Low Power, High Throughput Adaptive FIR Filter Designed for Real Time Audio Denoising

C. Ezhilarasi[1], A. Rajan[2]
PG **S**cholar[1], Assistant. Professor[2]
Shri Andal Alagar College of Engineering, TN, India

Abstract - Adaptive filters find extensive use in many signal processing applications such as channel equalization, echo processing, noise cancellation etc. It can be implemented using one or more multiply and accumulate (MAC) units. But the Memory-based structures provides better performance in area minimization compared with multiply-accumulate structures and have many other advantages such as reduced latency since the memory-access-time is much shorter than the usual multiplication-time compared to the conventional multipliers. So the efficient memory computing system alternative to conventional logic computing is required in many DSP applications. The LUT designed using combined APC-OMS technique is used. This APC-OMS technique reduces the LUT size to one-fourth of its original area. If conventional LUT is used, the on-chip memory size is getting larger. The memory size can be reduced by decomposing the LUT. FIR filter is designed using multiplexer which is used to select the filter coefficients. Hence, the combination of these two techniques provides reduction in LUT size to one fourth in adaptive FIR filter when compared with the conventional Look up Table (LUT) of adaptive FIR filter. By the proposed method, area efficiency and throughput is achieved. The filter designs are simulated using Modelsim6.4a and are synthesized using Quartus II 9.0.

Index terms - Distributed arithmetic, FIR filter, throughput, Memory-based implementation, UT-multiplier-based approach.

I. INTRODUCTION

Adaptive filters find extensive use in many signal processing applications such as channel equalization, echo processing, noise cancellation etc. Such filters typically involve finite impulse response (FIR) filters whose weights are updated according to some minimizing criteria[1]. FIR filters can be implemented using one or more multiply and accumulate (MAC) units since the output of the filter is the weighted sum of input samples but this would be time consuming and multiple MAC units may increase the system complexity. Several attempts have been made in order to implement adaptive filters using reduction techniques where the filter weights are updated once for each output block[2]. Later, pipelined architectures using look-ahead and relaxed look-ahead have been proposed in order to implement high sampling rate adaptive filters[3]-[4].

ICGPC 2014
St.Peter's University, TN, India.

They proved to be effective but the pipelining or parallel implementations either produce adaptation delays or increase the hardware requirement. In order to overcome this, the first frequency domain adaptive filter has been proposed where a smaller block delay is maintained by using smaller block sizes. Once again the pipelining technique has been adopted but this time without the adaptation delay and degradation in the convergence performance. With semi-conductor memories becoming more cheaper and much faster since the memory access time is much less than the propagation delay, there is an inclination in the field of research towards the use of semiconductor memories. Distributed arithmetic is a preferable method since it can realize large filters without hardware multipliers to produce very high throughputs[5]. Few attempts have been made in order to implement adaptive filters based on the funcion of DA. The look-up-tables with special addressing produces the high throughput.

Finite Impulse Response (FIR) filters are one of the key building blocks of many signal processing applications in communication systems. Channel equalization, interference cancellation and matched filtering are some variety of FIR filter applications. Hence, the programmable and reconfigurable FIR filter architectures are needed for next generation communication systems with low power consumption, low complexity and high speed operation requirements. The major bottleneck in FIR filter implementation is coefficients multipliers, which are traditionally implemented by add/sub/shift operations.

Digital filters are the essential units for digital signal processing systems. Traditionally, digital filters are achieved in Digital Signal Processor(DSP), but DSP-based solution cannot meet the high speed requirements in some applications for its sequential structure. Nowadays, Field Programmable Gate Array (FPGA) technology is widely used in digital signal processing area because FPGA-based solution can achieve high speed due to its parallel structure and configurable logic, which provides great flexibility and high reliability in the course of design and later maintenance. Several architectures have been reported in the literature for memory-based implementation of DSP algorithms involving orthogonal transforms and digital filters [8]. However, we do not find any significant work on LUT optimization for memory-based multiplication and have presented a new approach to LUT design, where only the odd multiples of the fixed coefficient are required to be stored[9],,

which we have referred to as the odd-multiple storage (OMS) scheme. In this paper, we propose an effective architecture for adaptive filter where filter coefficients are frequently updated in order to minimize the error out. 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).The proposed LUT-based FIR involves reduction in area ,time delay, power consumption and high throughput.

