2017 30th International Conference on VLSI Design and 2017 16th International Conference on Embedded Systems

Low Complexity and Critical Path based VLSI Architecture for LMS Adaptive Filter using Distributed Arithmetic

Mphd Tasleem Khan
Electronics and Electrical Engineering
IIT Guwahati
Guwahati, India
Email: tasleem@iitg.ernet.in

Shaik Rafi Ahamed
Electronics and Electrical Engineering
IIT Guwahati
Guwahati, India
Email: rafiahamed@iitg.ernet.in

Forrest Brewer
Electrical and Computer Engineering
UC Santa Barbara
Santa Barbara, USA
Email: forrest@ece.ucsb.edu

***Abstract*—This paper presents a new architecture for distributed arithmetic (DA) based Least Mean Square (LMS) adaptive filter with low hardware complexity and critical path. It is well known that for DA based adaptive filter, the throughput depends on critical path and number of clock cycles to produce the output. In the proposed technique, we maintained the same number of clock cycles using multiplexed look-up tables (LUTs) which reduces the hardware complexity and critical path compared to best existing scheme. For instance, the hardware complexity can be lowered down by $\alpha.N$, whereas the critical path can be reduced by $T_A + T_M$, with α, N, T_A and T_M being the number of reduced hardware elements, number of filter taps, adder and multiplexer computational delays, respectively. Synthesis result shows that for almost similar area and power performance, the proposed scheme achieves a gain of 27.6% due to clock speedup which results in more throughput and power can be lowered compared to best existing scheme.**

***Index Terms*—Adaptive filter (ADF), distributed arithmetic (DA), least mean square (LMS), offset binary coding (OBC).**

I. Introduction

Adaptive filtering finds extensive application in digital signal processing (DSP) applications such as noise and echo cancellation, channel equalization, system identification, channel estimation etc. [1]. The finite-impulse-response (FIR) filter are generally implemented using one or more multiply and accumulate (MAC) units based on the filter order. The filter weights are updated according to the well known Widrow Hoff criteria known as least mean square (LMS) algorithm. The output of MAC based FIR filter is weighted sum of input samples and filter weights whose complexity grows linearly with filter order mainly by multipliers in MAC units.

With the advancements in semiconductor technology the memories are getting more cheaper, faster and efficient. It results in less memory access time which can improve the efficiency of memory based DSP architecture [2]. In the past, an efficient multiplierless approach known as distributed arithmetic (DA) [3] to transform the MAC based FIR architecture into a memory and shift accumulate unit. The DA based filter structure is superior compared to MAC based structure due to regularity and availability of efficient memories. In recent past, several works have been reported [4]–[8] to improve the performance of tapped delay line filter using DA. Allred $et\ al.$ [4] proposed LMS adaptive filters using two LUTs for filtering and weight updating operation which results in low throughput due to memory structure which require update of each address location. However, it involve less number of hardware components due to large memory size. Guo $et\ al.$ [5], [6] employed single memory which is responsible for both filtering and weight updating operation. This scheme overcomes the issue of increased memory size by simultaneously improving the throughput. However, the number of hardware components are increased due to complex logic involved. To overcome throughput bottleneck of [4], Surya $et\ al.$ [8] used offset binary combinations for both input samples and filter weights and stored them in separate memories. It results in significant improvement of throughput due to decomposition of memory into two small memories using a multiplexer. Further, it involves the reading of contents from successive address locations of memory to eliminate the oldest sample and again the addition or subtraction of recent sample based on even or odd memory locations respectively. This would increase the update time of memory due to two adders and physical address rotation circuitry. Further, a new pipelined approach [7] for DA based adaptive filter using pipelined version of LMS known as delayed LMS (DLMS) which results into convergence performance degradation. Recently, a multiplier based low complexity and small critical path for LMS adaptive filter is proposed in [9]. However, physical multipliers occupies significant large chip area and limits the performance of the system in terms of both area and throughput.

In this paper, a new architecture is presented using multiplexed memories to maintain the same number of clock cycles as that of [8] based on the framework of [4] which results in low complexity and critical path compared to best existing scheme. In addition, we have modified the weight update algorithm for the proposed technique to achieve significant clock speedup.

