International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 3 Issue 2, February - 2014

Adaptive FIR Filter based on Distributed Arithmetic and LMS Algorithm for Low-Area and Low-Power

K. Jebin Roy
M.E. VLSI Design
Anand Institute of Higher Technology
Affiliated to Anna University,
Chennai – 603103, India

R. Ramya
M.Tech. VLSI Design,
Sathyabama University,
Accredited by NAAC,
Chennai – 600119, India

***Abstract*— In this manuscript, we proposed a novel pipelined architecture for low-power and low-area adaptive FIR filter based on distributed arithmetic (DA) and LMS algorithm. DA is bit-serial computational process and uses parallel look-up table (LUTs) apprise and equivalent implementation of filtering and weight-update operations to appliance high throughput filter rates irrespective of the filter length. The full adder based conditional signed carry save accumulation for DA-based inner product computation is replaced and design by using 10 transistor full adder based carry save accumulation of shift accumulation, with the intention of the proposed design, which can reduce the area complexity and power consumption. The least-mean-square (LMS) algorithm adaptation is functioned to update the weight and abate the mean square error between the assessed and chosen output. The weight increment block based adder/subtractor cells is swapped by carry save adder in order to reduce area difficulty. It comprises of multiplexors, smaller LUT, and practically half the number of transistors compared to the present DA-based design.**

Index Terms: Adaptive Filter, Distributed Arithmetic (DA), Finite Impulse Response (FIR), Least Mean Square (LMS) Algorithm, Lookup table (LUT).

I. INTRODUCTION

Adaptive filters find extensive use in many signal processing applications such as channel equalization, echo cancellation, noise cancellation [1]. The finite impulse response (FIR) filters whose weights are updated by the famous Widrow-Hoff least mean square (LMS) algorithm is the most popularly used adaptive filter not only due to its simplicity but also due to its satisfactory convergence performance [5]. The direct form configuration on the onward path of the FIR filter results in a long critical path due to an inner product computation to obtain a filter output. Consequently, it is required to reduce the critical path of the structure if the input signal has high sampling rate. By reducing the critical path of the structure, thereby, the critical path could not exceed the sampling period.

Distributed arithmetic (DA) is so named because it performed arithmetic operation. DA is bit serial computation in nature and it eliminates the need for hardware multipliers and is capable of implementing large order filters with very high throughput. A lot of study has been done to implement the DA based adaptive FIR filter for area efficient design, the multiplier-less distributed arithmetic (DA) based technique has achieved plenteous popularity for its high throughput, but it results are increased in cost-effective, area and time efficient computing structures [8]. DA based hardware efficient adaptive FIR filter inner product has been suggested by Allred *et al.* [2] using two separate lookup tables (LUTs) Filtering lookup table and Auxiliary lookup table for filtering and weight updating module. Later, Guo and DeBrunner [3], [4] have improved the design structure in [2] by using only one lookup table instead of two LUTs for both filter and weight updating module. On the other hand, the design process in [2], [3], [4] and [8] require more cycles for lookup table (LUT) update for each new sample, hence it do not support high sampling rate. Meher and Park have improved the design with low adaptation delay for high speed DA based adaptive filter [6]. In a recent paper, Meher and Park proposed a new DA based adaptive filter architecture for low power, low area and high throughput with very low adaptation delay [7].

This brief proposes an adaptive FIR filter using distributed arithmetic for area efficient design. High Throughput is achieved by using a parallel lookup table update and equivalent implementation of filtering and weight-updating operations. The conditional signed carry saved accumulation for DA-based inner product computation is designed by using 10 transistor full adder based carry saved accumulation of shift accumulation. The use of the proposed design helps to reduce the area complexity and power consumption.

In the next section, a brief study of the least mean square (LMS) adaptive algorithm, followed by the description of the proposed DA based technique filter in Section 3. The structure of the proposed adaptive filter and description of the proposed DA based adaptive FIR filter in Section 4. Results and Conclusions are given in Section 5 and 6.

