International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2013): 6.14 | Impact Factor (2013): 4.438

Distributive Arithmetic Formulation For An Optimized Adaptive Filter Design

Ashly Babu[1], Ajeesh A.V.[2]

[1]P G Scholar, VLSI and Embedded Systems, Department of ECE, T K M Institute of Technology, Kollam, India

[2]Assistant Professor, Department of ECE, T K M Institute of Technology, Kollam, India

Abstract: ***Distributed arithmetic (DA) is commonly used for signal processing algorithms where calculating the inner product of two vectors comprises most of the computational workload. An important signal processing area is adaptive filtering. Adaptive filtering is extensively used in several signal processing applications including signal de-noising, and channel equalization for communication and networking systems. But, DA has its issues also. The main problem here is the updating of the memory table. Several methods have been adopted to accelerate memory updating, but it had led to additional memory usage and convergence speed. Hence it is necessary to develop structures for an adaptive DA filter with the maximum reduction of these disadvantages. Various methods can be adopted to achieve this result. One among them is parallel lookup table (LUT) update and concurrent implementation of filtering and weight-update operations. The DA-based inner-product computation can be done by conditional signed carry-save accumulation instead of conventional adder-based shift accumulation and a fast bit clock for carry-save accumulation but a much slower clock for all other operations can also be adopted. The coding of each module is simulated and synthesized using the Xilinx ISE Design Suite 12.1 and ISim Simulator.***

Keywords: Adaptive filter, distributed arithmetic (DA), least mean square (LMS) algorithm, Finite Impulse Response(FIR), Field Programmable Gate Array(FPGA).

1. Introduction

LMS based adaptive filters are preferred for most of the DSP applications. When we are doing the direct form configuration of the filters it leads to long critical path because of the inner-product computation to get the filter output[1]. Hence for high sampling rate, the critical path of structure should not exceed the sampling period. So, we go for Distributed Arithmetic (DA).

DA is basically a bit serial computational operation that forms an inner (dot) product of a pair of vectors in a single direct step. The advantage of DA is its efficiency of mechanization. One of the major disadvantages of it is the slowness because of its bit serial nature [2]. This disadvantage is not real if the number of elements in each vector is in proportion with the number of bits in each vector element. For example the time required to input eight 8-bit words one at a time in a parallel fashion is exactly the same as the time required to input all eight words serially. Other modifications to increase the speed can be done by employing techniques such as bit pairing or partitioning the input words into the most significant half and least significant half ,the least significant half of the most significant half, etc.,thereby introducing parallelism in the computation. Another major disadvantage is the large memory requirement [3]. So when it comes to employ filters with high sampling rate it becomes a serious issue.

Various methods can be adopted to avoid these disadvantages .One among them is parallel lookup table (LUT) update and concurrent implementation of filtering and weight-update operations. The DA-based inner-product computation can be done by conditional signed carry-save accumulation [8] instead of conventional adder-based shift accumulation. A fast bit clock for carry-save accumulation but a much slower clock for all other operations can also be adopted. The coding can be synthesized by the Xilinx ISE Design Suite 12.1, simulated using ISim simulator and can be implemented using Spartan 3E FPGA

2. Theory

2.1. Adaptive FIR Filter

An adaptive filter is a filter that adjusts its transfer function all by itself according to an adaptive algorithm. This is based on the error signal produced which is the difference between the desired output of the filter and the actual output of the filter. Most adaptive filters are digital filters due to the complexity of the optimization algorithm. A non-adaptive filter has a fixed transfer function. Adaptive filters are necessary for some appliances because some parameters of the desired processing operation are not known in advance. The adaptive filter uses feedback in the form of an error signal to improve its transfer function to match the varying parameters.

Due to increase in the power of digital signal processors, adaptive filters have become much more common and are now regularly used in devices such as mobile phones and other communication devices, digital cameras, and medical monitoring equipment.

