LMS Adaptive Filter Implementation Using Distributed Arithmetic For Noise Cancellation

Samayam Pullarao
M Tech Student
Dept. of ECE
BMS College of Engineering
Bangalore

Dr. D Seshachalam
Professor& HOD
Dept. of ECE
BMS College of Engineering
Bangalore

***Abstract*—In this paper, distributed arithmetic (DA) based adaptive filter for noise cancellation application is presented. Adaption of filter weights is done by using least mean square (LMS) algorithm. Distributed arithmetic (DA) is bit serial in nature, it is used to design bit-level architectures for vector-vector multiplications. It replaces the multiply-accumulate (MAC) operations in filtering process with a series of look-up-table (LUT) access. However, implementing adaptive DA filters require recalculating the LUT's for each adaptation, it increases the logical complexity. In this paper anti symmetric product coding (APC)- odd multiple storage (OMS) techniques are also implemented to reduce the computational complexity of recalculating the LUT contents and it is also increases the throughput of the filter. The zero-mean random noise signal is completely eliminated by using proposed DA based adaptive filter.**

***Index Terms*—APC, DA, LMS, OMS, ANC, MAC, LUT.**

1. INTRODUCTION

Most of the portable electronic devices such as cellular phones, personal digital assistants, and wireless devices often require digital signal processing (DSP) for high performance. Due to increase in the demand for complex DSP applications low power, low area and high performance system-on-chip (SOC) implementations of DSP algorithms are receiving increased attention. Several types of DSP operations are employed in practice. Filtering is one of the most widely used signal processing operations. A discrete-time linear finite impulse response (FIR) filter generates the output $y[n]$ as a sum of delayed and scaled input samples $x[n]$ is represented by the equation.

$$y[n]=\sum_{k=0}^{N-1} w_k x[n-k] \qquad (1)$$

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.

However, in many applications such as acoustic echo cancellation, signal de-noising, sonar signal processing, clutter rejection in radars, and channel equalization for communications and networking systems coefficient adaptation is needed. This adaption makes it challenging to implement DA-based adaptive filters with low cost due to the necessity of updating LUTs. Several approaches have been developed for DA-based adaptive filters, i.e., from the point of view of reducing logic complexity [1]–[3], [5].

In this paper we developed DA based adaptive filter for noise cancellation of zero-mean random noise signal. Adaptive noise cancellation (ANC) is a technique of estimating noise or interference in a corrupted signal by passing it through an adaptive filter. DA based LMS adaptive filter implementation is done in verilog. The LUT optimization using the APC coding and OMS methodology are the primary factors for LUT based adaptive FIR filter is designed for DSP applications. The odd integer representation is always used for input and output address transformation [2],[6]. For noise cancellation application original signal and noisy signals are generated from MATLAB and converted in to 2's compliment binary format.

2. BACKGROUND

A. LMS Algorithm

LMS algorithm uses the estimates of the gradient vector from the available data. LMS algorithm incorporates an iterative procedure that makes successive corrections to the weight vector in the direction of the negative of the gradient vector which eventually leads to min-mean square error.
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 eq no.2

$$w_k[n+1] = w_k[n] + \mu e[n]x[n-k] \qquad (2)$$

Where $e[n] = d[n] - y[n]$, y[n]= w[n] * x[n-k]

Several approximations of the LMS algorithms[2] are often used for hardware implementations. In this paper, an LMS-type algorithm is implemented where the term $\mu e[n]$ is quantized to a power of 2.

