ISSN 2395-695X (Print)
ISSN 2395-695X (Online)
International Journal of Advanced Research in Biology Engineering Science and Technology (IJARBEST)

Vol. 2, Special Issue 10, March 2016

IMPLEMENTATION OF AN ADAPTIVE FIR FILTER USING HIGH SPEED DISTRIBUTED ARITHMETIC

Thangamonikha.A[1], Dr.V.R.Balaji[2]
[1]PG Scholar, Department OF ECE, [2]Assitant Professor, Department of ECE
[1, 2] Sri Krishna College of Engineering and Technology, Kuniamuthur P.O., Coimbatore-641008, Tamil Nadu, India.
[1]monikhaarun@gmail.com, [2]vrbalaji@skcet.ac.in

Abstract— **Distributed Arithmetic (DA) is a different methodology for implementing digital filters and it is a multiplier-less implementation process. 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. This paper presents about an implementation of an adaptive FIR filter based on Distributed Arithmetic (DA) to achieve low power, low area and to increase the speed of the circuit. The sampling period could be substantially reduced by using carry-save accumulation instead of shift-accumulation for DA-based inner-product implementation. This also reduces the area complexity. The proposed system involves parallel Lookup table for weight updating operations and half the number of registers compared to the existing DA-based design to store the sum of different combinations of input samples. Reduction of power consumption is achieved in the proposed design by using a fast bit clock for carry-save accumulation but a much slower clock for all other operations. It involves same number of multiplexers, smaller LUT and nearly half the number of adders compared to the existing DA-based design.**

Index terms— FIR Filter, Distributive Arithmetic (DA), Carry Sava Accumulation, Inner product.

1 .INTRODUCTION

1.1 FIR

In signal processing, a finite impulse response (FIR) filter is a filter whose impulse response (or response to any finite length input) is of finite duration, because it settles to zero in finite time. This is in contrast to infinite impulse response (IIR) filters, which may have internal feedback and may continue to respond indefinitely (usually decaying). The impulse response of an Nth-order discrete-time FIR filter (i.e., with a Kronecker delta impulse input) lasts for N+ 1 samples, and then settles to zero. FIR filters can be discrete-time or continuous-time and digital or analog.

Digital filters that have an impulse response which reaches zero in a finite number of steps are (appropriately enough) called Finite Impulse Response (FIR) filter. An FIR filter can be implemented non-recursively by convolving its impulse response (which is often used to define an FIR filter) with the time data sequence it is filtering. FIR filters are somewhat simpler than Infinite Impulse Response (IIR) filters, which contain one or more feedback terms and must be implemented with difference equations or some other recursive technique. They can easily be designed to be "linear phase" (and usually are). Put simply, linear-phase filters delay the input signal, but don't distort its phase. They are simple to implement. On most DSP processors, the FIR calculation can be done by looping a single instruction.

ISSN 2395-695X (Print)
ISSN 2395-695X (Online)
International Journal of Advanced Research in Biology Engineering Science and Technology (IJARBEST)
Vol. 2, Special Issue 10, March 2016

They are suited to multi-rate applications. By multi-rate, we mean either "decimation" (reducing the sampling rate), "interpolation" (increasing the sampling rate), or both. Whether decimating or interpolating, the use of FIR filters allows some of the calculations to be omitted, thus providing an important computational efficiency. In contrast, if IIR filters are used, each output must be individually calculated, even if it that output will be discarded (so the feedback will be incorporated into the filter). They have desirable numeric properties. In practice, all DSP filters must be implemented using "finite-precision" arithmetic, that is, a limited number of bits.

The use of finite-precision arithmetic in IIR filters can cause significant problems due to the use of feedback, but FIR filters have no feedback, so they can usually be implemented using fewer bits, and the designer has fewer practical problems to solve related to non-ideal arithmetic. They can be implemented using fractional arithmetic. Unlike IIR filters, it is always possible to implement a FIR filter using coefficients with magnitude of less than 1.0. (The overall gain of the FIR filter can be adjusted at its output, if desired.) This is an important consideration when using fixed-point DSP's, because it makes the implementation much simpler.