II. REVIEW OF ADAPTIVE FILTER USING LUT

A fixed coefficient filter can be implemented using DA efficiently regardless of the filter length. However, problem arises when implementing an adaptive filter whose coefficients are to be adapted which requires that the contents of the LUT are to be updated. In-case of a Least Mean Square (LMS) adaptive filter, the weights are updated to minimize the mean squared error ξ according to the update equation

$$w_i[n+1] = w_i[n] + \mu x[n-i]e[n]$$

$d[n]$ being the desired signal and μ being the step size.

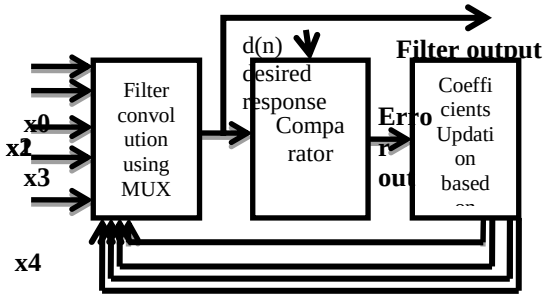

Fig. 1. Structure of adaptive filter using LUT multiplier

The block diagram of adaptive filter is depicted in Fig. 1. It can be seen that the LUT designed with memory size fully gives the large area usage and which inturn logic elements increases the area and power increases and throughout is low. Therefore in the conventional algorithm the coefficients are updated by changing the LUT multiplier by APC-OMS method.

The LUT designed using combined APC-OMS technique can be used for these applications. APC-OMS technique reduces the LUT size to one-fourth of its original. FIR filter is designed using conventional LUT. If conventional LUT is used, the on-chip memory size is getting larger. The memory size can be reduced by decomposing the LUT. FIR filter is designed using multiplexer which is used to select the filter co-efficient. With LUT, area and delay efficiency cannot be achieved. This design has less number of memory accesses and area than LUT based FIR design. By the proposed method, area efficiency and throughput is achieved.

III. PROPOSED SYSTEM

Transposed form FIR filter can have a small cycle time of one MAC operation only, but it can produce one correct output in every N cycles for N tap filter in training phase. Due to its high-latency, it could be used only as a low-hardware solution for off-line processing. The direct-form FIR filters on the other hand can be implemented without any pipeline delay, but that would have a very large critical-path. A straight-forward implementation of direct-form FIR adaptive filter, therefore, would not be suitable for high-speed adaptive filtering. We can reduce the critical path by introducing pipeline registers on the critical path, but the performance of adaptive filter degrades due to the adaptation delays, which primarily results from the number of pipeline stages in computation of error during the training process. It is, therefore, important to decide suitable pipelining scheme to perform the computation of direct-form FIR filter for efficient implementation of DLMS algorithm.

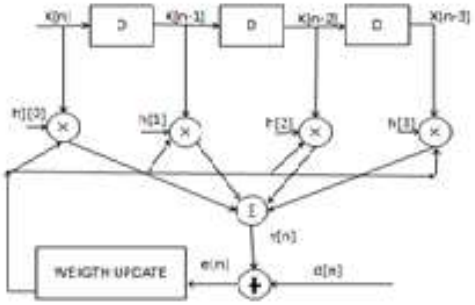

Fig. 2. Adaptive FIR filter design

As shown in Fig. 2., there are two main computing functions in the adaptive filter architecture, namely error computation and weight update. Error computation is done by Multiplexer that is input is taken from the weight update block which is done by LUT optimization.

They can offer shape factor accuracy and stability equivalent to very high-order linear active filters that cannot be achieved in the analog domain. Unlike IIR (Infinite Impulse Response) filters, FIR filters are formed with only the equivalent of zeros in the linear domain. This means that the taps depress or push down the amplitude of the transfer function. The amount of depression for each tap depends upon the value of the multiplier coefficient. Hence, the total number of taps determines the "steepness'" of the slope.

