An Lut Adaptive Filter Using DA

[1]k.krishna reddy , [2]ch k prathap kumar m
[1]M.Tech Student, [2]Assistant Professor
[1]CVSR College of Engineering, Department of Electronics & Communications, A. P. India
[2]Department of Electronics & communication Engineering, Anurag group of Institutions, A.P. India
[1]kkr281 @gmail.com, [2]chaithu58@gmail.com

***Abstract* - Distributed arithmetic (DA) is performed to design bit-level architectures for vector–vector multiplication with a direct application for the implementation of convolution, which is necessary for digital filters. In this brief, A novel DA-based implementation scheme is proposed for adaptive finite-impulse response filter. To propose adaptive filter where filter coefficients are frequently updated in order to minimize the error out. Least-mean-square adaptation is performed to update the coefficients and minimize the mean square error between the estimated and desired output It involves a reduction in LUT(Look-up Table) size to one-fourth of the conventional LUT based on Anti-symmetric Product Coding(APC) and modified Odd Multiple Storage(OMS).**

***Keywords* - Distributed arithmetic (DA), Anti-symmetric Product Coding(APC) and modified Odd Multiple Storage(OMS), Least-mean-square(LMS).**

I.INTRODUCTION

MOST PORTABLE electronic devices such as cellular phones, personal digital assistants, and hearing aids require digital signal processing (DSP) for high performance. Due to the increased demand of the implementation of sophisticated DSP algorithms, low-cost designs, i.e., low area and power cost, are needed to make these handheld devices small with good performance. Various types of DSP operations are employed in practice. Filtering is one of the most widely used signal processing operations . For FIR filters,output y(n) is a linear convolution of weights w_n and inputs. For an Nth-order FIR filter, the generation of each output sample y(n) takes N +1 multiply-accumulate (MAC) operations. Since general-purpose multipliers require significant chip area, alternate methods of implementing multiplication are often used, particularly when the coefficients values are known prior to implementation. Distributed arithmetic (DA) is one way to implement convolution multiplierlessly, where the MAC operations are replaced by a series of LUT access and summations.

The implementation of an adaptive filter based on the DA concept poses several challenges. Since the DA filtering operation is based on an LUT, changes to the filter can require extensive changes to the LUT. This can be impractical for large LUT sizes. Several past attempts have been made to, changes to the filter can require extensive changes to the LUT. This can be impractical for large LUT sizes. In the new approach to LUT design, where only the odd multiples of the fixed coefficient are required to be stored, which we have referred to as the *odd-multiple-storage* (OMS) scheme in this brief. In addition, we have shown that, by the *anti symmetric product coding* (APC) approach, the LUT size can also be reduced to half, where the product words are recoded as anti symmetric pairs . The APC approach, although providing a reduction in LUT size by a factor of two, Several past attempts have been made to implement adaptive filters using DA but the approximations made to standard adaptation algorithms may be unsuitable for practical applications. In this paper, we develop and present an implementation of a FIR adaptive filter based on the well known LMS algorithm using the DA concept. A novel approach for updating the LUT tables of the DA filter is presented. The details of the proposed design and a description of the constituent modules of the DA-based LMS adaptive filter is provided in Sec. 3

II.BACKGROUND

A. DA

DA was first studied by Croisier et al in 1973 and popularized by Peled and Liu .DA is used to design bit- level architecture for vector multiplication.

Distributed Arithmetic, along with Modulo Arithmetic, are computation algorithms that perform multiplication with look-up table based schemes. Both stirred some interest over two decades ago but have languished ever since. Indeed, DA specifically targets the sum of products (sometimes referred to as the vector dot product) computation that covers many of the important DSP filtering and frequency transforming functions. The input samples are used as addresses to access a series of LUTs whose entries are sums of coefficients. Consider a discrete Nth-order FIR filter with constant coefficients, and input samples coded as B-bit two's complement numbers with only the sign bit to the left of the binary point as follows.

$$X(n-k) = -x_{k0} + \sum_{j=1}^{B-1} x_{kj}\, 2^{-j} \qquad (1)$$

Using (1) to compute the FIR output gives

$$y(n) = -\sum_{k=0}^{N} w_k x_{k0} + \sum_{j=1}^{B-1}\left[\sum_{K=0}^{N} w_k x_{kj}\right] 2^{-j} \qquad (2)$$

With $C_J = \sum_{K=0}^{N} w_k\, x_{kj}$, ∀j ∈ [1,B−1] and $C_0 = -\sum_{K=0}^{N} w_K x_{k0}$, (2) can be rewritten as

$$y(n) = \sum_{j=0}^{B-1} C_J\, 2^{-j} \qquad (3)$$

The C_j values can be precomputed and stored in a LUT with the input used as the address. This technique allows the FIR filter with known coefficients to be implemented without general-purpose multipliers. This implementation requires a LUT with a size that increases exponentially with the number of taps N +1, which results in a large time cost for accessing the LUT for a high-order filter. Therefore, reducing the LUT size improves system performance as well as area cost.To reduce LUT size, the anti-symmetric product coding (APC) and modified odd-multiple-storage (OMS) techniques. This technique reduces the conventional LUT size to its ¼ th.