B. DA

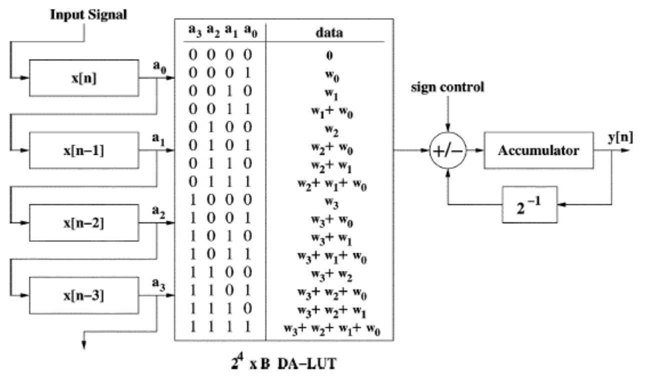
DA was first introduced by Croisier et al [5]in 1973 and further developed by Peled and Lui [3]. DA provides a multiplier-less implementation of FIR filters through a bit-serial computation utilizing all possible combination sums of the filter coefficients. It is assumed that the inputs to the filter are represented as *B* bit 2's complement binary numbers with only the sign bit to the left of the radix point. Then

$$x[n-k] = -x_{k0} + \sum_{j=1}^{B-1} x_{kj} 2^{-j} \qquad (3)$$

Substitute eq. no(3) in eq. no(1), we get

$$y[n] = -\sum_{k=0}^{N} w_k x_{k0} + \sum_{j=1}^{B-1} \left[\sum_{k=0}^{N} w_k x_{kj} \right] 2^{-j} \qquad (4)$$

It is noted that the terms in the square brackets may take only one of 2^k possible values; these values, which are all the possible combination sums of the filter coefficients, are stored in an LUT, denoted as the DA filtering LUT . The filtering operation may then be implemented, according to Eq. 3, by *B* look-up, shift, and accumulate operations. The block diagram of a typical DA implementation of a four tap (*K* = 4) FIR filter is shown in Fig. 1.



a_3 a_2 a_1 a_0	data
0 0 0 0	0
0 0 0 1	w_0
0 0 1 0	w_1
0 0 1 1	w_1+w_0
0 1 0 0	w_2
0 1 0 1	w_2+w_0
0 1 1 0	w_2+w_1
0 1 1 1	$w_2+w_1+w_0$
1 0 0 0	w_3
1 0 0 1	w_3+w_0
1 0 1 0	w_3+w_1
1 0 1 1	$w_3+w_1+w_0$
1 1 0 0	w_3+w_2
1 1 0 1	$w_3+w_2+w_0$
1 1 1 0	$w_3+w_2+w_1$
1 1 1 1	$w_3+w_2+w_1+w_0$

Fig.1. Block diagram of DA implementation of a 4-tap (k=4) FIR filter. Each coefficient has B bits of precision.

The bank of shift registers in Fig. 1 stores four consecutive input samples. The concatenation of the rightmost bits of the shift registers becomes the address of the LUT. The shift registers are shifted right at every clock cycle. The corresponding LUT entries are also shifted and accumulated *B* consecutive times where *B* is the precision of the input data.

3. PROPOSED DA ADAPTIVE FILTER DESIGN

To implement the LMS adaptive filter using the DA architecture, the entries of the LUT, which contains all possible combination sums of the filter weights, need to be recalculated and updated on a sample-by-sample basis[10],[11]. A brute-force implementation that updates each weight individually according to (2), and then regenerates the LUT using the new weights, will be computationally expensive and time consuming, causing significant reduction in the filter throughput.

In this paper LUT optimization is done by using anti symmetric product coding (APC) and odd multiple storage (OMS) to reduce the complexity in updating LUT contents [6]. The tables of multiplication are pre-calculated and stored in memory. For fast accessing of values from the memory, LUT's are used for saving the computation complexity. In digital logic, an n-bit LUT can be implemented with a multiplexer whose select lines are the inputs of LUT and inputs are constants. An n-bit LUT can encode any n-input Boolean function by modelling with truth tables. LUT's with 4-6 bits of input are the key component of modem FPGAs and this is an efficient way of encoding functions. General representation of LUT for multiplication bits are shown in fig.2.