The "impulse response" of a FIR filter is actually just the set of FIR coefficients. (If you put an "impulse" into a FIR filter which consists of a "1" sample followed by many "0" samples, the output of the filter will be the set of coefficients, as the 1 sample moves past each coefficient in turn to form the output).A FIR "tap" is simply a coefficient/delay pair. The number of FIR taps, (often designated as "N") is an indication of

1. The amount of memory required to implement the filter.
2. The number of calculations required.
3. The amount of "filtering" the filter can do; in effect, more taps means more stop band attenuation, less ripple, narrower filters, etc.

In a FIR context, a "MAC" is the operation of multiplying a coefficient by the corresponding delayed data sample and accumulating the result. FIRs usually require one MAC per tap. Most DSP implement the MAC operation in a single instruction cycle. The band of frequencies between pass band and stop band edges. The narrower the transition band, the more taps are required to implement the filter. (A "small" transition band results in a "sharp" filter). The set of memory elements that implement the "Z^-1" delay elements of the FIR calculation.

A special buffer which is "circular" because incrementing at the end causes it to wrap around to the beginning, or because decrementing from the beginning causes it to wrap around to the end. Circular buffers are often provided by DSP microprocessors to implement the "movement" of the samples through the FIR delay-line without having to literally move the data in memory. When a new sample is added to the buffer, it automatically replaces the oldest one.

1.2 LMS

LMS based adaptive filters are preferred for most of the DSP applications. The goal of the adaptation is to adjust the characteristics of the filter through an interaction with the environment in order to reach the desired values. The operation of adaptive filters is based on the estimation of the

ISSN 2395-695X (Print)
ISSN 2395-695X (Online)
International Journal of Advanced Research in Biology Engineering Science and Technology (IJARBEST)
Vol. 2, Special Issue 10, March 2016

statistical properties of the signal in its environment, while modifying the value of its parameters in order to minimize a certain criterion function.

The criterion function may be determined in a number of ways, depending on the particular purpose of the adaptive filter, but usually it is a function of some reference signal. The reference signal may be defined as the desired response of the adaptive filter, and in that case the role of the adaptive algorithm is to adjust the parameters of the adaptive filter in such a way to minimize the error signal, which represents the difference between the signal at the output of the adaptive filter and the reference signal. 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. Hence for high sampling rate, the critical path of structure should not exceed the sampling period. So, we go for distributive arithmetic (DA).

1.3 DISTRIDUTED ARITHMETIC

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. 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. 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 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 is simulated using MODELSIM SE 10.1C.

2 .EXISTING SYSTEM

In recent years, the multiplier-less distributed arithmetic (DA)-based technique [7] has gained substantial popularity for its high-throughput processing capability and regularity, which result in cost-effective and area–time efficient computing structures. Hardware-efficient DA-based design of adaptive filter has been suggested by Allred *et al.* [1] using two separate lookup tables (LUTs) for filtering and weight update. Guo and DeBrunner [2], [3] have improved the design in [1] by using only one LUT for filtering as well as weight updating. However, the structures in [1]–[3] do not support high sampling rate since they involve several cycles for LUT updates for each new sample. In a recent paper, we have proposed an efficient architecture for high-speed DA-based adaptive

ISSN 2395-695X (Print)
ISSN 2395-695X (Online)
International Journal of Advanced Research in Biology Engineering Science and Technology (IJARBEST)
Vol. 2, Special Issue 10, March 2016

filter with very low adaptation delay [5].

Adaptive digital filters have tremendous applications in signal processing. According to the LMS algorithm, there is a delay in the feedback error for updating the weights which does not favor the pipeline implementation, when the sampling rate is high. [2] Have proposed the delayed LMS algorithm for pipeline application of LMS based ADF. In LMS algorithm, the adaptation step can be performed after a fixed delay only, in some practical situations. In such a cases the implemented algorithm is a modified version of LMS algorithm known as the delayed LMS (DLMS) algorithm. In DLMS the coefficient adaptation is performed after a delay. The work shows the conditions for convergence and estimates of convergence rate, both for the mean of the DLMS filter coefficients and for its excess mean square error. The only difference between the LMS and DLMS algorithm is that the correction term for updating the filter weights of the current iteration are computed from error of corresponding to the past iteration. Many methods have been proposed to implement BLMS based adaptive digital filters efficiently in systolic VLSI with minimum adaptation delay [2].