II. REVIEW OF LMS ADAPTIVE ALGORITHM

The LMS algorithm computes a filter output and an error value that is equal to the difference between the current

filter output and the desired response for every clock cycle. In every training cycle, the estimated error is then used to update the filter weights. The weights of LMS adaptive filter during the *n*th iteration is updated according to the following equations [6]:

$$w(n+1) = w(n) + \mu \,.\, e(n) \qquad (1a)$$

Where

$$e(n) = d(n) - y(n) \qquad (1b)$$

$$y(n) = w^{qT}(n) \,.\, x(n) \qquad (1c)$$

The input vector $x(n)$ and the weight vector $w(n)$ at the *n*th training iteration are respected given by

$$x(n) = [x(n), x(n-1), \dots, x(n-N+1)]^T \qquad (2a)$$

$$w(n) = [w_0(n), w_1(n), \dots, w_{N-1}(N)]^T \qquad (2b)$$

$d(n)$ is the desired response, and $y(n)$ is the filter output of the nth iteration. $e(n)$ denotes the error value generated during the nth iteration, which is used to update the weights, μ is the convergence factor, and N is the filter length.

In the case of filter designs, the feedback error $e(n)$ becomes available after certain number of cycles, called the "adaptation delay". The pipelined architectures therefore use the delayed error $e(n-m)$ for updating the current weight instead of the most recent error, where m is the adaptation delay. The weight update equation of such delayed LMS adaptive filter is given by

$$w(n+1) = w(n) + \mu \,.\, e(n-m) \,.\, x(n-m) \qquad (3a)$$

III. Proposed DA-Based Approach for Inner Product Computation

In each cycle, the LMS adaptive filter needs to perform an inner-product computation which contributes to the most of the critical path. Let the inner product computation of (1c) be given by

$$y = \sum_{k=0}^{N-1} r_k \,.\, s_k \qquad (4)$$

Where r_k and s_k for $0 \le k \le N-1$ form the N – point vectors corresponding to the current weights and most recent $N-1$ input respectively. Let us assume L be the bit width of the weight, every component of the vector weight may be expressed in 2's complement representation

$$r_k = -r_{k0} + \sum_{l=1}^{L-1} r_{kl} \,.\, 2^{-l} \qquad (5)$$

Where w_{kl} denotes the lth bit of r_k. Substituting (5), we can write (4) in an expanded form

$$y = -\sum_{k=0}^{N-1} s_k \,.\, r_{k0} + \sum_{k=0}^{N-1} s_k \,.\, [\sum_{l=1}^{L-1} r_{kl} \,.\, 2^{-l}] \qquad (6)$$

To convert the sum-of-product form of (4) into a distributed form, the order of summations over the indices k and l in (6) can be interchanged to have

$$y = -\sum_{k=0}^{N-1} s_k \,.\, r_{k0} + \sum_{l=0}^{L-1} 2^{-l} \,.\, [\sum_{k=0}^{N-1} s_k \,.\, r_{kl}] \qquad (7)$$

and the inner product given by (7) can be computed as

$$y = [\sum_{l-1}^{L-1} 2^{-l} \,.\, y_l] - y_0 \,,\; y_l = \sum_{k=0}^{N-1} s_k \,.\, r_{kl} \qquad (8)$$

Meanwhile any element of the N-point bit sequence $\{r_{kl}\ for\ 0 \le k \le N-1\}$ can either be 1 or 0, the partial sum y_l for $l = 0, 1, \dots, L-1$ can have 2^N possible values. If the entire 2^N possible values sum y_l are precomputed and stored in a LUT, the partial sum y_l can be read out from the LUT using the bit sequence $\{r_{kl}\}$ as address bits for computing the inner product.