The rest of this paper is organized as follows. In Section-I, we first present the brief about LMS algorithm and DA

2380-6923/16 $31.00 © 2016 IEEE
DOI 10.1109/VLSID.2017.16

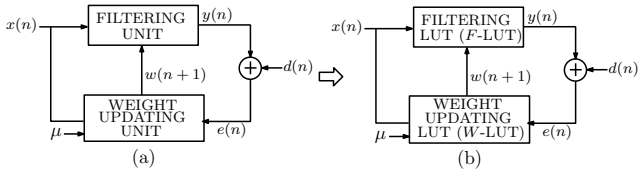

Fig. 1: Transformation using DA (a) Conventional LMS adaptive FIR filter (b) DA based LMS adaptive FIR filter.

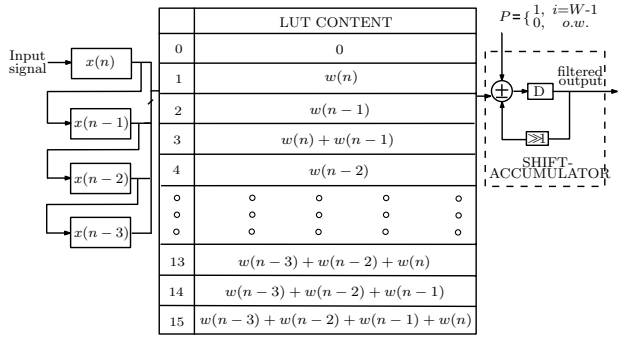

Fig. 2: Contents of F-LUT for 4-th order FIR adaptive filter.

transformation technique. Next, we present the proposed architecture with weight updating scheme in Section-III. The performance comparison of proposed scheme with memory based existing schemes in section IV. Conclusions are given in section V.

Consider a N-th order FIR filter that process input samples $x[n] = [x(n), x(n-1), ..., x(n-N+1)]^T$ with weights denoted by w_j, where $j \in [0, N-1]$ and N being the filter order. The output signal $y[n]$ at any time instant n can be written as

$$y[n] = \sum_{j=0}^{N-1} w_j x[n-j] \tag{1}$$

The output signal in (1) when subtracted by desired signal $d(n)$ generates the error signal $e(n)$ as

$$e(n) = d(n) - y(n) \tag{2}$$

Using error signal $e(n)$, the filter weights are updated using least mean square (LMS) criterion as

$$w_j(n+1) = w_j(n) + \mu e(n) x(n-j) \tag{3}$$

where, μ is convergence factor, which is generally a positive constant number and taken in negative powers of 2.

In order to apply DA transformation, the input signal $x(n)$ as shown in Fig. 1, need to be represented in twos complement form as

$$x(n-i) = x_i = -x_{i,0} + \sum_{j=0}^{B-1} x_{i,j} 2^{-j} \tag{4}$$

By substituting (4) into (1), we get

$$y[n] = \sum_{i=0}^{N-1} [-x_{i,0} + \sum_{j=0}^{B-1} x_{i,j} 2^{-j}] w_i$$

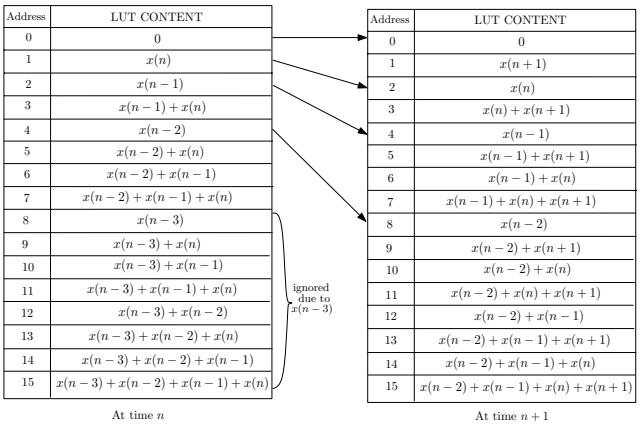

Fig. 3: Weight update scheme modified from [4] for conventional-DA.

