ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 8, August 2015

FPGA Implementation of Memory Efficient DA-Based LMS Adaptive Filter

D.Venkatesh[1], S.Sruthi[2]

PG Student, Dept. of ECE, Shree Institute of Technical Education, Tirupati, A.P, India[1]

Assistant Professor, Dept. of ECE, Shree Institute of Technical Education, Tirupati, A.P, India[2]

ABSTRACT*:* In this paper, an productive implementation of adaptive filter is presented which reduces the area and power consumed by the Least Mean Square (LMS) algorithm. To achieve the lesser area and power efficient the memory based structures are replaced with the MAC units. The LUTs are memory based units used for the design of the filter. Distributed Arithmetic is a bit serial-computational action and uses parallel look up tables apprise and equal implementation of filtering and weight updated operations to appliance high throughput filter rates irrespective of the filter length. The least mean square algorithm adaption is functioned to update the weight and abate the mean square the error between the assessed and chosen output. The weight increment block based adders and a subtractor cell is exchanged by carry save adder in order to reduce area difficulty. It comprises of multiplexers, small LUTs and practically half number of adders contrasted to the Distributed Arithmetic based design.

KEYWORDS: Adaptive filter, circuit optimization, Distributed Arithmetic (DA), Least Mean Square algorithm (LMS).

I. INTRODUCTION

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. Recently, software defined radio (SDR) application has increased the demand for reconfigurable communication systems capable of multi standard operations. 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 [1].

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. Now a day's 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 [3].

In general, Digital filters are divided into two categories, including Finite Impulse Response and Infinite Impulse Response (IIR). FIR filters are widely applied to a variety of digital signal processing areas for the virtues of providing linear phase and system stability. The FPGA-based FIR filters using traditional direct arithmetic costs considerable multiply-and-accumulate (MAC) blocks with the augment of the filter order[4] - [5]. However, according to Distributed Arithmetic, we can make a Look-Up-Table (LUT) to conserve the MAC values and callout the values according to the input data if necessary. Therefore, LUT can be created to take the place of MAC units so as to save the hardware resources. This paper provide the principles of Distributed Arithmetic, and introduce it into the FIR filters design, and then presents a 31-order FIR low-pass filter using Distributed Arithmetic, which save considerable MAC blocks to decrease the circuit scale, meanwhile, divided LUT method is used to decrease the required memory units and pipeline structure is also used to increase the system speed [2].

DOI: 10.15662/ijareeie.2015.0408032

II. ADAPTIVE FILTER

An adaptive filter is a computational device that attempts to model the relationship between two signals in real time in an iterative manner. Adaptive filters are often realized either as a set of program instructions running on an arithmetical processing device such as a microprocessor or DSP chip, or as a set of logic operations implemented in a field-programmable gate array or in a semicustom or custom VLSI integrated circuit. However, ignoring any errors introduced by numerical precision effects in these implementations, the fundamental operation of an adaptive filter can be characterized independently of the specific physical realization that it takes. For this reason, we shall focus on the mathematical forms of adaptive filters as opposed to their specific realizations in software or hardware. By choosing a particular adaptive filter structure, one specifies the number and type of parameters that can be adjusted.