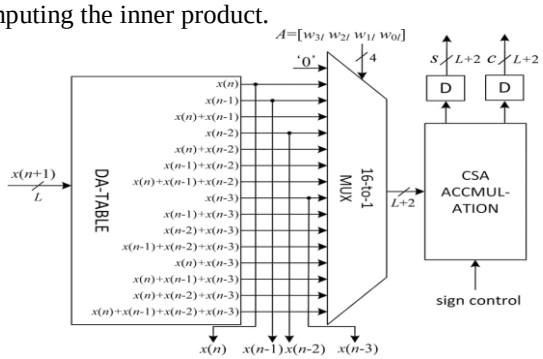

Figure 1: DA-based implementation of four point inner product

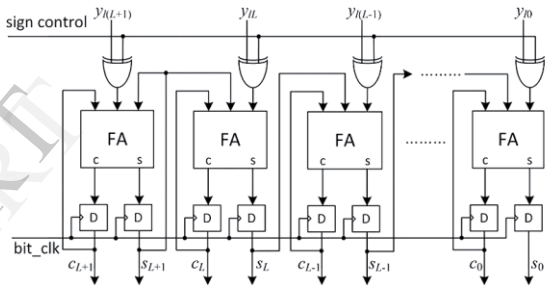

Figure 2: Carry save implementation of shift accumulation

The inner product of (8) can therefore be calculated in L cycles of carry save implementation of shift accumulation, followed by LUT-read operations corresponding to L number of bit slices $\{r_{kl}\}$ for $0 \le l \le L-1$, as shown in Fig. 1. Since the carry save implementation of shift accumulation in Fig. 2 required more area and power consumption.

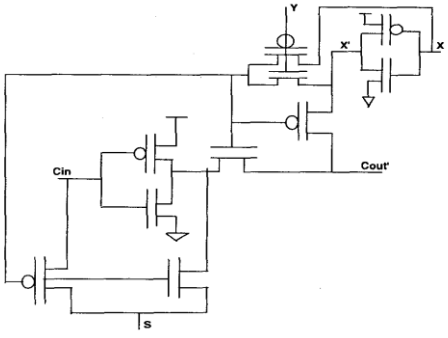

Figure 3: 10T 1-Bit Full Adder

The carry save implementation of shift accumulation based full adder is design by using 10 transistor one bit-full adder [9] as shown in Fig. 4. The bit slices of vector r are fed one after the next in the LSB to the MSB order to the carry save

accumulator. Finally, the sum and carry output of the carry save accumulator is obtained after L clock cycle are required to be added by a final adder. The content of the kth LUT location can be expressed as

$$c_k = \sum_{j=0}^{N-1} x_j \,.\, k_j \qquad (9)$$

where k_j is the $(j+1)$th bit of the N - bit binary representation of integer k for $0 \le k \le 2^N - 1$ can be precomputed and stored in RAM based LUT of 2^N words. However, instead of storing 2^N words in LUT, we store $(2^N - 1)$ words in a DA table of $(2^N - 1)$ registers.

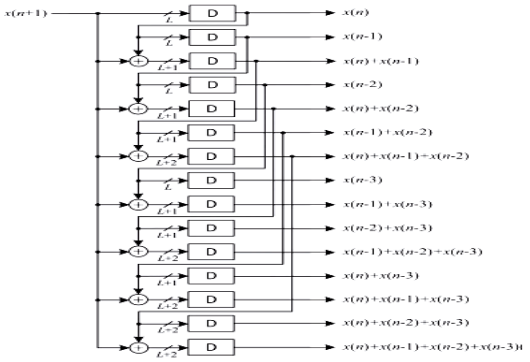

Figure 5: Distributed arithmetic table

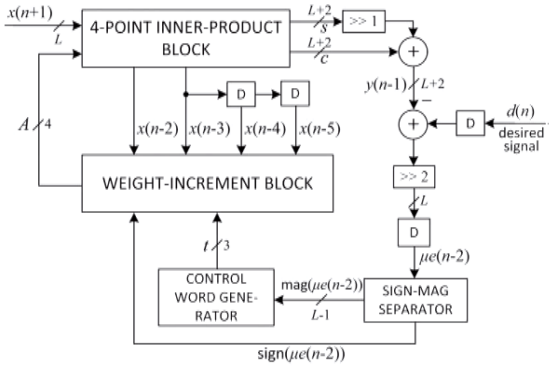

Figure 6: Proposed structure of DA-based LMS adaptive filter length $N = 4$

DA table for N=4 is shown in Fig. 5. DA table contains only 15 registers to store the precalculated sums of input words. In DA table, seven new values of c_k are computed by seven adders in parallel.