B.APC AND OMS

APC-OMS based Filter design for low area and low static/dynamic power dissipation. The pre computed, stored values of LUT will be addressed by its inputs .This Proposed technique reduces the conventional LUT size to its ¼ th.

i) APC

For simplicity of presentation, we assume both *X* and *A* to be positive integers.2 The words for different values of *X* for *L* = 5 are shown in Table I. It may be observed in this table that the input word *X* on the first column of each row is the two's complement of that on the third column of the same row.

In addition, the sum of product values corresponding to these two input values on the same row is 32*A*. Let the product values on the second and fourth columns of a row be *u* and *v*, respectively. Since one can write *u* = [(*u* + *v*)/2 − (*v* − *u*)/2] and *v* = [(*u* + *v*)/2 + (*v* − *u*)/2], for (*u* + *v*) = 32*A*, we can have

$$u = 16A - \left[\frac{v-u}{2}\right] \quad v = 16A + \left[\frac{v-u}{2}\right].$$

The product values on the second and fourth columns of Table I therefore have a *negative mirror symmetry*. This behavior of the product words can be used to reduce the LUT size, where, instead of storing *u* and *v*, only [(*v* − *u*)/2] is stored for a pair of input on a given row. The 4-bit LUT addresses and corresponding coded words are listed on the fifth and sixth columns of the table, respectively. Since the representation of the product is derived from the anti symmetric behavior of the products, we can name it as *anti symmetric product code*. The 4-bit address *X*_ = (*x*_3*x*_2*x*_1*x*_0) of the APC word is given by

$$X' = \begin{cases} X_L, & \text{if } x_4 = 1 \\ X'_L, & \text{if } x_4 = 0 \end{cases}$$

where *XL* = (*x*3*x*2*x*1*x*0) is the four less significant bits of *X*, and *X*_*L* is the two's complement of *XL*. The desired product could be obtained by adding or subtracting the stored value (*v* − *u*) to or from the fixed value 16*A* when *x*4 is 1 or 0, respectively.

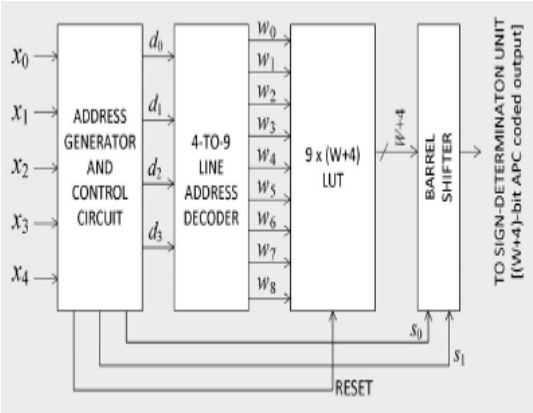


Fig 1 Proposed APC–OMS combined LUT design

Product word = 16*A* + (sign value) × (APC word)

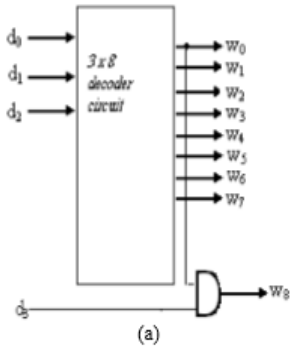
where sign value = 1 for *x*4 = 1 and sign value = −1 for *x*4 = 0. The product value for *X* = (10000) corresponds to APC value "zero," which could be derived by resetting the LUT output, instead of storing that in the LUT.