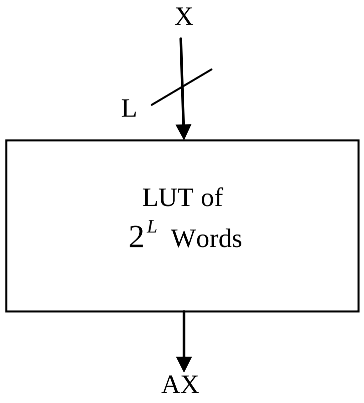
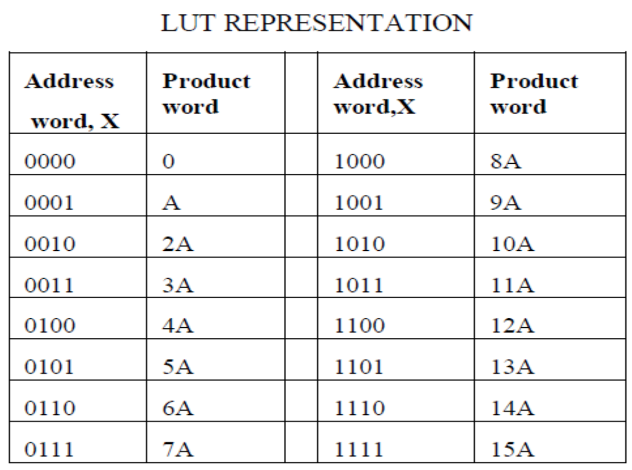


Fig.2. Normal LUT multiplier

In general LUT multiplier it has the input bit 'X' of length 'L' and 'AX' as output bit, where A is the constant depends on the LUT value. 2^L words are required for multiplying X of L-bit with constant. With the increase in input size LUT size increases exponentially. LUT for input of word length L=4 requires 16 address lines to store the input bit sequence is shown in below table 1



LUT REPRESENTATION

Address word, X	Product word		Address word,X	Product word
0000	0		1000	8A
0001	A		1001	9A
0010	2A		1010	10A
0011	3A		1011	11A
0100	4A		1100	12A
0101	5A		1101	13A
0110	6A		1110	14A
0111	7A		1111	15A

Table 1. LUT representation.

Address Word	Product Word
0001	A
0011	3A
0101	5A
0111	7A
1001	9A
1011	11A
1101	13A
1111	15A

Table 2. OMS based reduction scheme for LUT

By using the OMS scheme only odd multiplies are stored in the LUT and the even multiplies of the LUT are derived by left shifting the odd multiplies by using the barrel shifter scheme. By using the barrel shifter we can produce the maximum (L-1) no. of left shifts to produce the even multiples.

The implementation of the APC-OMS combined LUT for memory based multiplier uses two techniques. This method is supposed to reduce the area to one fourth. The address generation block converts our input to address d0, d1, d2which is produced by combining both the APC and OMS method. The 3-to-8 address line decoder converts the address d0, d1, d2 to LUT address from w1 to w7. The memory array is an LUT and barrel shifter converts the LUT output to the desired output. The control circuit is used to produce the controls s0, s1 which is used in the proceeding blocks [4] [6]. The control and reset circuit can be designed as

S0=x0+(x1+x2’) (5)

S1=(x0+x1) (6)

Reset= x3 and x2’x1’ (7)

The barrel shifter will right shift circularly according to the control values (s0 s1), using the basic gates to produce the control elements reset, s0, s1. From the barrel shifter, thus producing the address (d0d1d2) to use in the next sections. The top-level circuit diagram of this proposed scheme for an example four-tap FIR filter is shown in Fig. 3 and address generator and controller module is shown in fig.4.