Interchange the order of summation, we arrive at

$$y[n] = \sum_{j=0}^{B-1} c_{B-1-j} 2^{-j} \tag{5}$$

where, $c_{B-1-j} = \sum_{i=0}^{N-1} w_i x_{i,B-1-j}$ and B is input signal wordlength. It can be observed from (5) that actual nature which was word-serial is transformed to bit-serial results in a memory and a shift accumulate unit. The number of memory locations required to store c_{B-1-j} term are 2^N for $i \in [0, N-1]$ since $x_{i,B-1-j} \in \{0, 1\}$ and $j \in [0, B-1]$. In Fig. 2, the contents of filtering look-up table (F-LUT) are shown for 4-th order filter followed by a shift accumulate operation to produce filter output signal $y[n]$. Note that, the least significant bit (LSB) of input samples are used as address bits for F-LUT. The contents of F-LUT at time instant n can be written as

$$\begin{aligned} F_{i_F}^n &= \langle w[n - i_F].i_F \rangle \\ i_F &= \phi([a_{N-1,iF}, a_{N-2,i_F}, ...a_{0,i_F}]^T) \end{aligned} \tag{6}$$

where, $\langle . \rangle$ represents inner product operator, i_F is address index of F-LUT and ϕ is a mapping function for i_F with $a_{i,iF} \in \{0, 1\}$ and $i \in \{0, N-1\}$.

II. Proposed DA-based architecture for adaptive filter

In previous section, it was shown that DA based FIR filtering requires a memory and shift accumulate units. In this section, the design of proposed scheme is presented based on the framework of [4]. It enables the parallel update of F-LUT and weight updating LUT (W-LUT). Moreover, it also involves smaller LUTs that would improve the processing speed of the proposed filter.

A. Proposed Approach

The filter weights are stored as binary combinations in F-LUT followed by a shift accumulate unit to perform filtering

	ignored	$n+1$			ignored	$n+2$			ignored	$n+3$			ignored	$n+4$		
AL	x_3	x_2	x_1	x_0	x_3	x_2	x_1	x_0	x_3	x_2	x_1	x_0	x_3	x_2	x_1	x_0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	1	0	0	0	0	1	0	0	0	0	1
2	0	0	1	0	0	0	0	1	0	1	0	0	0	0	1	0
3	0	0	1	1	0	1	0	1	0	1	1	0	0	0	1	1
4	0	1	0	0	0	0	1	0	0	0	0	1	0	1	0	0
5	0	1	0	1	0	1	1	0	0	0	1	1	0	1	0	1
6	0	1	1	0	0	0	1	1	0	1	0	1	0	1	1	0
7	0	1	1	1	0	1	1	1	0	1	1	1	0	1	1	1

AL=Address Locations $x_i = x(n-i)$

Fig. 4: Data flow table (DFT) to explain the weight adaptation scheme for $N = 4$.

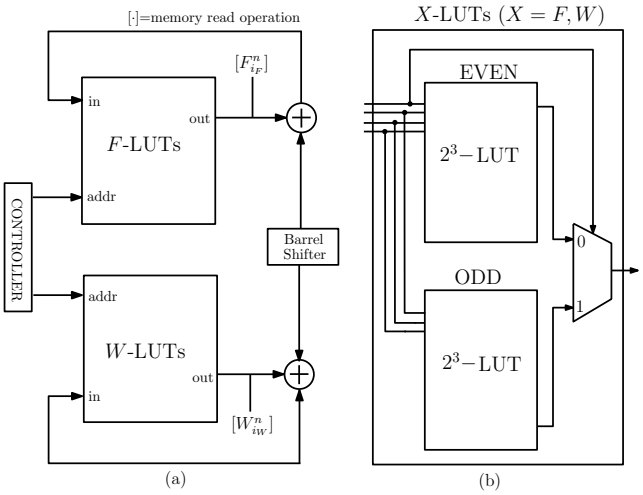

Fig. 5: (a) Top-level diagram of proposed architecture of DA based adaptive filter (b) Multiplexed LUTs for F-LUT and W-LUT.

operation on input samples. To update the filter weights the same binary combination of input samples $x[n]$ are stored in another look-up table referred as W-LUT. Fig. 3 shows the contents of W-LUT at time instant n with address index i_W and represented by $W^n_{i_W}$. Mathematically, it can be written as

$$W^n_{i_W} = \langle x[n - i_W].i_W \rangle \tag{7}$$
$$i_W = \psi([a_{N-1,i_W}, a_{N-2,i_W}, ...a_{0,i_W}]^T)$$

where $\langle . \rangle$ denotes the inner product operator, ψ is mapping function for address index i_W of W-LUT with $a_{i,i_W} \in \{0, 1\}$ and $i \in \{0, B-1\}$ At time $n+1$, the contents of memory changed as per

$$W^{n+1}_{i_W} = \langle x[n - (i_W - 1)].i_W \rangle \tag{8}$$

It means that only first $N-1$ past input sample combinations are involved in the update of W-LUT contents which means that only lower half memory locations are required for the update of even memory locations at time instant $n+1$ using address rotation circuitry [4]. However, to update the