Licensed Under Creative Commons Attribution CC BY

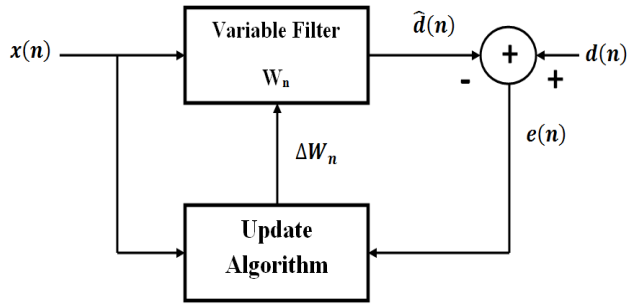

Figure 1: Adaptive FIR Filter Structure

The block diagram, shown above, is an example of a foundation for particular adaptive filter realizations, such as Least Mean Squares (LMS).The block diagram here indicates that a variable filter can be made to extract an estimate of the desired signal.

2.2. Least Mean Square (LMS) Algorithm

The LMS algorithm is an adaptive algorithm which adapts the coefficients of FIR filters iteratively. The following steps are followed to find out the coefficients of an adaptive FIR filter:

1. Calculate the output signal **y(n)** of the FIR filter

$$y(n) = \bar{u}^{T}(n).\bar{w}(n) \quad (1)$$

Where,
$\bar{u}(n)$ is the filter input vector and
$\bar{u}(n) = [x(n)x(n-1) \ldots x(n-N+1)]^{T}$
$\bar{w}(n)$ is the filter coefficients vector and
$\bar{w}(n) = [w_0(n)w_1(n) \ldots w_{n-1}(n)]^{T}$

2. Calculate the error signal **e(n)** by using the following equation:

$$\mathbf{e(n) = d(n) – y(n)} \quad (2)$$

Where,
d(n) is the desired output
y(n) is the filter output

2.3. Distributed Arithmetic

DA is a multiplier-less implementation process .It can be used to compute the inner-product of a pair of vectors which is a common computation method used in digital signal processing. They are most suitable for implementing high throughput FIR filters. A is a bit-serial computation. It forms an inner product of a pair of vectors in a few steps by storing all possible combination sums of weights in a memory table. The advantage of DA is its efficiency of mechanization. The output y is given as the sum of delayed and scaled input samples x[k].

$$y = \sum_{k=1}^{N-1} w_k x_k \quad (3)$$

Where,
w_k is the fixed coefficients
x_k is the input data words
Also,

$$w_k = -w_{k0} + \sum_{l=1}^{L-1} w_{kl} 2^{-l} \quad (4)$$

Where,
w_{kl} are the bits 0 or1
w_{k0} is the sign bit
Substituting (4) in (3) ;

$$y = -\sum_{k=0}^{N-1} x_k w_{k0} + \sum_{k=0}^{N-1} x_k .[\sum_{l=1}^{L-1} w_{kl} \ 2^{-l}] \quad (5)$$

In-order to produce an distributed form, the order of summations are interchanged.

$$y = -\sum_{k=0}^{N-1} x_k w_{k0} + \sum_{l=1}^{L-1} 2^{-l}.[\sum_{k=0}^{N-1} x_k .w_{kl}] \quad (6)$$

Hence the output is computed as

$$y = [\sum_{l=1}^{L-1} 2^{-l}. y_l] - y_0 \quad (7)$$

3. Methodology

Distributed Arithmetic (DA) is a different methodology for implementing digital filters. The basic idea of this method is to use a DA table and a shifter accumulator as a substitute for all multiplications and additions. DA is a bit-serial computational operation which allows digital filters to be implemented with high throughput rates, despite of the filter length.

3.1 System Overview

The basic block diagram for the design of an Adaptive FIR Filter using Distributive Arithmetic is as shown below in figure 2.