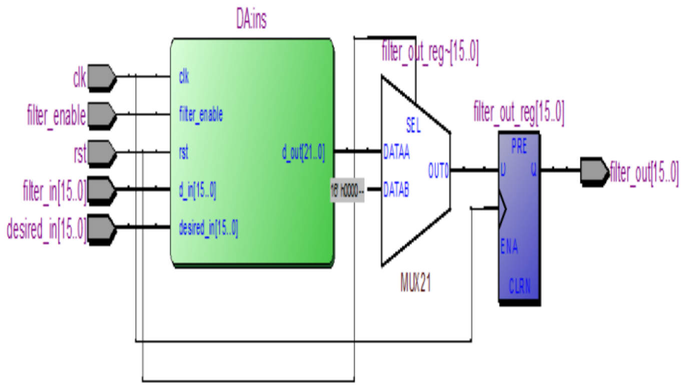


Fig 3. Top level circuit diagram of proposed DA adaptive filter.

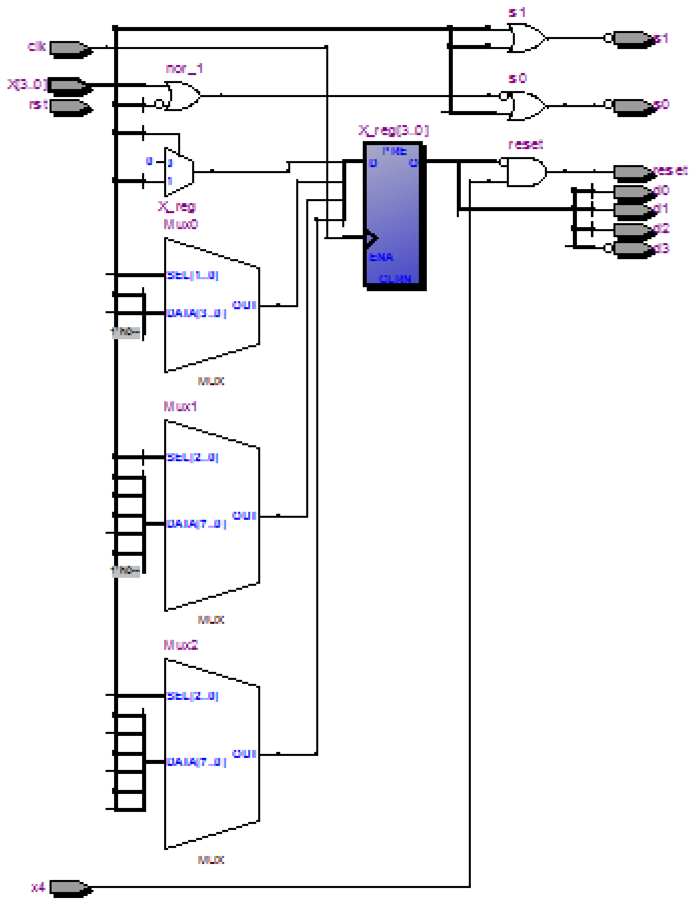


Fig.4. RTL schematic diagram of address generator and controller unit.

International Journal of Advanced Information Science and Technology (IJAIST) ISSN: 2319:2682
Vol.4, No.4, April 2015 DOI:10.15693/ijaist/2015.v4i4.26-30

4. SIGNAL GENERATION

The input signals to the filter are generated from the MATLAB [1]. Two types of signals are required for the application of noise cancellation using adaptive filters. First is original or actual signal, here we considered it as sinusoidal signal. Second one is noisy signal, here we considered it as zero-mean random noise signal. These signals are converted to 2's compliment binary format and saved as text files. These text files are accessed by the Verilog implementation. Noisy signal is the primary input and original signal is the desired signal to the adaptive filter [12]. The signals are shown in the fig. 5.