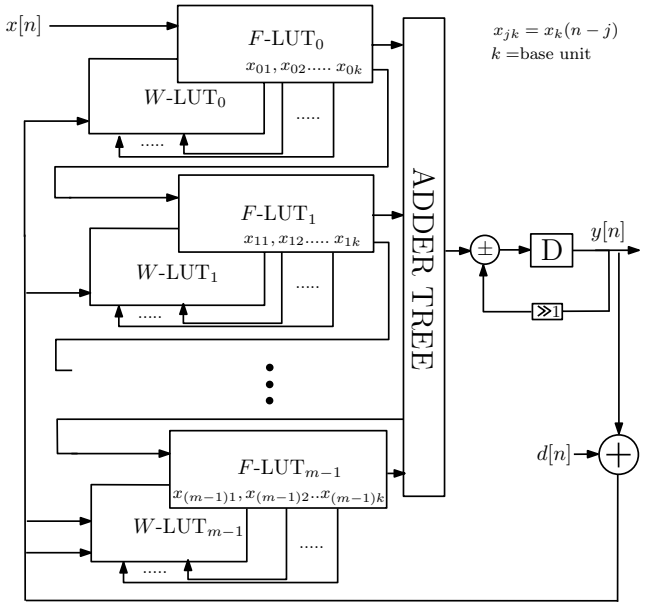

Fig. 6: Top-level diagram of the proposed adaptive filter for large order adaptive filter.

odd memory locations new sample $x(n+1)$ is added with contents of previous memory locations. Mathematically, it can be represented as follows

$$W^{n+1}_{i_W + \lfloor i_W/2 \rfloor + \lceil i_W/2 \rceil} = W^n_{i_W} \qquad i_W \% 2 == 0$$
$$W^{n+1}_{i_W} = W^n_{i_W - 1} + x(n+1) \qquad (o.w.) \tag{9}$$

When a new input sample $x(n+1)$ has arrived the address bits of W-LUT are circularly right shifted except most significant bit (MSB) to update of even locations as shown in Fig. 4. It can be noted that oldest sample $x(n-3)$ does not contribute in W-LUT update process. At time instant $n+1$, the content at 0-th address location is unchanged, the contents at 1st address locations goes to 2^1 memory locations and so on. In general, at time instant $n+k$, the memory content at i-th address locations moves to 2^k i-th memory locations. The remaining memory locations in k-th iteration can be updated using proposed LUT update algorithm (9).

Both W-LUT and F-LUT are updated using (3) however, due to memory architecture require the update of contents at every address locations according to

$$F^{n+1}_{i_F} = F^n_{i_F} + \mu.e(n).W^n_{i_W} \tag{10}$$

that is, F-LUT at $n+1$ time instant is updated by reading the same memory locations of F-LUT and W-LUT at time instant n by multiplying the term $\mu.e(n)$ using barrel shifter as shown in Fig. 5(a) finally storing the result back in same memory location of F-LUT. This process is repeated for all address locations until all the address locations of both memories gets updated. Further, it is clear that F-LUT takes 2^N clock cycles to update each memory location take one clock cycle to update. It can be avoided using proposed structure shown in Fig. 5(b)

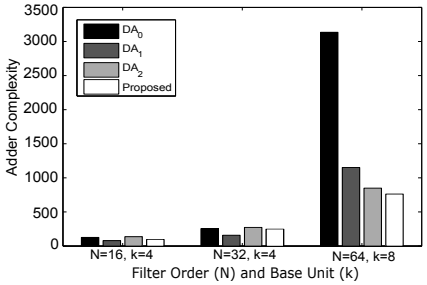

Fig. 8: Bar graph showing adder complexity.