Input, X	product values	Input, X	product values	address $x'_3x'_2x'_1x'_0$	APC words
0 0 0 0 1	A	1 1 1 1 1	$31A$	1 1 1 1	$15A$
0 0 0 1 0	$2A$	1 1 1 1 0	$30A$	1 1 1 0	$14A$
0 0 0 1 1	$3A$	1 1 1 0 1	$29A$	1 1 0 1	$13A$
0 0 1 0 0	$4A$	1 1 1 0 0	$28A$	1 1 0 0	$12A$
0 0 1 0 1	$5A$	1 1 0 1 1	$27A$	1 0 1 1	$11A$
0 0 1 1 0	$6A$	1 1 0 1 0	$26A$	1 0 1 0	$10A$
0 0 1 1 1	$7A$	1 1 0 0 1	$25A$	1 0 0 1	$9A$
0 1 0 0 0	$8A$	1 1 0 0 0	$24A$	1 0 0 0	$8A$
0 1 0 0 1	$9A$	1 0 1 1 1	$23A$	0 1 1 1	$7A$
0 1 0 1 0	$10A$	1 0 1 1 0	$22A$	0 1 1 0	$6A$
0 1 0 1 1	$11A$	1 0 1 0 1	$21A$	0 1 0 1	$5A$
0 1 1 0 0	$12A$	1 0 1 0 0	$20A$	0 1 0 0	$4A$
0 1 1 0 1	$13A$	1 0 0 1 1	$19A$	0 0 1 1	$3A$
0 1 1 1 0	$14A$	1 0 0 1 0	$18A$	0 0 1 0	$2A$
0 1 1 1 1	$15A$	1 0 0 0 1	$17A$	0 0 0 1	A
1 0 0 0 0	$16A$	1 0 0 0 0	$16A$	0 0 0 0	0

Table 1: APC words for different input values

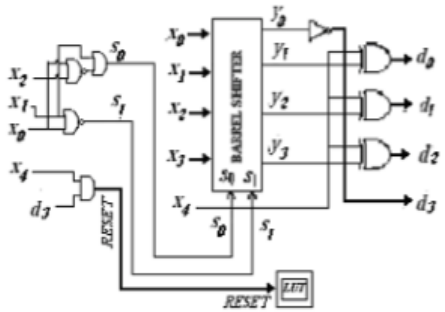
ii) OMS

A-OMS method is a different approach for implementing digital filters. The basic idea is to replace all multiplications and additions by a table & shifter-accumulator**.** In this method a barrel shifter is used to perform the shift operations through which the even multiples are computed from the obtained odd multiple values by simple shift operations which provides LUT optimized in terms of area by storing only the odd multiple values rather than whole values**.**



(a)

Figure. 2. (a) Decoder circuit.



(b) Control circuit and Address generation circuit (AGC)

In addition to the shifter circuit, a memory unit for product values and a decoder circuit for mapping bits as well as a control circuit and address generation circuit are required. The mapping process of 5-bit input word to a 4- bit LUT address (d0, d1, d2, d3) is done by a simple set of mapping relations. The address bits are thus generated from the AGC as shown in figure 2(b). OMS-BASED DESIGN OF THE LUT OF APC WORDS FOR *L = 5* AS SHOWN BELOW.

input X' $x'_3x'_2x'_1x'_0$	product value	# of shifts	shifted input, X''	stored APC word	address $d_3d_2d_1d_0$
0 0 0 1	A	0	0 0 0 1	$P0 = A$	0 0 0 0
0 0 1 0	$2 \times A$	1			
0 1 0 0	$4 \times A$	2			
1 0 0 0	$8 \times A$	3			
0 0 1 1	$3A$	0	0 0 1 1	$P1 = 3A$	0 0 0 1
0 1 1 0	$2 \times 3A$	1			
1 1 0 0	$4 \times 3A$	2			
0 1 0 1	$5A$	0	0 1 0 1	$P2 = 5A$	0 0 1 0
1 0 1 0	$2 \times 5A$	1			
0 1 1 1	$7A$	0	0 1 1 1	$P3 = 7A$	0 0 1 1
1 1 1 0	$2 \times 7A$	1			
1 0 0 1	$9A$	0	1 0 0 1	$P4 = 9A$	0 1 0 0
1 0 1 1	$11A$	0	1 0 1 1	$P5 = 11A$	0 1 0 1
1 1 0 1	$13A$	0	1 1 0 1	$P6 = 13A$	0 1 1 0
1 1 1 1	$15A$	0	1 1 1 1	$P7 = 15A$	0 1 1 1

Table2: OMS based design of the lut of APC words.

III.DA BASED ADAPTIVE FILTER

An adaptive filter changes its weights wk with time to match a desired performance objective. Typically, the performance of the adaptive filter is quantified in terms of the mean square value of the error between its output y[n] and a desired signal d[n]. The least mean-square (LMS) adaptation algorithm updates the weights to minimize the mean-square error (MSE) of the output. The weight adaptation in an LMS adaptive filter is given by

$w_k[n+1] = w_k[n]+\mu e[n]x[n-k]$ (4)

where $e[n]=d[n]-y[n]$.

Several approximations of the LMS algorithms are often used for hardware implementations. The sign error LMS (SE-LMS) approximates the error as $e[n]=sign(d[n] - y[n])$ and the sign data LMS (SD-LMS) replaces the term $x[n-k]$ by $sign(x[n-k])$. In this paper, an LMS-type algorithm is implemented where the term $\mu e[n]$ is quantized to a power of 2. Implementation results demonstrate that this quantized error LMS (QE-LMS) outperforms the SE-LMS and is comparable to the LMS algorithm in terms of the convergence speed.