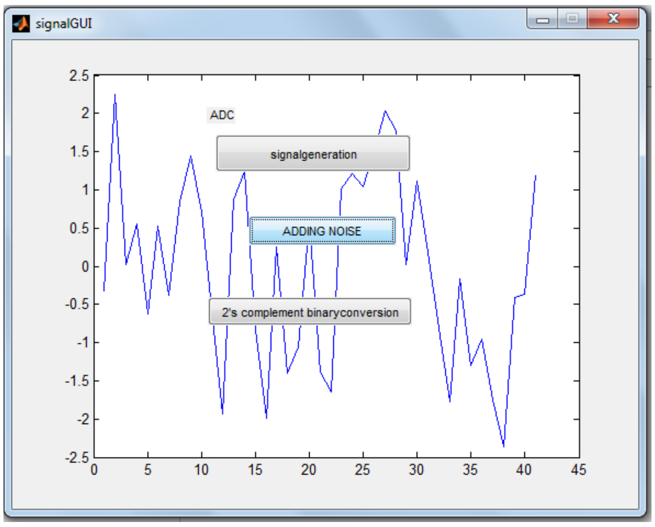


Fig.5. GUI representation of zero-mean random noise signal from MATLAB.

5. IMPLEMENTATION METHODOLOGY.

The whole implementation of DA based LMS adaptive filter implementation is done by using Verilog HDL. Implementation consist of four modules, 1. Address generator and control module, 2. Adder or subtract module, 3. Barrel shifter module, 4. APC module.

The address generator and controller module performs the effective address calculations necessary to address data operands in memory and contains the registers used to generate the addresses and it also generates control signals to indicate the filtering operation at that particular sample time is completed or not, it is shown in fig.4. A barrel shifter is a digital circuit that can shift a data word by a specified number of bits in one clock cycle. It can be implemented as a sequence of multiplexers (mux.), and in such an implementation the output of one mux is connected to the input of the next mux in a way that depends on the shift distance. It is implemented with only combinational logic.

6. IMPLEMENTATION RESULTS

To evaluate the performance of the proposed 16-tap DA adaptive filter was implemented on Altera Cyclone IV GX EP4CGX22CF19C6 FPGA. At each instant of time the output signal bit pattern of the filter is exactly equal to the desired signal bit pattern at that instant of time with some processing delay. Simulation results are obtained by simulating the design on Xilinx ISE design suite 14.2 and the simulation wave forms are shown in fig.6.

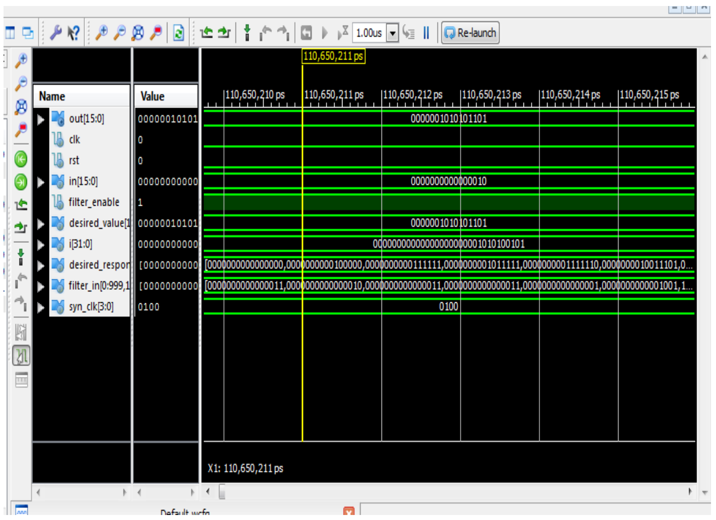

1.00us
Re-launch
110,650,211 ps
Name
Value
110,650,210 ps
110,650,211 ps
110,650,212 ps
110,650,213 ps
110,650,214 ps
110,650,215 ps
out[15:0] 00000010101 0000001010101101
clk 0
rst 0
in[15:0] 00000000000 0000000000000010
filter_enable 1
desired_value[1 00000010101 0000001010101101
i[31:0] 00000000000 00000000000000000000001010100101
desired_respor [0000000000 [0000000000000000,0000000000100000,0000000000111111,0000000001011111,0000000001111110,0000000010011101,0...
filter_in[0:999,1 [0000000000 [0000000000000011,0000000000000010,0000000000000011,0000000000000011,0000000000000001,0000000000001001,1...
syn_clk[3:0] 0100 0100
X1: 110,650,211 ps
Default.wcfg

Fig.6. simulation results of DA adaptive filter for the application of noise cancellation.