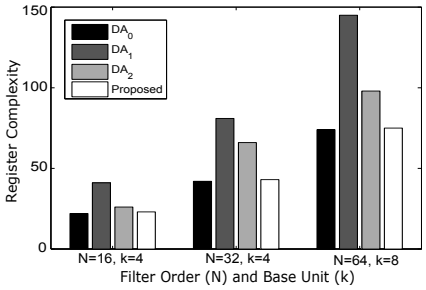

Fig. 9: Bar graph showing register complexity.

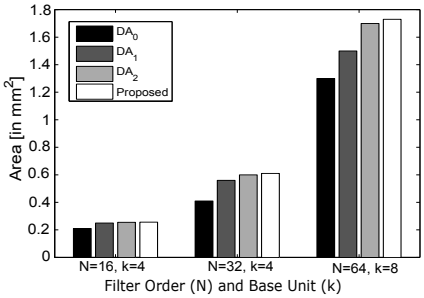

Fig. 10: Area comparison chart for proposed and various existing schemes.

1: **loop**
   $y[n] = \sum_{j=0}^{B-1} c_{B-1-j} 2^{-j}$
2: **for** $i_W = 0$ to $2^N - 1$ **do**
3: **if** $i_W \text{mod}(2) == 0$ **then**
   $W^{n+1}_{i_W + \lfloor i_W/2 \rfloor + \lceil i_W/2 \rceil} \leftarrow W^n_{i_W}$
4: **else**
   $W^{n+1}_{i_W} \leftarrow W^n_{i_W - 1} + x(n+1)$
5: **end if**
6: **end for**
7: $e[n] \leftarrow d[n] - y[n]$
8: **for** $i_W = 0$ to $2^N - 1$ **do**
   $F^{n+1}_{i_F} \leftarrow F^n_{i_F} + \mu e[n] \{W^n_{i_W + \lfloor i_W/2 \rfloor + \lceil i_W/2 \rceil}\}$
9: **end for**
10: **return** $y[n]$
11: $n \leftarrow n + 1$
12: **end loop**

Fig. 7: Algorithm explaining proposed scheme for DA based adaptive filter.

which multiplexed small F-LUTs and W-LUTs based on least significant bit of address lines. By doing so, the number of clock cycles of both look-up tables are reduced by $2^{N/2}$ similar to [8]. It can be noted from (9) that only one adder is involved for the update of memory locations compared to [8].

B. Design of large order ADF

The look-up table (LUT) grows exponentially with filter order and sets a upper limit on system throughput due its size. Therefore, it is important to reduce the size of memory in addition to multiplexing technique for both LUT to decompose the large filter order as shown in Fig. 6. It is because the resources are not optimally used which utilize divide and conquer approach for implementation. The decomposition of large filter order into sub-filters requires an adder tree to get final response $y[n]$ of the filter followed by a single shift accumulator. The output signal $y[n]$ is subtracted from desired signal $d[n]$ to generate error signal $e[n]$. Thr convergence factor μ is taken in powers of 2 to simplify the multiplication of μ and $e[n]$ in (3).

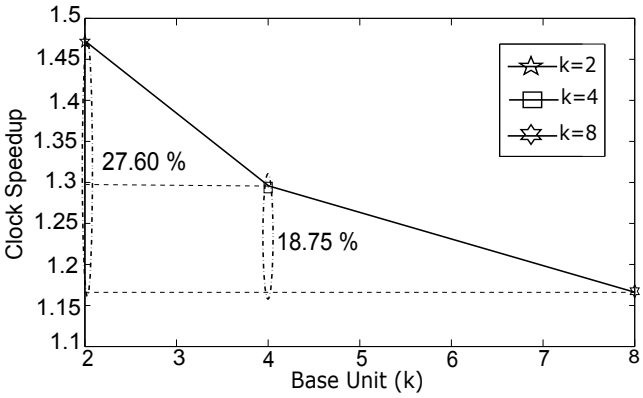

Fig. 11: Clock speedup of proposed scheme over DA_2 scheme for $k = 2, 4$ and 8.