IV. PROPOSED STRUCTURE OF ADAPTIVE FIR FILTER

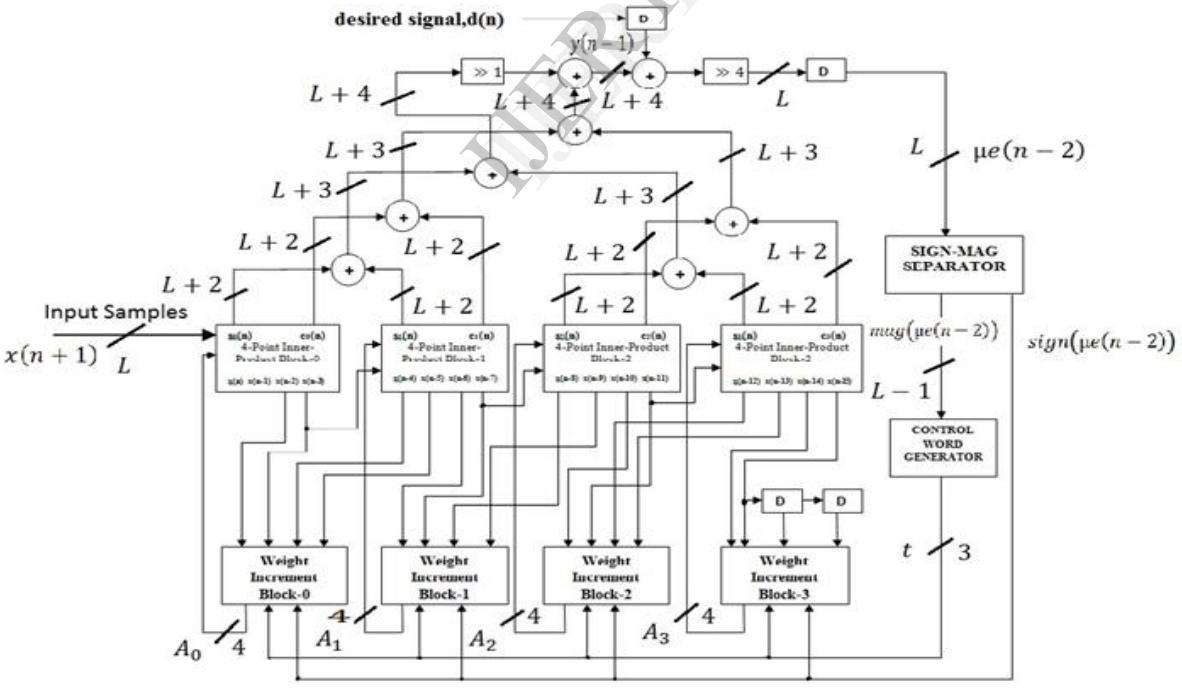

Figure 7: Proposed structure of DA-based LMS adaptive filter of length N=16

A straight-forward DA-based implementation of inner product requires LUT of very large size. For that reason, the computation of the inner products of large orders needs to be decomposed [4] into small adaptive filtering blocks as shown in Fig. 6 and large order adaptive filters shown in Fig. 7.

The structure of DA-based adaptive filter of length N=4 comprises 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 [shown in Fig. 1] contains a DA table consisting of an array of 15 registers as shown in Fig. 5 which stores the partial inner products y_l for $0 < l \le 15$ and a 16:1 multiplexor to select the content of one of those registers from the DA table. 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 using 10T full adder as shown in Fig. 4. After L bit cycles, the carry save accumulator shift accumulates all the partial inner products and generates a sum and carry output 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 $e(n)$.