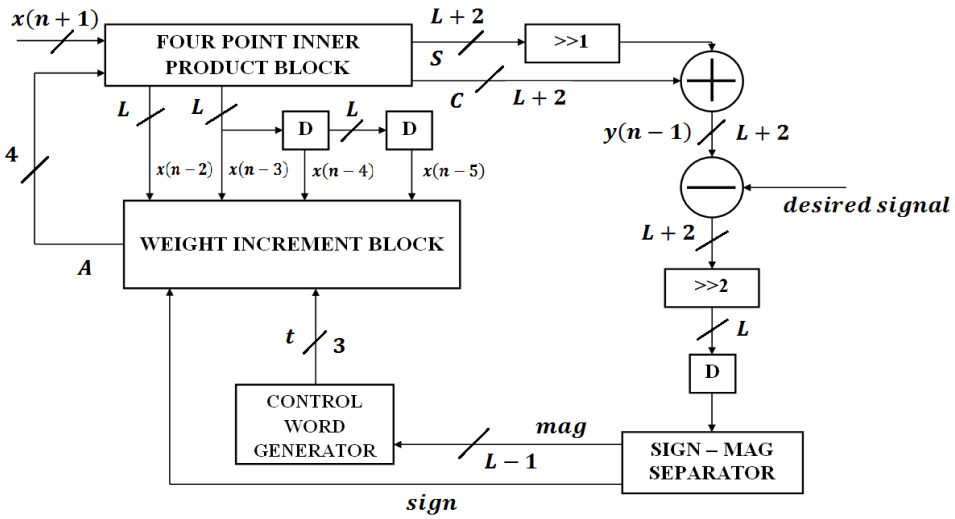

Figure 2: DA-based LMS Adaptive Filter

Licensed Under Creative Commons Attribution CC BY

The main parts of the block diagram are as follows:

1. Four-Point Inner product block, having;
   - Distributed Arithmetic table
   - 16-to-1 MUX
   - Carry Save Accumulator
2. Sign- Magnitude Separator
3. Control Word Generator
4. Weight-Increment Block, having;
   - Barrel shifters
   - Adder/subtractor blocks

3.2 Four-Point Inner product block

The initial part of the filtering process of the LMS adaptive filter, for each cycle, is the need to perform an inner-product computation. This is the task that contributes to most of the critical path. The block diagram of four-point inner product block is as shown in figure 3.

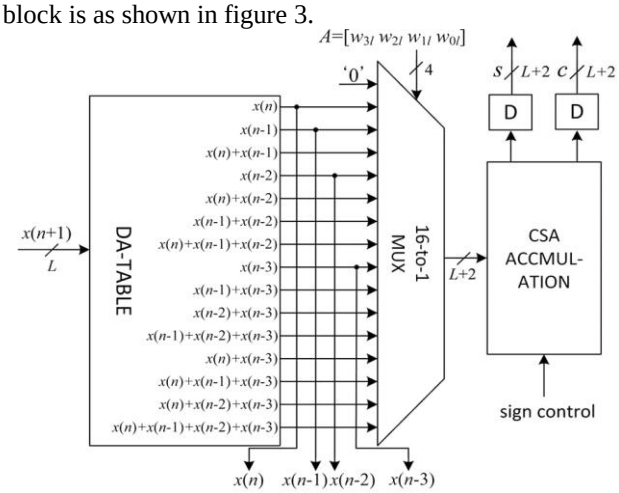

Figure 3: Four-point inner product block

3.2.1. DA Table

It forms the initial part of the 4-point inner product block, such that it consists of an array of 15 registers. They are used for the purpose of storage of partial inner products y. The structure of the DA table is as shown in figure 4.