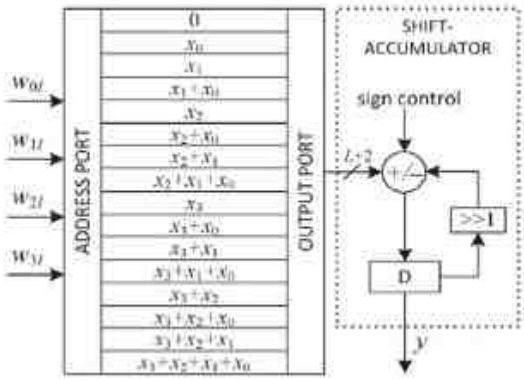


Fig 2.1. Conventional DA-based implementation four-point inner product

In order to avoid adaptation delay [3] has proposed a modified DLMS algorithm. In some of the applications of the adaptive finite impulse response filtering, the adaptation algorithm can be applied with a delay only in the coefficient update. This has a dissimilar effect on the convergence behavior of the algorithm. In this work it is shown how the delayed LMS algorithm can be converted into the typical LMS algorithm at simply slight increase in the computational expense. The modified DLMS is used by [4] to derive a systolic architecture but it requires large amount of hardware resources as compared to the earlier one. BLMS is useful for fast and computationally-efficient implementation of adaptive digital filters.

The convergence performance of BLMS ADFs and LMS ADFs are similar, but for block length L BLMS ADFs offers L fold higher throughput. Considering this, many BLMS algorithms like time and frequency domain block filtered-X LMS (BFXLMS) has been proposed for specific applications. Computationally more efficient BFXLMS using FFT and fast Hartley transform (FHT). A delayed block LMS algorithm and a concurrent multiplier-based design for high throughput pipeline execution of BLMS ADFs have been proposed. In a block LMS algorithm with delayed weight adaptation for hardware execution of FIR adaptive filters has been proposed. The delayed block least mean square algorithm take a block of L input samples and produces block of L output, in every training cycle. The simulation result shows that the DBLMS algorithm has convergence performance same as that of the DLMS algorithm. A highly synchronized systolic architecture for FIR adaptive filters has been derived. The suggested architecture can support L time higher sampling rate when compared with the other pipelined designs and hence include less samples of adaptation delays and would provide a more effective execution of LMS based adaptive filters. [6], [7] have suggested structure for FPGA implementation of BLMS ADFs based on distributed arithmetic. [6] Derived a design and implement a high throughput ADF using Fast Block Least Mean Squares (FBLMS) adaptive algorithm.

ISSN 2395-695X (Print)
ISSN 2395-695X (Online)
International Journal of Advanced Research in Biology Engineering Science and Technology (IJARBEST)
Vol. 2, Special Issue 10, March 2016

The structure of filter is built on Distributed Arithmetic. The structure calculate the inner product as: (i) shifting (ii) accumulating (iii) storing in look-up table. The desired adaptive digital filter obtained will be multiplier less. Hence a DA based execution of adaptive filter is area efficient. FPGA implementation results imitates that the proposed DA based adaptive filter in [6] can implement with meaningfully smaller area usage, (about 45%) less than that of the remaining FBLMS algorithm based adaptive filter. The structure in [10] replaces multiply-and accumulates operations with a series of look-up-tables (LUT).

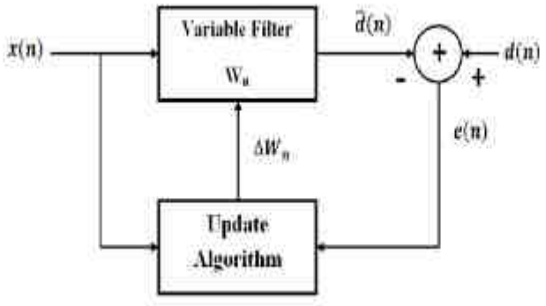
FPGA implementation results in [7] conforms that the suggested DA based adaptive filter can implement with expressively smaller area usage about 52% less than that of the existing FBLMS algorithm based adaptive filter applications. The structure in [4] for block LMS ADFs supports a very low sampling rate because it uses single multiply-accumulate cell for the computation of filter output and the weight increment term. DA uses bit-serial operations and LUTs to implement high throughput filters which uses simply about one cycle per bit of resolution irrespective of filter length. Though, building adaptive DA filters requires recalculation of the LUTs for every adaptation which can deny any performance advantages of DA filtering. With the help of an auxiliary LUT with distinctive addressing, the efficiency and throughput of DA adaptive filters can be made as same order as fixed DA filters. In this paper, a new hardware adaptive filter structure has been suggested for very high throughput LMS adaptive filters have described the development of DA adaptive filters and showed that practical executions of DA adaptive filters have very high throughput comparative to multiply and accumulate architectures also showed that DA adaptive filters have a potential area. The power consumption advantage over digital signal processing microprocessor architectures is also achieved.