III. Performance Comparison

In order to compare the results obtained from of proposed scheme to those of existing schemes [4], [5] and [8], we refer them as DA_0, DA_1 and DA_2, respectively. We have listed hardware and time complexity of various existing schemes in Table I. It is clear from Table I that the hardware complexity of proposed scheme is reduced DA_0 by some factor α. For better insight, we have shown complexity of adders and registers in Fig. 8 and Fig. 9, respectively. It is clear from Fig. 8 that for all values of N and k, the number of adders per clock cycle are always less compared to DA_0, DA_1 and DA_2 schemes. It can be noted that for $N = 32$ with $k = 4$ the adders are slightly increased compared to DA_2 scheme. However, for large filter order with more base unit, the savings in additions are significant. Further, the number of registers required in the proposed design are similar to DA_0 scheme. It is also clear from Fig. 9 that DA_0 and proposed scheme have similar register complexity and much less than that of DA_1 and DA_2 scheme. Throughput of a adaptive filter can also be defined [4] as

$$\text{Throughput} = \frac{1}{\text{Critical path x Number of clock cycles}} \quad (11)$$

It is clear from (11) that for the same number of clock cycles throughput of the adaptive filter is determined mainly

TABLE I: Time and Hardware Complexity of Various Existing Schemes.

Design	**Throughput**	**Adders** (per cycle)	**Registers**	**SH**	**Memory**
DA_0	$1/[m_0(T_R+(k-1).T_M+T_A)]$	$m.(2^{k-1}+2^k)+m.B-1$	$m.(1+k)+2$	m	$2m.(2^k-1)$
DA_1	$1/[m_1(T_R+T_A)]$	$m.(2^{k-1}+k)+m.B-1$	$m.(2+2k)+1$	$m.k$	$m.(2^k-1)$
DA_3	$1/[m_2(T_R+(k+1).T_M+2T_A)]$	$m.(2^{k/2+1}+2^{k/2+2}+2)+m.B+1$	$m.(4+k)+2$	m	$2m.2^{k-1}$
Proposed	$1/[m_2(T_R+k.T_M+T_A)]$	$m.(2^{k/2+1}+2^{k/2+2})+m.B-1$	$m.(1+k)+3$	m	$2m.(2^k-1)$

$N=mxk$, m_0=2^k+max(B,2^{k-1})+$\log_2 m$, m_1=2^{k-1}+$\log_2 N$+W+1, m_2=$2^{k/2}$+max(W,$2^{(k/2)-1}$)+$\log_2 m$+1, T_R = LUT access time, T_M = 2-to-1 multiplexer delay and T_A = adder delay. Note that [7]: based on delayed LMS (DLMS) and involve register look-up table.

TABLE II: Number of logic elements of various schemes for $N=16,32$ and 64 with $k=2,4$ and 8.

Design	**Filter Order**								
	16			32			64		
	$k=2$	$k=4$	$k=8$	$k=2$	$k=4$	$k=8$	$k=2$	$k=4$	$k=8$
DA_0	1309	915	798	2244	1429	1073	4140	2426	1624
DA_1	1198	845	588	2018	1289	812	3256	2134	1245
DA_2	986	712	467	1588	1023	602	2115	1357	905
Proposed	1367	945	815	2309	1478	1096	4211	2511	1689

TABLE III: Power consumption estimates of various schemes for $N=16,32$ and 64 with $k=2,4$ and 8.

Design	**Filter Order**								
	16			32			64		
	$k=2$	$k=4$	$k=8$	$k=2$	$k=4$	$k=8$	$k=2$	$k=4$	$k=8$
DA_0	14.43	15.83	11.87	24.54	19.82	14.94	34.54	37.15	24.95
DA_1	13.1	14.15	8.65	20.63	17.65	11.45	24.89	32.81	17.93
DA_2	10.34	12.23	8.34	11.48	14.63	9.56	18.45	21.56	15.87
Proposed	14.78	15.95	12.08	25.67	20.13	15.78	35.67	38.95	25.68

by its critical path. The time required for updating the filter weights is depends on critical path which is decided by the LUT update algorithm for weight adaptation. It can be noted that DA_2 scheme reduces the number of clock cycles but increases the critical path due to physical address rotation, adders and multiplexer due to multiplexed LUTs for reducing the LUT access time. It is clear from Table I that the proposed scheme has same number of clock cycles as DA_2 scheme. Further, the proposed scheme reduced the critical path by T_M+T_A, where T_M and T_A are computational delay times of 2-to-1 multiplexer and adder respectively. This reduction in clock period results an improvement in clock speed. We define the term clock speedup as figure of merit (FOM) as for the comparison with DA_2 scheme which is equal to $\frac{T_R+(k+1)T_M+2T_A}{T_R+kT_M+T_A}$. It depends on computational delays of LUT access, multiplexer and adder as well base unit (k) of larger filter. We have shown the clock speedup of proposed scheme for $k=2,4$ and 8 in Fig. 11. It can be observed that an the additional gain in clock speedup of 27.6% for $k=2$ compared to $k=8$ with respect to DA_2 scheme.