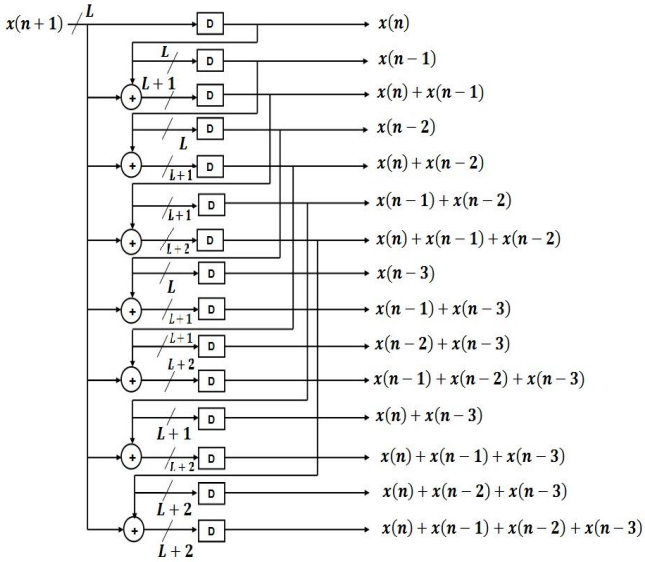

Figure 4: DA Table

3.2.2. 16-to-1 MUX

It is used to select the contents of the registers of the DA table. For the MUX the bit slices of weights A= {w_{3l} w_{2l} w_{1l} w_{0l}} are fed as control or select lines to draw the contents of the DA table. The control here is in the LSB-to-MSB order. And then the output of it fed to the carry save accumulator.

3.2.3. Carry Save Accumulator:

The process of shift accumulation is done in the CSA block. The input of the block is from the 16-to-1 MUX. The bit slices are fed one after the other in the order LSB-to- MSB to the CSA block. For MSB slices, negative of LUT output accumulation is very much necessary, so the LUT output is passed through XOR gates and sign control input. Hence, when MSB slice occurs the XOR gate produce 1's complement as the sign control bit is set to 1. And then the sum and carry are obtained after L clock cycles. The structure of carry-save accumulator is as shown in figure 5.

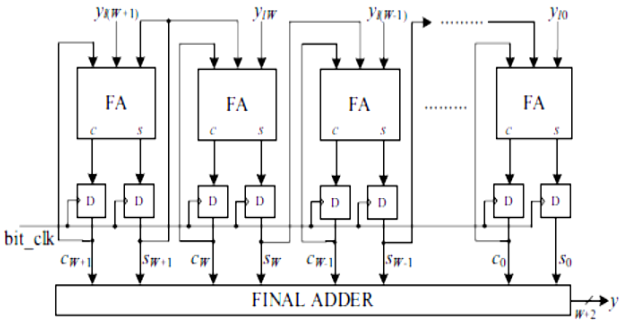

Figure 5: Carry-Save Accumulator

3.3. Weight Increment block

It forms one of the important parts of the filter design structure. The production of the weight for the updating of four-point inner product block is done by the weight increment block. The block diagram of weight increment block is as shown in figure 6.

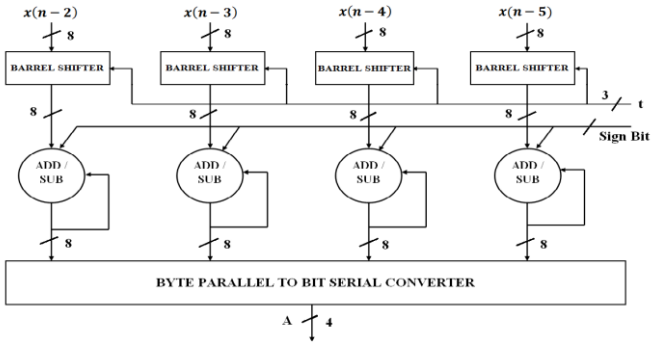

Figure 6: Weight Increment block

3.3.1. Barrel Shifter

A barrel shifter is a digital circuit that can shift a data word by a specified number of bits in one clock cycle. The control word 't' for the working of the BS is produced by decoding the magnitude of error. This error that is decomposed is the difference between the desired output of the filter and the actual output produced by the filter. The logic for the

Licensed Under Creative Commons Attribution CC BY

selection for the control-word 't' for the BS is given in the table below.