The number of taps (delays) and values of the computation coefficients (h0, h1,..hn..) are selected to "weight" the data being shifted down the delay line to create the desired amplitude response of the filter. In this configuration there are no feedback paths to cause instability. The calculation coefficients are not constrained to particular values and can be used to implement filter functions that do not have a linear system equivalent. Note: more taps increase the steepness of the filter roll-off while increasing calculation time (delay) and for high order filters, limiting bandwidth. The filter delay is easily calculated for the above structure. Delay = (½ x Taps)/Sampling rate.

A. APC & OMS

This paper deals with the anti-symmetric product coding (APC) and modified odd-multiple-storage (OMS) techniques for lookup-table (LUT) design for memory-based multiplication and

division process. The combined approach provides a reduction in LUT size to one-fourth of the conventional LUT. It is shown that the proposed LUT design for small input sizes can be used for efficient implementation of high-precision multiplication by input operand decomposition. It is found that the proposed LUT involves comparable area and time complexity for a word size of 8 bits, but for higher word sizes, it involves significantly less area and less multiplication time than the canonical-signed-digit (CSD)-based multipliers.

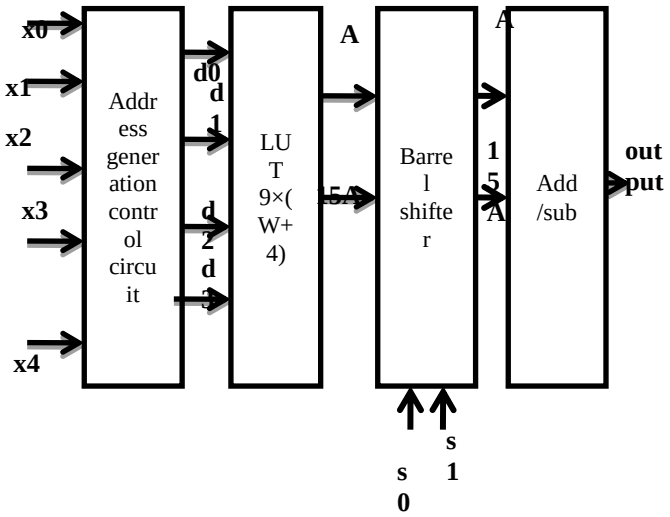

Fig. 3. APC AND OMS for LUT multiplier

The implementation of the proposed APC-OMS combined LUT for memory based multiplier uses two techniques, APC and OMS method is shown in Fig. 3. This method is supposed to reduce the area to one fourth. This multiplier uses four blocks. The first block converts the given input to address d0d1d2d3, which is produced by combining both the APC and OMS method. The second block converts the address d0d1d2d3 to LUT address. The third block is an stored LUT and fourth block converts the LUT output to the desired output.

B. IMPLEMENTATION OF LUT USING THE OPTIMIZATION SCHEME

In TABLE I, the proposed APC–OMS combined design of the LUT for $L = 5$ and for any coefficient width W. It consists of an LUT of nine words of $(W + 4)$-bit width, a four-to-nine-line address decoder, a barrel shifter, an address generation circuit, and a control circuit for generating the RESET signal and control word ($s1s0$)for the barrel shifter. The precomputed values of $A \times (2i + 1)$ are stored as $.Pi$ for $i = 0, 1, 2, \ldots, 7$, at the eight consecutive locations of the memory array, as specified in Table.2, while $2A$ is stored for input $X = (00000)$ at LUT address "1000,"

$s0 = (x0+(x1+x4')')'$

$s1 = (x0 + x1)'$

TABLE I

ANTISYMMETRIC PRODUCT CODING

Input	Product values	Input	Product values	Address	APC words
00001	A	11111	31A	1111	15A
00010	2 A	11110	30A	1110	14A
00011	3 A	11101	29 A	1101	13A
00100	4 A	11100	28A	1100	12A
00101	5 A	11011	27A	1011	11A
00011	6 A	11010	26A	1010	10A
00111	7 A	11001	25A	1001	9A
01000	8 A	11000	24A	1000	8A
01001	9 A	10111	23A	0111	7A
01010	10 A	10110	22A	0110	6A
01011	11 A	10101	21A	0101	5A
01100	12 A	10100	20A	0100	4A
01101	13 A	10011	19A	0011	3 A
01110	14 A	10010	18A	0010	2A
01111	15 A	10001	17A	0001	A
10000	16 A	10000	16A	0000	0