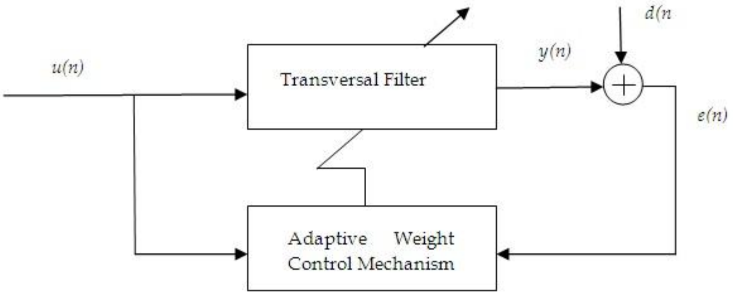


Fig.1: Block diagram of adaptive filter

The adaptive algorithm is used to update the parameter values of the system can take on a myriad of forms and is often derived as a form of optimization procedure that minimizes an error criterion that is useful for the task at hand. In this section, we present the general adaptive filtering problem and introduce the mathematical notation for representing the form and operation of the adaptive filter. We then discuss several different structures that have been proven to be useful in practical applications. We provide an overview of the many and varied applications in which adaptive filters have been successfully used finally, we give a simple derivation of the least-mean-square (LMS) algorithm, which is perhaps the most popular method for adjusting the coefficients of an adaptive filter, and we discuss some of this algorithm's properties. As for the mathematical notation used throughout this section, all quantities are assumed to be real-valued. Scalar and Vector quantities shall be indicated by lowercase (x) and uppercase-bold (X) letters, respectively. We represent scalar and vector sequences or signals as x (n) and X (n), respectively, where n denotes the discrete time or discrete spatial index, depending on the application [12].

Here w represents the coefficients of the FIR filter tap weight vector, x(n) is the input vector samples, z^{-1} is a delay of one sample periods, y(n) is the adaptive filter output, d(n) is the desired echoed signal and e(n) is the estimation error at time n. The aim of an adaptive filter is to calculate the difference between the desired signal and the adaptive filter output, e(n). This error signal is fed back into the adaptive filter and its coefficients are changed algorithmically in order to minimize a function of this difference, known as the cost function. In the case of acoustic echo cancellation, the optimal output of the adaptive filter is equal in value to the unwanted echoed signal. When the adaptive filter output is equal to desired signal the error signal goes to zero. In this situation the echoed signal would be completely cancelled and the far user would not hear any of their original speech returned to them.

III. EXISTING SYSTEM

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 *antisymmetric product coding* (APC) approach, the LUT size can also be reduced to half, where the product words are recoded as antisymmetric pairs. The APC approach, although providing a reduction in LUT size by a factor of two, incorporates substantial overhead of area and time to perform the two's complement operation of LUT output for sign modification and that of the input operand for input mapping.

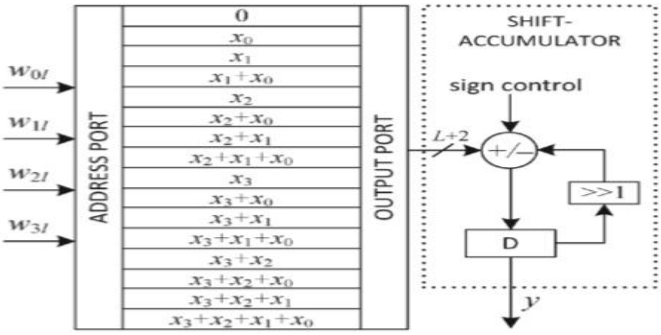


Fig.2: Conventional DA-based implementation of four-point inner product

However, we find that when the APC approach is combined with the OMS technique, the two's complement operations could be very much simplified since the input address and LUT output could always be transformed into odd integers. However, the OMS technique in cannot be combined with the APC scheme since the APC words generated according to odd numbers. Moreover, the OMS scheme does not provide an efficient implementation when combined with the APC technique. In this brief, we therefore present a different form of APC and combined that with a modified form of the OMS scheme for efficient memory based multiplication [4].

IV. PROPOSED SYSTEM

The proposed structure of DA-based adaptive filter of length N = 4 is shown in Fig. 3. It consists of a four-point inner product block and a weight-increment block along with additional circuits for the computation of error value $e(n)$ and control word t for the barrel shifters. The four-point inner-product block includes a DA table consisting of an array of 15 registers which stores the partial inner products y_l for $0 < l \le 15$ and a 16 : 1 multiplexor (MUX) to select the content of one of those registers. Bit slices of weights $A = \{w_{3l}\ w_{2l}\ w_{1l}\ w_{0l}\}$ for $0 \le l \le L - 1$ are fed to the MUX as control in LSB-to- MSB order, and the output of the MUX is fed to the carry-save accumulator. After L bit cycles, the carry-save accumulator shift accumulates all the partial inner products and generates a sum word and a carry word of size (L + 2) bit each. The carry and sum words are shifted added with an input carry "1" to generate filter output which is subsequently subtracted from the desired output $d(n)$ to obtain the error $e(n)$ [4].