3 .PROPOSED SYSTEM

3.1 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.

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.



3.2 LEAST MEAN SQUARE (LMS) ALGORITHM

ISSN 2395-695X (Print)
ISSN 2395-695X (Online)
International Journal of Advanced Research in Biology Engineering Science and Technology (IJARBEST)
Vol. 2, Special Issue 10, March 2016

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:

Fig 3.1. Adaptive FIR Filter Structure

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

$$y(n) = \bar{u}^T(n).\bar{w}(n) \quad (4.1)$$

Where,

$\bar{u}(n)$ is the filter input vector and

$$\bar{u}(n) = [x(n)\ x(n-1)\ldots\ldots x(n-N+1)]^T \quad (4.2)$$

$\bar{w}(n)$ is the filter co-efficient vector and

$$\bar{w}(n) = [w0(n)\ w1(n)\ldots.wn-1(n)]^T \quad (4.3)$$

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

$$E(n) = d(n) - y(n) \quad (4.4)$$

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

3.3. DISTRIBUTED ARITHMETIC

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. 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. It 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].

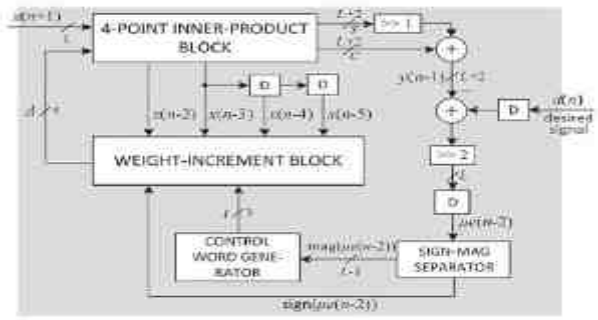
$$y = \Sigma(k)\ N-1K=1x\ (k) \quad (4.5)$$

Where,

w (k) is the fixed co-efficients

x (k) is the input data words

The basic block diagram for the design of an Adaptive FIR Filter using Distributive Arithmetic is as shown below in figure3.2 and 3.3.



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

ISSN 2395-695X (Print)
ISSN 2395-695X (Online)
International Journal of Advanced Research in Biology Engineering Science and Technology (IJARBEST)
Vol. 2, Special Issue 10, March 2016

Fig 3.2. DA based LMS Adaptive Filter N=4

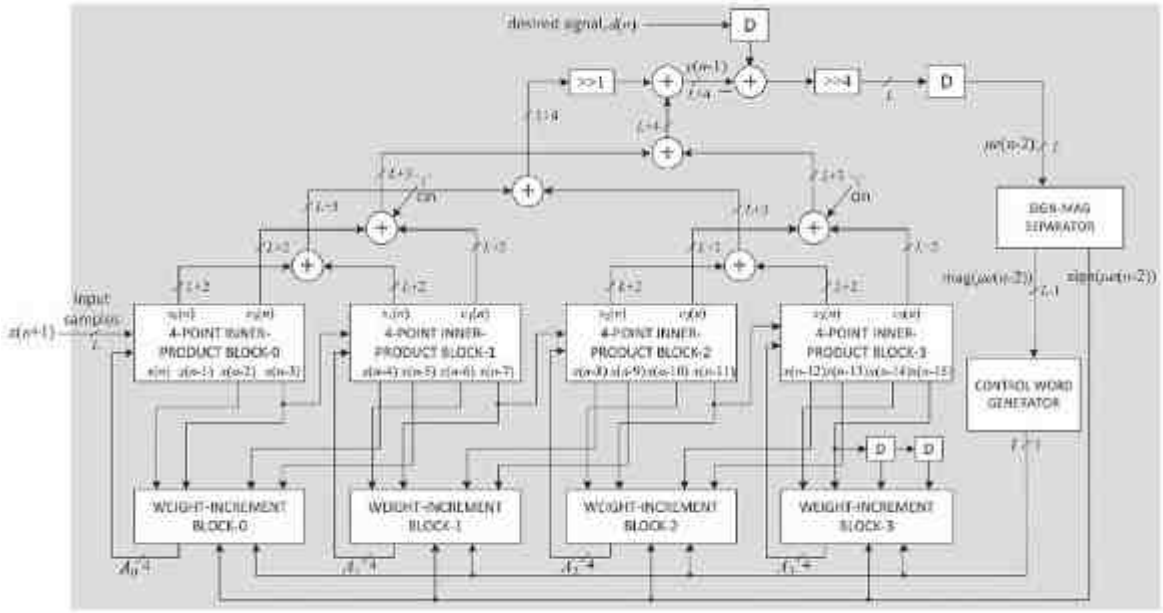
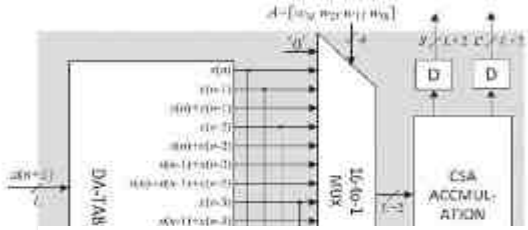


