and T. Umeda, "An adaptive filtering algorithm using an orthogonal projection to an affine subspace and its properties," *Electron. Commun. Jpn. I, Commun.*, vol. 67, no. 5, pp. 19–27, 1984.
[9] S. Kalluri and G. R. Arce, "A general class of nonlinear normalized adaptive filtering algorithms," *IEEE Trans. Signal Process.*, vol. 47, no. 8, pp. 2262–2272, Aug. 1999.
[10] S. L. Gay, "The fast affine projection algorithm," in *Acoustic Signal Processing for Telecommunication*. Springer, 2000, pp. 23–45.
[11] D. Slock and T. Kailath, "Numerically stable fast transversal filters for recursive least-squares adaptive filtering," in *Numerical Linear Algebra, Digital Signal Processing and Parallel Algorithms*. Springer, 1991, pp. 605–615.
[12] B. Parhami, "Computing with logarithmic number system arithmetic: Implementation methods and performance benefits," *Comput. Electr. Eng.*, vol. 87, Oct. 2020, Art. no. 106800.
[13] F. Albu, J. Kadlec, N. Coleman, and A. Fagan, "Pipelined implementations of the *a priori* error-feedback LSL algorithm using logarithmic arithmetic," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, vol. 3, May 2002, p. 2681.
[14] M. M. Chansarkar and U. B. Desai, "A robust recursive least squares algorithm," *IEEE Trans. Signal Process.*, vol. 45, no. 7, pp. 1726–1735, Jul. 1997.
[15] E. Walach and B. Widrow, "The least mean fourth (LMF) adaptive algorithm and its family," *IEEE Trans. Inf. Theory*, vol. IT-30, no. 2, pp. 275–283, Mar. 1984.
[16] A. Feuer and E. Weinstein, "Convergence analysis of LMS filters with uncorrelated Gaussian data," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. ASSP-33, no. 1, pp. 222–230, Feb. 1985.
[17] J. E. Mazo, "On the independence theory of equalizer convergence," *Bell Syst. Tech. J.*, vol. 58, no. 5, pp. 963–993, May/Jun. 1979.
[18] L. Zhoufan, L. Dan, X. Xinlong, and Z. Jianqiu, "New normalized LMS adaptive filter with a variable regularization factor," *J. Syst. Eng. Electron.*, vol. 30, no. 2, pp. 259–269, 2019.
[19] 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.
[20] N. R. Shanbhag and K. Parhi, *Pipelined Adaptive Digital Filters*, vol. 274. Springer, 2012.
[21] B. Hoefflinger, "ITRS: The international technology roadmap for semiconductors," in *Chips*. Springer, 2011, pp. 161–174.
[22] J. Choi, H. You, C. Kim, H. Young Yeom, and Y. Kim, "Comparing unified, pinned, and host/device memory allocations for memory-intensive workloads on Tegra SoC," *Concurrency Comput., Pract. Exp.*, vol. 33, no. 4, 2021, Art. no. e6018.
[23] P. K. Meher, "New approach to look-up-table design and memory-based realization of FIR digital filter," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 57, no. 3, pp. 592–603, Mar. 2010.
[24] A. Peled and B. Liu, "A new hardware realization of digital filters," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. ASSP-22, no. 6, pp. 456–462, Dec. 1974.
[25] A. Croisier, D. Esteban, M. Levilion, and V. Riso, "Digital filter for PCM encoded signals," U.S. Patent 3 777 130, Dec. 4, 1973.
[26] S. A. White, "Applications of distributed arithmetic to digital signal processing: A tutorial review," *IEEE ASSP Mag.*, vol. 6, no. 3, pp. 4–19, Jul. 1989.