The LMS adaptation algorithm requires the filter weights wk[n] to be updated according to Eq. (4) every sample that is filtered. After

calculating the updated weights, the entries of the LUT, which are all possible combination sums of the weights, are recalculated and updated. Doing this on a sample-by-sample basis is computation- ally expensive and time consuming, causing significant reduction in the filter throughput. For example, a brute-force update of the DA-F-LUT could take approximately 1000 clock cycles for a 128 tap FIR filter. The overall system level diagram of the proposed implementation is shown in Fig3.

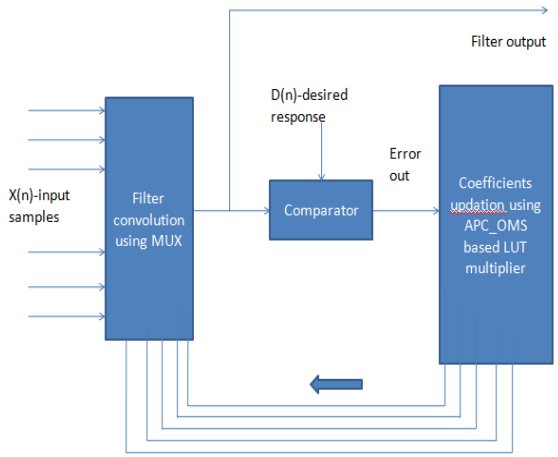
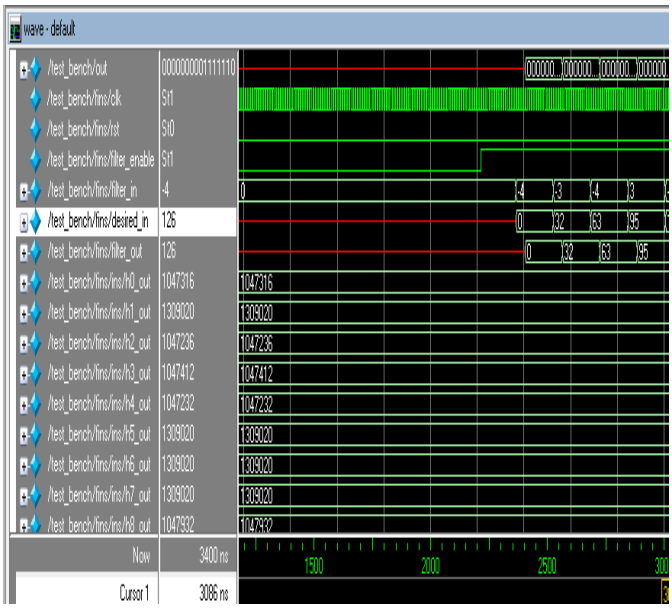


Fig3: Proposed DA Adaptive Filter System

The simulated results are as shown in the figure 4. The overall design process enhances the system performance in terms of speed and area that doubles the transmission rate, increasing the overall throughput.



IV.CONCLUSION

In this brief, An efficient DA-based FIR adaptive filter implemented. In contrast to conventional DA-based schemes, APC-OMS based Filter design for low area and low static/dynamic power dissipation. This technique reduces the conventional LUT size to its ¼ th. This can be used in echo cancelation and system identification, coefficient adaptation is needed. This adaptation done by LMS Algorithm.This is shown in tabular re-presentations and explained briefly in the above sections. The overall implementation process is thus simulated using XILINX ISE Project Navigator and the simulated result is shown.

REFERENCES

[1] C. H. Wei and J. J. Lou, "Multi memory block structure for implementing a digital adaptive filter using distributed arithmetic," Proc. Inst. Elect. Eng., vol. 133, no. 1, pt. G, pp. 19–26, Feb. 1986.

[2] C. F. N. Cowan and J. Mavor, "New digital adaptive-filter implementation using distributed-arithmetic techniques," Proc. Inst. Elect. Eng., vol. 128, no. 4, pt. F, pp. 225–230, Feb. 1981.

[3] 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.

[4] D. J. Allred, H. Yoo, V. Krishnan, W. Huang, and D. V. Anderson, "A novel high performance distributed arithmetic adaptive filter implemen- tation on an FPGA," in Proc. IEEE Int. Conf. Acoust., Speech, Signal Process., 2004, vol. 5, pp. V-161–V-164.

[5] P. K. Meher, 'New Approach to Look-up-Table Design and Memory- Based Realization of FIR Digital Filter,' IEEE Trans on Circuits &Systems-I pp 592-Systems I, pp.592 603, March 2010.