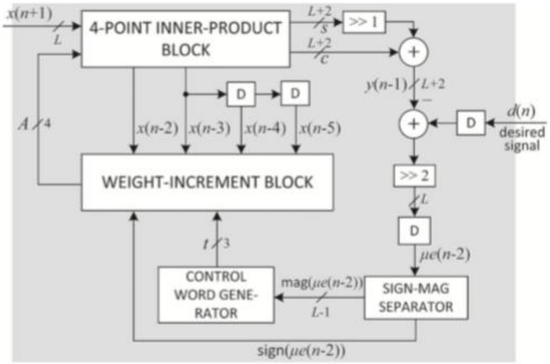


Fig. 3: Proposed structure of DA-based LMS adaptive filter

IMPLEMENTATION OF THE LMS ALGORITHM

Each iteration of the LMS algorithm requires 3 distinct steps in this order:

1. The output of the FIR filter, y (n) is calculated using equation 3.

$$y(n) = \sum_{i=0}^{N-1} w(n)x(n-i) = \mathbf{w}^T(n)\mathbf{x}(n) \quad \text{(eq. 3)}$$

2. The value of the error estimation is calculated using equation 4.

$$E(n) = d(n) - y(n) \quad \text{(eq. 4)}$$

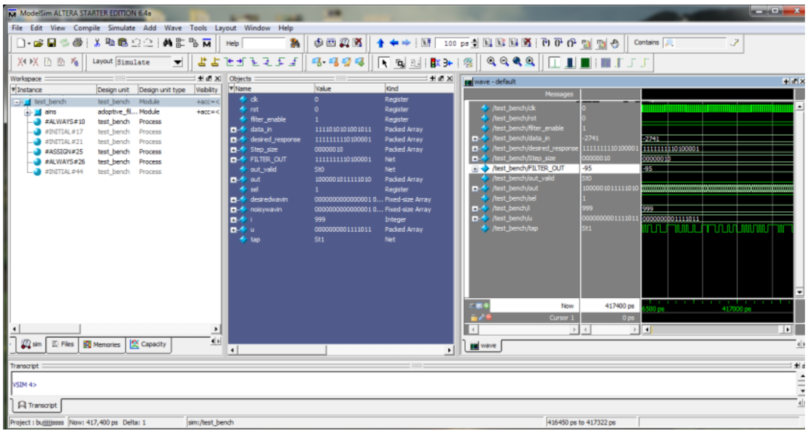
3. The tap weights of the FIR vector are updated in preparation for the next iteration, by equation 5.

$$\mathbf{w}(n+1) = \mathbf{w}(n) + 2\mu e(n)\mathbf{x}(n) \quad \text{(eq. 5)}$$

The main reason for the LMS algorithms popularity in adaptive filtering is its computational simplicity, making it easier to implement than all other commonly used adaptive algorithms. For each iteration the LMS algorithm requires 2N additions and 2N+1 multiplications (N for calculating the output, y(n), one for 2µe(n) and an additional N for the scalar by vector multiplication) [4].

V. RESULTS AND DISCUSSION

The simulation waveform shown below is the desired output response for the conventional DA based FIR filter. The output is simulated using Xilinx Model sim simulator based on ALTERA stater edition 6.4 a. and is shown in figure 4.



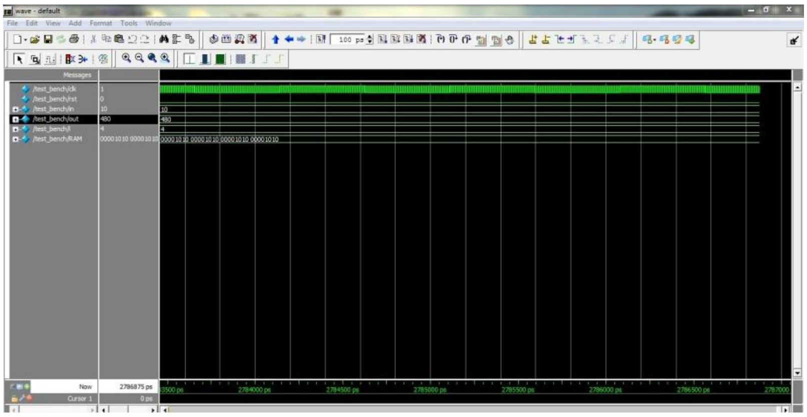
Name	Value	Kind
clk	0	Register
rst	0	Register
filter_enable	1	Register
data_in	1111010101001011	Packed Array
desired_response	1111111110100001	Packed Array
Step_size	00000010	Packed Array
FILTER_OUT	1111111110100001	Net
out_valid	St0	Net
out	1000001011111010	Packed Array
sel	1	Register
desiredwavin	0000000000000001 0...	Fixed-size Array
noisywavin	0000000000000001 0...	Fixed-size Array
i	999	Integer
u	0000000001111011	Packed Array
tap	St1	Net

```
/test_bench/clk              0
/test_bench/rst              0
/test_bench/filter_enable    1
/test_bench/data_in          -2741
/test_bench/desired_response 1111111110100001
/test_bench/Step_size        00000010
/test_bench/FILTER_OUT       -95
/test_bench/out_valid        St0
/test_bench/out              1000001011111010
/test_bench/sel              1
/test_bench/i                999
/test_bench/u                0000000001111011
/test_bench/tap              St1
Now                          417400 ps
Cursor 1                     0 ps
```