Fig 3.3. DA based LMS Adaptive Filter N=16

3.4. FOURPOINT 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 3.4.



3.4.1 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 3.5.

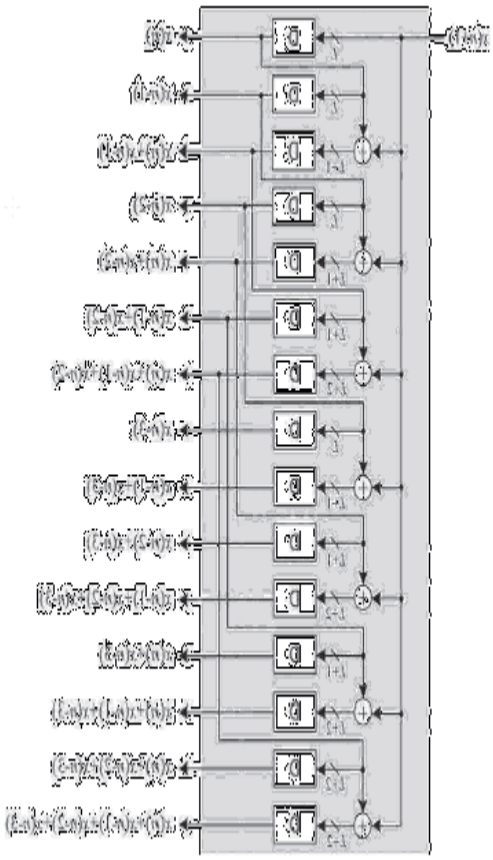


Fig 3.4. Structure of the four-point inner-product block

3.4.2. 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= {w3l, w2l, w1l, w0l} 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

ISSN 2395-695X (Print)
ISSN 2395-695X (Online)
International Journal of Advanced Research in Biology Engineering Science and Technology (IJARBEST)
Vol. 2, Special Issue 10, March 2016

the carry save accumulator.

3.4.3. 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 3.6.

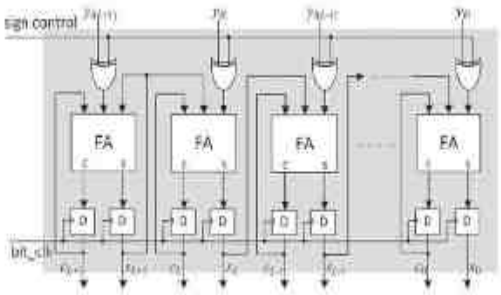


Fig 3.6. Carry-save implementation of shift accumulation

Fig 3.5. DA table for generation of possible sums of input samples

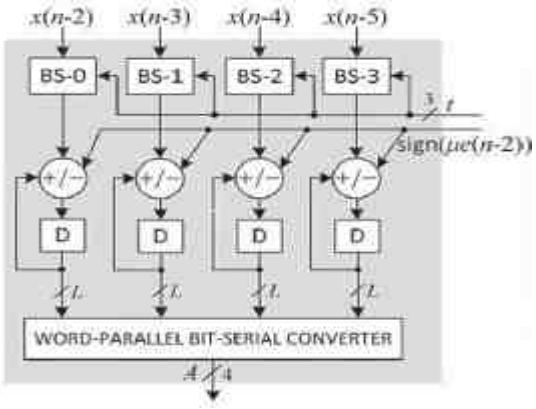
3.5. WEIGHT INCREMENT BLOCK

It forms one of the important parts of the filter design structure. The production of the weight for the updation 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 3.7.