Note that ($s1s0$) is a 2-bit binary equivalent of the required number of shifts specified. The RESET signal can alternatively be generated as ($d3$ AND $x4$). The address-generator circuit receives the 5-bit input operand X and maps that onto the 4-bit address word ($d3d2d1d0$).

In TABLE II, it is given that, the eight odd multiples, $A \times (2i + 1)$ are stored as Pi, for $i = 0, 1, 2, \ldots, 7$ at eight locations.

The even multiples $2A$, $4A$, and $8A$ are derived by left-shift operations of A. In similar way, $6A$ and $12A$are derived by left shifting $3A$, while $10A$ and $14A$ are derived by left shifting $5A$ and $7A$, respectively. A barrel shifter for producing a maximum of three left shifts could be used to derive all the even multiples of A.

The word to be stored for $X = (00000)$ is not 0, it is supposed to be $16A$, which can be obtained from A by four left shifts using a barrel shifter. However, if $16A$ is not derived from A, only a maximum of three left shifts is required to obtain all other even multiples of A-stage barrel shifter. Therefore, if we store $2A$ at (1000), the product $16A$ can be derived by three arithmetic left shifts which is required at location X=(00000). The product values and encoded words for input words $X = (00000)$ and (10000) are shown in TABLE I. For $X = (00000)$, the desired

encoded word16*A* is derived by 3-bit left shifts of 2*A* which is stored at address (1000).

TABLE II

ODD MULTIPLE STORAGE

Input x3'x2'x1'x 0'	Product value	No of shifts	Shifted input Y	Stored APC word	Address d3'd2'd1'd0'
0001	A	0	0001	A	0000
0010	2 ×A	1	0001	A	0000
0100	4 ×A	2	0001	A	0000
1000	8 ×A	3	0001	A	0000
0011	3 A	0	0011	3A	0001
0110	2×3A	1	0011	3A	0001
1100	4×3 A	2	0011	3A	0001
0101	5A	0	0101	5A	0010
1010	2× 5A	1	0101	5A	0010
0111	7 A	0	0111	7A	0011
1110	2× 7A	1	0111	7A	0011
1001	9 A	0	1001	9A	1000
1011	11 A	0	1011	11 A	0101
1101	13 A	0	1101	13 A	0110
1111	15 A	0	1111	15 A	1111

For X = (10000), the APC word "0" is derived by resetting the LUT output, by an active-high RESET signal given by

RESET = ~($x0$+$x1$+$x2$+$x3$) .$x4$

IV. IMPLEMENTATIONS AND COMPLEXITIES

We have coded the proposed structure in verilog and synthesised by Quartus II 9.0 for different filter orders. The proposed designs were also implemented on the field-programmable gate array platform of Xilinx devices. The simulated and analysis results are of LUT with full memory size of conventional method has more logical elements and LUT using APC-OMS optimization method gives less logical elements so the flow summary shows that memory size reduces compared with the conventional method as shown in the Fig. 4.,Fig. 5.and Fig. 6. The power of the corresponding conventional and proposed system gives the reduction in power comparatively and throughput increases correspondingly. The overall design process enhances the system performance in terms of speed and area increasing the overall throughput.