[27] D. J. Allred, H. Yoo, V. Krishnan, W. Huang, and D. V. Anderson, "LMS adaptive filters using distributed arithmetic for high throughput," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 52, no. 7, pp. 1327–1337, Jul. 2005.
[28] R. Guo and L. S. DeBrunner, "Two high-performance adaptive filter implementation schemes using distributed arithmetic," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 58, no. 9, pp. 600–604, Sep. 2011.
[29] M. S. Prakash and R. A. Shaik, "Low-area and high-throughput architecture for an adaptive filter using distributed arithmetic," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 60, no. 11, pp. 781–785, Nov. 2013.
[30] S. Ahmad, S. G. Khawaja, N. Amjad, and M. Usman, "A novel multiplierless LMS adaptive filter design based on offset binary coded distributed arithmetic," *IEEE Access*, vol. 9, pp. 78138–78152, 2021.
[31] M. T. Khan, S. R. Ahamed, and F. Brewer, "Low complexity and critical path based VLSI architecture for LMS adaptive filter using distributed arithmetic," in *Proc. 30th Int. Conf. VLSI Design 16th Int. Conf. Embedded Syst. (VLSID)*, Jan. 2017, pp. 127–132.
[32] M. T. Khan and S. R. Ahamed, "Area and power efficient VLSI architecture of distributed arithmetic based LMS adaptive filter," in *Proc. 31st Int. Conf. VLSI Design 17th Int. Conf. Embedded Syst. (VLSID)*, Jan. 2018, pp. 283–288.
[33] Y. Tsunekawa, K. Takahashi, S. Toyoda, and M. Miura, "High-performance VLSI architecture of multiplierless LMS adaptive filters using distributed arithmetic," *Electron. Commun. Jpn. III, Fundam. Electron. Sci.*, vol. 84, no. 5, pp. 1–12, 2001.
[34] R. Guo and L. S. DeBrunner, "A novel adaptive filter implementation scheme using distributed arithmetic," in *Proc. Conf. Rec. 45th Asilomar Conf. Signals, Syst. Comput. (ASILOMAR)*, Nov. 2011, pp. 160–164.
[35] P. K. Meher and S. Y. Park, "High-throughput pipelined realization of adaptive FIR filter based on distributed arithmetic," in *Proc. IEEE/IFIP 19th Int. Conf. VLSI Syst.-Chip*, Oct. 2011, pp. 428–433.
[36] S. Y. Park and P. K. Meher, "Low-power, high-throughput, and low-area adaptive FIR filter based on distributed arithmetic," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 60, no. 6, pp. 346–350, Jun. 2013.
[37] M. T. Khan, R. A. Shaik, and S. P. Matcha, "Improved convergent distributed arithmetic based low complexity pipelined least-mean-square filter," *IET Circuits, Devices Syst.*, vol. 12, no. 6, pp. 792–801, 2018.
[38] M. T. Khan and R. A. Shaik, "Optimal complexity architectures for pipelined distributed arithmetic-based LMS adaptive filter," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 66, no. 2, pp. 630–642, Feb. 2019.
[39] M. T. Khan and R. A. Shaik, "High-performance VLSI architecture of DLMS adaptive filter for fast-convergence and low-MSE," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 69, no. 4, pp. 2106–2110, Apr. 2022.
[40] K. Takahashi, Y. Tsunekawa, N. Tayama, and K. Seki, "Analysis of the convergence condition of LMS adaptive digital filter using distributed arithmetic," *IEICE Trans. Fundam. Electron., Commun. Comput. Sci.*, vol. 85, no. 6, pp. 1249–1256, Jan. 2002.
[41] C. Caraiscos and B. Liu, "A roundoff error analysis of the LMS adaptive algorithm," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. ASSP-32, no. 1, pp. 34–41, Feb. 1984.
[42] 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.
[43] S. Venkatachalam and S.-B. Ko, "Approximate sum-of-products designs based on distributed arithmetic," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 26, no. 8, pp. 1604–1608, Aug. 2018.

MOHD. TASLEEM KHAN received the B.Tech. degree in electronics from the Zakir Hussain College of Engineering and Technology, Aligarh Muslim University, Aligarh, India, in 2013, and the Ph.D. degree in VLSI from the Indian Institute of Technology Guwahati, India, in 2019. He was a Principal Engineer at Taiwan Semiconductor Manufacturing Company (TSMC), Hsinchu, Taiwan. He worked as an Assistant Professor with the Department of Electronics Engineering, Indian Institute of Technology Dhanbad, India. He is currently working as a Postdoctoral Researcher at Linköping University, Sweden. His research and teaching interests include study of algorithms and architectures for VLSI implementation of signal processing, communication, machine learning, and artificial intelligence applications.

MOHAMMED A. ALHARTOMI (Member, IEEE) received the Ph.D. degree in electronic and electrical engineering from Leeds University, U.K., in 2016. He is currently an Assistant Professor with the Department of Electrical Engineering, Tabuk University. His research interests include signal processing, OW systems design, and VLC.

SAEED ALZAHRANI (Member, IEEE) received the B.S. degree from King Abdulaziz University, Jeddah, Saudi Arabia, the M.S. degree from the University of Colorado at Boulder, and the Ph.D. degree from The Ohio State University, all in electrical engineering. He was with the Microelectronics Research Laboratory, University of Colorado at Boulder, from 2012 to 2014, involved in developing tunable ferroelectric-based filters, VCOs, amplifiers, and antennas. From 2014 to 2019, he was with the ElectroScience Laboratory, The Ohio State University, focusing on developing design techniques for wide TR VCOs at mm-wave frequency band. He is currently working as an Assistant Professor with the Electrical Engineering Department, University of Tabuk, Saudi Arabia. His research interests include addresses design and technological challenges related to RF/millimeter-wave and mixed-signal integrated circuits and systems for emerging technologies, including communication, automotive, and biomedical applications.

RAFI AHAMED SHAIK (Member, IEEE) received the B.Tech. and M.Tech. degrees in electronics and communication engineering from Sri Venkateswara University, Tirupati, India, in 1991 and 1993, respectively, and the Ph.D. degree from the Indian Institute of Technology Kharagpur, India, in 2008. He is currently a Professor with the Department of Electronics and Electrical Engineering, Indian Institute of Technology Guwahati, India. He was a Faculty Member of the Deccan College of Engineering and Technology, Hyderabad, India, from 1993 to 1995, and the Bapatla Engineering College, Bapatla, India, from 1995 to 2003. His teaching and research interests include digital and adaptive signal processing, biomedical signal processing, and VLSI signal processing.

RUWAYBIH ALSULAMI (Member, IEEE) was born in Mecca, Saudi Arabia. He received the B.S. degree in electrical and computer engineering from the University of Colorado at Boulder, Boulder, CO, USA, in 2011, the M.S. degree in electrical engineering from the University of Colorado at Colorado Springs, Colorado Springs, CO, USA, in 2013, and the Ph.D. degrees in electrical and computer engineering from The Ohio State University, Columbus, OH, USA, in 2020 and 2021, respectively. He is currently an Assistant Professor in electrical engineering with Umm Al-Qura University, Mecca. His current research interests include circuit design, such as antennas, filters, multiband power amplifiers, PA linearization, and measurements of nonlinear microwave devices and circuits.