3.5.1. 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 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"
```



ISSN 2395-695X (Print)
ISSN 2395-695X (Online)
International Journal of Advanced Research in Biology Engineering Science and Technology (IJARBEST)
Vol. 2, Special Issue 10, March 2016

Fig 3.7. Weight Increment Block

3.5.2. 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.

3.6. FLOWCHART

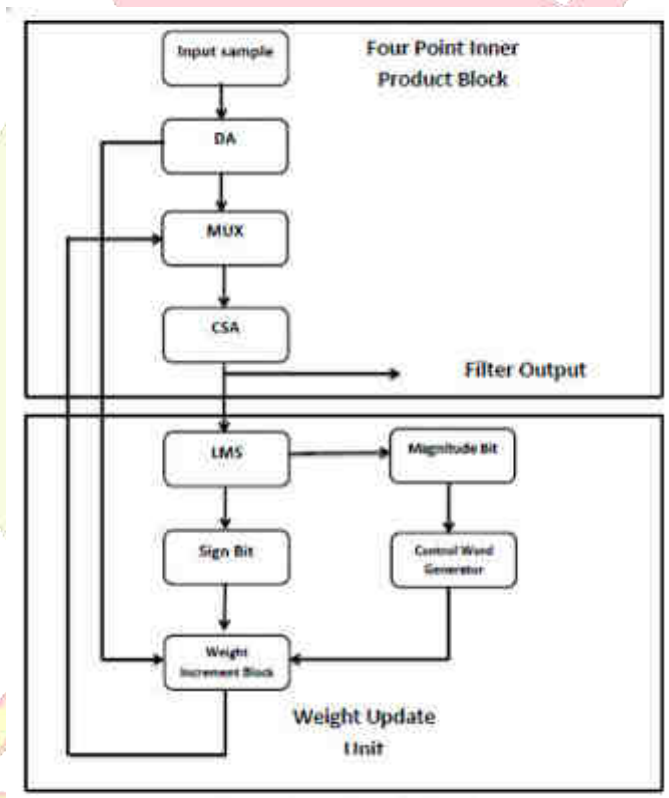


Fig 3.8. Filtering proces

4. RESULTS AND DISCUSSION

The synthesis was implemented with XILINX ISE 14.5 design suite, here all the three techniques were realized. The realization is mainly based on the delay optimization and the speed

ISSN 2395-695X (Print)
ISSN 2395-695X (Online)
International Journal of Advanced Research in Biology Engineering Science and Technology (IJARBEST)
Vol. 2, Special Issue 10, March 2016

improvement, the comparison was shown for all the three techniques with the existing and proposed system and the area reduction occurs in the few technique and the power remains same with the minor changes and they were mentioned with the thermal properties for the detail studies.

TABLE I

Synthesis Result Comparison

DESIGN	EXISTING	PROPOSED
Filter Length (N)	16	16
Minimum Sample Period (MSP) (ns)	12.92	9.927
Area Delay Product (sq.µm x ns)	26288	15189
Data Arrival Time (ns)	0.68	2.5
Power (mW)	9.41	5.05

5. CONCLUSION

The proposed technique suggests an efficient implementation of DA-based adaptive FIR filter to achieve low power, low area and to increase the speed of the circuit by reducing the delay. We have also proposed a carry-save accumulation scheme of signed partial inner products for the computation of filter output. From the synthesis results, we find that the proposed design consumes less power and less ADP over our previous DA-based FIR adaptive filter in average for filter length. Compared to the best of other existing designs our proposed design is better for area and power consumption. Thus the Distributed Arithmetic (DA) based implementation of an adaptive FIR filter have been simulated using MODELSIM software. The power analysis is done by Xilinx Design suite. The power is reduced and the delay is optimized comparing to the existing system.

REFERENCES

[1] V. and for *Reg*. pp.

D. J. Allred, H. Yoo, Krishnan, W. Huang, D. V. Anderson, "LMS adaptive filters using distributed arithmetic high throughput," *IEEE Trans. Circuits Syst. I, Papers*, vol. 52, no. 7, 1327–1337, Jul. 2005.

[2] 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.

[3] 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.

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

[5] 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.

[6] 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.

[7] 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.