ModelSim ALTERA STARTER EDITION 6.4a
File Edit View Compile Simulate Add Wave Tools Layout Window Help
Workspace
Objects
wave - default
Messages
Transcript
VSIM 4>

Fig. 4: FIR Filter

The figure 5 shows the simulation results obtained for the proposed DA based LMS adaptive filter, which is extension work of conventional DA based LMS filter. The results obtained are satisfactory due to the replacement of MACs instead of memory based units.



```
/test_bench/clk   1
/test_bench/rst   0
/test_bench/in    10
/test_bench/out   480
/test_bench/i     4
/test_bench/RAM   00001010 00001010
Now               2786875 ps
Cursor 1          0 ps
```

Fig. 5: Distributed Arithmetic based LMS Adaptive Filter

Fig .6: LMS output on Spartan 3E Kit

The figure 6 shows the proposed LMS output on the Spartan 3E FPGA board. The improvement and optimization of DA algorithm has made the operation speed faster and occupy less memory to obtain satisfactory results.

VI. CONCLUSION

We have suggested an efficient pipelined architecture for low-power, high-throughput, and low-area implementation of DA-based LMS adaptive filter. Throughput rate is significantly enhanced by parallel LUT update and concurrent processing of filtering operation and weight-update operation. From the simulation the LMS adaptive filter has the reduced size LUT when compared to conventional DA based adaptive filter and the memory efficiency of LMS adaptive filter is verified using Spartan 3E FPGA kit.

REFERENCES

[1] S. Haykin and B. Widrow, *Least-Mean-Square Adaptive Filters*. Hoboken, NJ, USA: Wiley, 2003.

[2] M. Keerthi, Vasujadevi Midasala, S Nagakishore Bhavanam, Jeevan Reddy K, "FPGA Implementation Of Distributed Arithmetic For FIR Filter," in International Journal of Engineering Research & Technology on Vol.1 - Issue 9 (November - 2012).

[3] S. A. White, “Applications of the distributed arithmetic to digital signal processing: A tutorial review,” *IEEE ASSP Mag.*, vol. 6, no. 3, pp. 4–19, Jul. 1989.

[4] Sang Yoon Park and Pramod Kumar Meher, "Low-Power, High-Throughput, and Low-Area Adaptive FIR Filter Based on Distributed Arithmetic," in Circuits and Systems II: Express Briefs, IEEE Transactions on (Volume:60 , Issue: 6), pp. 346-350.

[5] 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.

[6] 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.

[7] R. Guo and L. S. DeBrunner, “A novel adaptive filter implementation scheme using distributed arithmetic,” in *Proc. Asilomar Conf. Signals, Syst., Comput.*, Nov. 2011, pp. 160–164.

[8] P. K. Meher and S. Y. Park, “High-throughput pipelined realization of adaptive FIR filter based on distributed arithmetic,” in *VLSI Symp. Tech. Dig.*, Oct. 2011, pp. 428–433.

[9] M. D. Meyer and 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.

[10] P. K. Meher, Jul. 2008, “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.

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

[12] Douglas, S.C. “Introduction to Adaptive Filters”Digital Signal Processing HandbookEd. Vijay K. Madisetti and Douglas B. WilliamsBoca Raton: CRC Press LLC, 1999.