if r6=1 then *t* = "000"; **else if r5 = 1 then *t* = "001";** **else if r4 = 1 then *t* = "010";** **else if r3 = 1 then *t* = "011";** **else if r2 = 1 then *t* = "100";** **else if r1 = 1 then *t* = "101";** **else if r0 = 1 then *t* = "110";** **else then *t* = "111"**

3.3.2. Adder/subtractor block:

In adder/subtractor is a digital circuit that is capable of adding or subtracting numbers. The circuit does the adding or subtracting process depending on a control signal. When Sign Bit = '0', the circuit perform addition and when Sign Bit = '1' the circuit perform subtraction.

3.4. Flowchart

The flowchart explaining the whole filtering process is shown in the Figure 7.

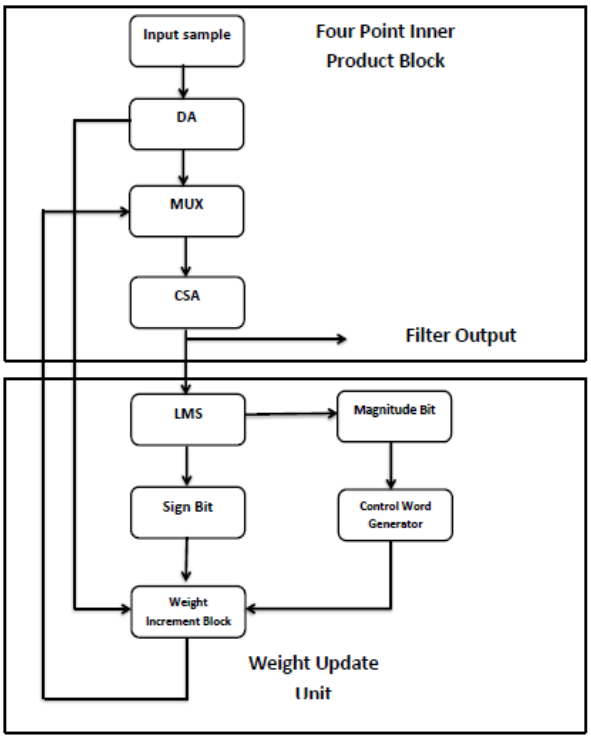

Figure 7: Flow chart describing the process

3.5. Algorithm

The algorithm that briefly explains the process that happens inside the block diagram for the design of the adaptive FIR filter using Distributed Arithmetic is as follows:

Step 1: The input of four-point inner product block is 8-bit samples.

Step 2: The four-point inner-product block includes DA Table, 16:1 Multiplexer and Carry Save Accumulation block.

Step 3: The DA table consist of an array of 15 registers which stores the partial inner products.

Step 4: It is connected to 16:1 MUX, so that the output of DA table is fed to the MUX.

Step 5: Weights A= $\{w_{3l}\ w_{2l}\ w_{1l}\ w_{0l}\}$ are fed to the MUX as control or selection input to select any one of the inputs from those 16 inputs (i.e.; the output of the MUX).

Step 6: The output of the MUX is then fed to the Carry Save Accumulator.

Step 7: The carry-save accumulator shift accumulates all the partial inner products.

Step 8: The output of CSA block is Sum and Carry bits.

Step 9: It is then shifted and added to generate filter output y(n).

Step 10: Filter output is subsequently subtracted from the desired output d(n) to obtain the error e(n).

Step 11: The magnitude of the computed error is decoded to generate the control word t for the barrel shifters

Step 12: In weight increment block, the barrel shifters yields the desired increments that have to be fed to the adder/subtractor block.

Step 13: The sign bit of the error signal is used as the control signal for adder/subtractor block.

Step 14: The 8-bit output from the adder/subtractor block is fed to the parallel bit to serial nibble converter to convert it into 4-bit, which is then fed to the inner product block to complete the filtering process.