Implementation details :

Quartus II Version Web Edition	11.0 Build 157 04/27/2011 SJ
Revision Name	DA_Adaptive_filter
Top-level Entity Name	DA_Adaptive_filter
Family	Cyclone IV GX
Total logic elements	64 / 21,280 (< 1 %)
Total registers	64
Total pins	51 / 167 (31 %)
Device	EP4CGX22CF19C6

Total Thermal Power Dissipation	87.35 mW
Core Dynamic Thermal Power Dissipation	0.00 mW
Core Static Thermal Power Dissipation	81.18 mW
I/O Thermal Power Dissipation	6.17 mW

THROUGHPUT:

The throughput is defined as the number of signal samples processed by an adaptive filter per second [10]. If 't' is the number of clock cycles required for filtering and updating the filter weights according to the adaption algorithm, then

$$Throughput = \frac{clockrate}{t} \tag{8}$$

It is obvious that the throughput of any system is depends on it's processing time. The throughput of proposed design depends up on the size of LUTs since updating them takes the longest time in the entire system processing time. The size of LUTs are reduced by using the LUT optimization techniques. The throughput comparison between MAC based adaptive filter and proposed DA based adaptive filter is shown in fig.7.

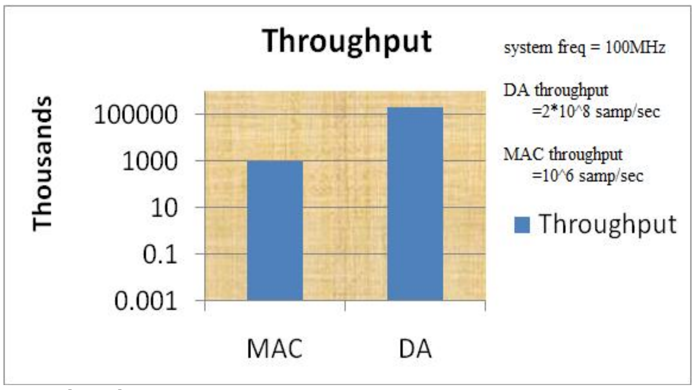


Fig 7: Throughput comparison

7. CONCLUSION

In this brief, LMS adaptive filter implementation based on DA for the application of noise cancellation has been presented. The DA concept involves the implementation of a multiply-and-accumulate operation using look-up-tables(LUT). The proposed scheme uses LUT optimization techniques anti symmetric product coding and odd multiple storage (APC-OMS) to reduce the computational complexity involved in LUT updation. By using this only half of the LUT contents are need to be recomputed, hence the logical complexity and area and power consumption are reduced significantly. Zero-mean random noise signal is completely eliminated from the original signal by passing it through the proposed DA adaptive filter is shown in the simulation results.

8. REFERENCES

[1] S. Haykin, *Adaptive Filter Theory*. Upper Saddle River, NJ: Prentice-Hall, 1996.

[2] B. Farhang-Boroujeny, *Adaptive Filters: Theory and Applications*. Chichester, U.K.: Wiley, 1998.

[3] K. K. Parhi, *VLSI Digital Signal Processing Systems: Design and Implementation*. Hoboken, NJ: Wiley, 1999.

[4] 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.

[5] Eldho John, P. Dinesh Kumar "MODIFIED APC-OMS COMBINED LUT FOR MEMORY BASED COMPUTATION", International Journal of Systems, Algorithms & Applications Volume 2, Issue 3, March 2012, ISSN Online: 2277-2677.

[6] Sang Yoon Park, Pramod Kumar Meher " Low Power, High-Throughput, and Low-Area Adaptive FIR Filter Based on Distributed Arithmetic" IEEE Trans. On Circuits and Systems Vol. 60, No.6, June 2013.

[7] 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.

[8] 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.

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

[10] 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.