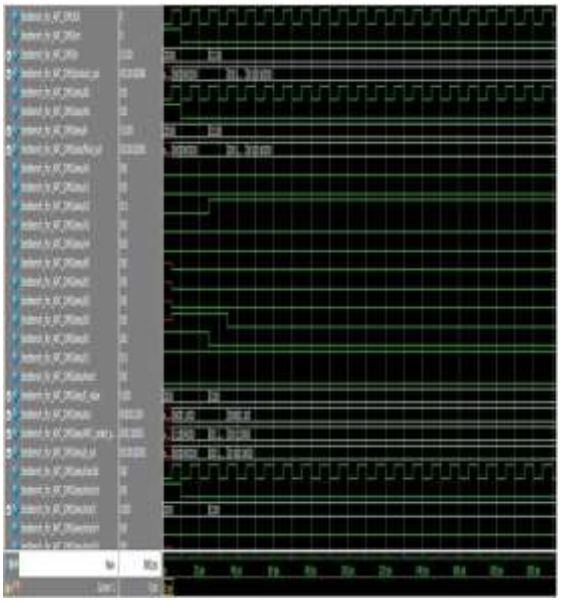

Fig. 4. Simulated output of LUT optimization using APC=OMS

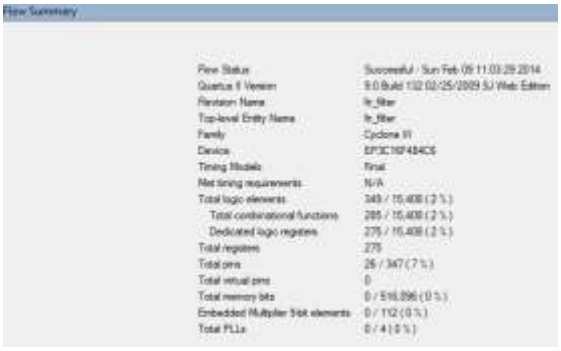

Fig. 5. Analysis report of LUT for Area

Fig. 6. Analysis report of LUT APC-OMS method-Area

V. CONCLUSION

We have suggested an efficient architecture for low-power, high-throughput, and low-area implementation of adaptive fir filter. Throughput rate is significantly enhanced by parallel LUT update and concurrent processing of filtering operation and weight-update operation. From the synthesis results, we find that the proposed design consumes less area and power over our previous DA-based FIR adaptive filter in average for different filter lengths. The design and implementation of a Digital audio broadcasting system (DABS) over a Field Programmable Gate Array (FPGA) platform for low bit rate voice communication using Audio codec has higher possibility of noises. Low bit rate voice communication has become essential to many applications like digital radio, satellite communication, underwater acoustic communication etc., where bandwidth is at a premium and voice intelligibility is imperative. For better audio quality, we are going to implement our proposed FIR with audio codec suitable for mobile communication and radio broadcasting.

REFERENCES

[1] B. Widrow and S. D. Stearns, Adaptive signal processing. Prentice Hall, Englewood Cliffs, NJ, 1985.

[2] GA Clark, SK Mitra and SR Parker, Block implementation of adaptive digital filters,IEEE Transactions on Acoustivs,Speech and Signal Processing, ASSP-29,1981.3,PP744-752.

[3] NR Shanbhag, KK Parhi,A pipelined adaptive lattice filter architecture, IEEE Transactions on Signal Processing,1993.41,PP1925-1939.

[4] NR Shanbhag, KK Parhi, Relaxed look-ahead pipelined LMS adaptive filters and their application to ADPCM coder. , IEEE Transactions on Circuits and Systems-II, 1993.40,PP753-766.

[5] SA White. Applications of distributed arithmetic to digital signal processing. A tutorial review. IEEE ASSP Magazine, July1989.

[6] P. K. Meher and M. Maheshwari, "A high-speed FIR adaptive filter architecture using a modified delayed LMS algorithm," in Proc. IEEE Int. Symp. Circuits Syst., May 2011, pp. 121–124.

[7] P. K. Meher and S. Y. Park, "Low adaptation-delay LMS adaptive filter part-I: Introducing a novel multiplication cell," in Proc. IEEE Int. Midwest Symp. Circuits Syst., Aug. 2011, pp. 1–4.

[8] P. K. Meher, 'LUT Optimization for Memory-Based Computation,' IEEE Trans on Circuits & Systems-II, pp.285-289, April 2010.

[9] P. K. Meher, "New approach to LUT implementation and accumulation for memory-based multiplication,"May 2009.

[10] P. K. Meher, "New look-up-table optimizations for memory-based multiplication," in Proc. ISIC, Dec. 2009.