4. Results and Discussion

The filter design process has the following blocks like four-point inner product block, sign-magnitude separator, control word generator and weight-increment block. Initially the coding for the individual block was completed and then all the modules were combined by using the structural modeling. Simulation is carried out using Xilinx ISim simulator and then synthesis is done using Xilinx ISE 12.1.

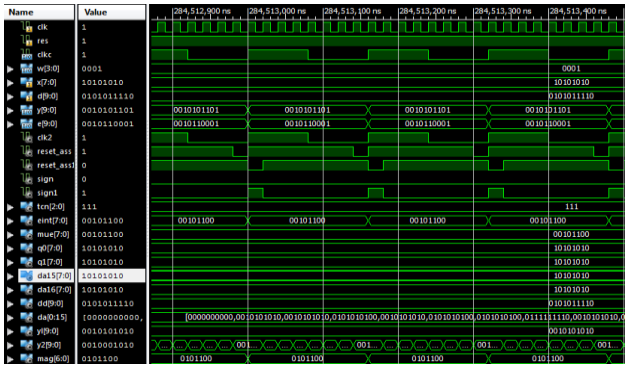

Figure 8: Adaptive FIR Filtering using DA

The simulation result of the Adaptive FIR Filtering using DA is shown in Figure 8 .A DA table was used instead of LUTs for the calculation of the partial-inner products. One of the contents of the table was then selected using a multiplexer and input to a carry save accumulator block. The output produced was subtracted from a desired signal to obtain the error. It was then decomposed to obtain the control and sign signals to run the weight increment block that is there to adjust weight so as to get the output similar to the desired signal.

Licensed Under Creative Commons Attribution CC BY

5. Conclusion

For an adaptive FIR filter, the contents of the memory table must be updated regardless of the filter coefficients or input samples present. So an efficient method for fully updating a memory table that is composed of the combinations of the input samples and that is addressed by concatenating bits of the filter coefficients need to be developed. So that it is able to support high sampling rate.

Distributive Arithmetic (DA) is one such method. While comparing DA with other alternative methods it was observed that they require fewer arithmetic computing resources and also no multipliers. This concept of distributed arithmetic is the most favored one for computing environments with limited computational resources. The method used had their disadvantages too. They needed additional memory usage and convergence speed. So it's necessary to develop structures for an adaptive DA filter with the maximum reduction of these disadvantages. Various methods like usage of auxiliary LUTs, parallel look up tables, concurrent implementation of the filtering and weight updating operations, replacement of the adder-based shift accumulation by conditional signed carry-save accumulation were adopted.

An Adaptive FIR Filter using Distributed Arithmetic has a lot of applications. The main ones among them are noise cancellation and channel equalization. Adaptive noise cancellation is one of the best approaches that can be used for the purpose of separating the additive noise from the corrupted speech. These applications can be implemented on FPGAs.

References

[1] P. K. Meher and S. Y. Park, "Low-Power, High-Throughput, and Low-Area Adaptive FIR Filter Based on Distributed Arithmetic" IEEE Transactions On Circuits And Systems VOL. 60, NO. 6, JUNE 2013.
[2] S. Haykin and B.Widrow,Least-Mean-Square Adaptive Filters. Hoboken, NJ, USA: Wiley, 2003.
[3] S. A. White,"Applications of the distributed arithmetic to digital signal processing: A tutorial review" IEEE ASSP Mag. vol. 6, no. 3,Jul. 1989.
[4] 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,Jul. 2005.
[5] Walter G. Huang,"Thesis on Implementation of Adaptive Digital Fir and Reprogrammable Mixed-Signal Filters Using Distributed Arithmetic".School of Electrical and Computer Engineering Georgia Institute of Technology December 2009.
[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, 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
[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.
[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., vol. 9, no.3, August 2013.
[10] Aarti Sharma, "Report on VLSI Implementation Of Pipelined FIR Filter".Thapar University Patiala.

Licensed Under Creative Commons Attribution CC BY