Synthesis is performed using UMC 180nm CMOS technology by Cadence RTL compiler. It is clear from Fig. 10 that the area of proposed scheme is similar to that of existing DA_0 scheme but increased marginally due to extra 2-to-1 multiplexers for the proposed LUTs architecture.

The proposed scheme is implemented on Altera Cyclone III EP3C55F484C6 FPGA for 16-tap FIR filter at 100 MHz with wordlength of both input signal and weights are taken as 8-bits. Table II shows the FPGA implementation results in terms of logic elements for the proposed and various existing schemes. It can be noted that the number of logic elements are almost same as DA_0 scheme. This is due to two extra 2-to-1 multiplexers involved in multiplexed LUTs for the proposed scheme. Note that DA_2 scheme is based on OBC combinations of input samples and saves significant number of logic elements. Further, as base unit (k) increases require lesser number of logic elements due to less routing complexity.

Table III shows the estimates of power consumption for different base filter units and filter orders. It can be observed from the Table III that power consumption of the proposed scheme is marginally increased, however it can be reduced by the gain achieved in clock speedup for all DA base unit k compared to DA_2 scheme. Since the power consumption of digital system depends on the critical path [4] and can be lowered down due to gain in clock speedup and increase the throughput of the proposed scheme.

IV. Conclusions

This paper presents a low complexity and critical path architecture for LMS adaptive filter using distributed arithmetic. It is shown that for similar area and power performance, the proposed scheme maintains the same number of clock cycles using multiplexed LUTs which results in smaller critical path. The complexity of the proposed adaptive filter is significantly reduced. Moreover, due to additional gain in terms of clock speedup, the reduction in power consumption and increase the throughput can be possible. Hence, the proposed architecture provides an alternative design approach for high sampling rate and low complexity adaptive filters.

REFERENCES

[1] S. Haykin, *Adaptive Filter Theory (3rd Ed.)*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1996.

[2] P. K. Meher, "New approach to look-up-table design and memory-based realization of fir digital filter," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 57, no. 3, pp. 592–603, 2010.

[3] S. White *et al.*, "Applications of distributed arithmetic to digital signal processing: A tutorial review," *ASSP Magazine, IEEE*, vol. 6, no. 3, pp. 4–19, 1989.

[4] D. J. Allred, H. Yoo, V. Krishnan, W. Huang, and D. V. Anderson, "Lms adaptive filters using distributed arithmetic for high throughput," *Circuits and Systems I: Regular Papers, IEEE Transactions on*, vol. 52, no. 7, pp. 1327–1337, 2005.

[5] R. Guo and L. S. DeBrunner, "Two high-performance adaptive filter implementation schemes using distributed arithmetic," *Circuits and Systems II: Express Briefs, IEEE Transactions on*, vol. 58, no. 9, pp. 600–604, 2011.

[6] Guo, Rui and DeBrunner, Linda S, "A novel adaptive filter implementation scheme using distributed arithmetic," in *Signals, Systems and Computers (ASILOMAR), 2011 Conference Record of the Forty Fifth Asilomar Conference on*. IEEE, 2011, pp. 160–164.

[7] S. Y. Park and P. K. Meher, "Low-power, high-throughput, and low-area adaptive fir filter based on distributed arithmetic," *Circuits and Systems II: Express Briefs, IEEE Transactions on*, vol. 60, no. 6, pp. 346–350, 2013.

[8] M. S. Prakash and R. A. Shaik, "Low-area and high-throughput architecture for an adaptive filter using distributed arithmetic," *Circuits and Systems II: Express Briefs, IEEE Transactions on*, vol. 60, no. 11, pp. 781–785, 2013.

[9] P. K. Meher and S. Y. Park, "Critical-path analysis and low-complexity implementation of the lms adaptive algorithm," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 61, no. 3, pp. 778–788, 2014.