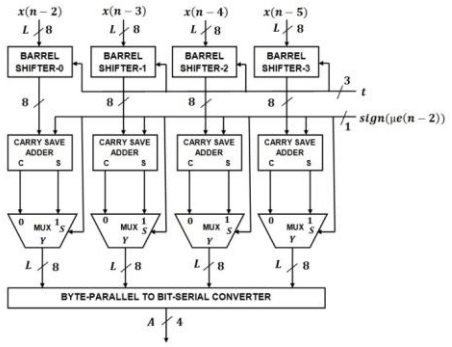

Figure 8: Structure of the weight-increment block for $N = 4$

```
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";
```

Figure 9: Logic used for generation of control word t for the barrel shifter for $L = 8$

As in the case in [4], all the bits of the error except the most significant bit (MSB) one are ignored (8^{th} Bit). The remaining bits are magnitude of the error, the magnitude of the computed error is decoded to generate the control word t for the barrel shifter. The logic used for the generation of control word t for the barrel shifter is shown in Fig. 9. The number of shifts t in that case is increased by i locations accordingly to reduce the hardware complexity. The weight increment unit as shown in Fig. 8 for $N = 4$ comprises of 4 barrel shifters and four carry save adder cells. The barrel shifter shifts the different input values x_k for k = 0, 1, 2… $N - 1$ by appropriate number of locations. The barrel shifter yields the desired increments are fed to the carry save adder with the sign bit from the error value. The sign bit of the error is used as the control for the 2:1 MUX to select any one of the sum or carry output from the Carry save adder. The output of the MUX is fed to the Byte-parallel to Bit-serial converter to convert 8 bit data into 1 bit data. The output waveform of DA-based adaptive FIR filter (N=16) as shown in Fig. 10.

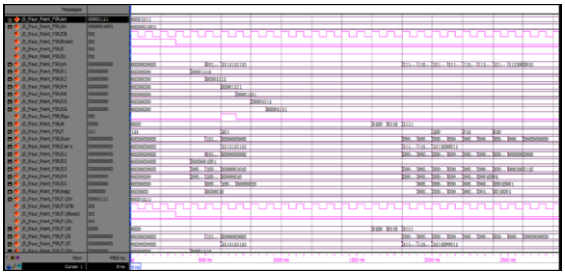

Figure 10: DA-based LMS adaptive FIR filter of length N=4

V. RESULTS

Thus the existing and proposed designs in [7] and [8] are implemented in Xilinx 14.1 using verilog code. Along with area and power of corresponding design are measured using Tanner 15.1 EDA in 250nm CMOS technology.

Table 1: Implementation Results Using Xilinx 14.1 and Tanner 15.1

Designs	Filter Length	Area (sq.μs)	Power (mW)
Existing	N = 16	18264	9.41
Proposed	N = 16	14520	6.40

VI. CONCLUSION

In this script, an adaptive FIR filter using distributed arithmetic (DA) for area efficient design is implemented. High throughput is drastically enriched by parallel (LUTs) update and equivalent implementation of filtering and weight-update operations. The proposed carry save accumulation using 10 transistor full adder schemes of signed partial inner products for the computation of the filter output and also modified in weight increment block. By this way it utilizes low area, low power consumption and the throughput of the filter rates increases irrespective of the filter length.

REFERENCES

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

[2] 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.

[3] 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.

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

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

[6] 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.

[7] Sang Yoon Park, Member, IEEE, and Pramod Kumar Meher, Senior Member, IEEE "Low-Power, High-Throughput, and Low-Area Adaptive FIR Filter Based on Distributed Arithmetic" IEEE Transactions On Circuits And Systems—Ii: Express Briefs, Vol. 60, No. 6 June 2013.

[8] 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.

[9] Hanan A.Mahmoud and Magdy.A. Bayoumi, " A 10-Transistor Low Power High Speed Full Adder Cell".