ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com
International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 3, Special Issue 1, March 2016

DESIGN OF AN EFFICIENT ADAPTIVE FIR FILTER BY 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,vrbalaji[2]@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. Additional delay which occurs in carry save accumulation affects the overall clock period. This is avoided by replacing full adder in the carry save accumulation by half adder. It involves same number of multiplexers 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

Finite impulse response (FIR) is a filter whose impulse response (or response to any finite length input) is of finite period because it settles to zero in finite time. This is in contrast to infinite impulse response (IIR) filters, which might have internal feedback and may continue to respond indefinitely (usually decomposing). The impulse response of an Nth-order distinct-time FIR filter (i.e., with a Kronecker delta impulse input) lasts for N+ 1 samples, and after thatsettles to zero. FIR filterbe able to be discrete-time or continuous-time and digital or analog.

Digital filters so as to have an impulse response which reaches zero in a limited 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 frequently used to describe an FIR filter) with the time data sequence it is filtering. FIR filters are fairly simpler than Infinite Impulse Response (IIR) filters, which enclose one or more feedback terms and must be implemented with difference equations or several other recursive technique. They can easily be designed to be "linear phase".

They are matched to multi-rate applications. By multi-rate, we mean either "decimation" or "interpolation" or both. While decimating or interpolating, the use of FIR filters allows some of the calculations to be mislaid thus giving an central computational efficiency. In distinction, if IIR filters are worn each output must be separately calculated, even if it that output will be discarded (so the feedback will be included into the filter). They have enviable numeric properties. All DSP filters must be implemented using "finite-precision" sums, that is, a restricted number of bits.

The exploit of finite-precision arithmetic in IIR filters can source noteworthy problems due to the exercise of feedback, but FIR filters have veto feedback, so they can usually be implemented using fewer bits, and the designer has less sensible problems to solve associated non-ideal arithmetic. They can be implemented with fractional arithmetic. Unlike IIR filters, it is forever probable to implement a FIR filter using coefficients with scale of less than 1.0. (The general gain of the FIR filter can be adjusted at its output, if desired.) This is an

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com
International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 3, Special Issue 1, March 2016

vitaldeliberationwhile using fixed-point DSP's, since it makes the implementation much simpler.

The "impulse response" of FIR filter is in factset of FIR coefficients. (If y output an "impulse" is included into a FIR filter which consists of a "1" model followed by many "0" samples, the output of the filter will be set of coefficients, as 1 sample moves fast to form the output).A FIR "tap" is simply a coefficient/delay pair. The number of FIR taps, (often selected as "N") is an indication of

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

In a FIR context, a "MAC" is the operation of proliferating a coefficient by the equivalent delayed data sample and accumulating the result. FIRs usually require one MAC per tap. Most DSP complete the MAC operation in a solitary instruction cycle.

The band of frequencies between pass band and stop band edges. Narrower the transition band, more taps are required to implement the filter. (A "small" transition band fallout 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" increments at the end causes it to wrap around to the beginning, or causes it to wrap around to the end. Circular buffers are frequentlyafforded by DSP microprocessors to execute the "movement" of the samples during the FIR delay-line without having to exactly move the data in memory. When a new sample is added to the buffer, it mechanically replaces the oldest one.

1.2 LMS

LMS based adaptive filters are favored for most of the DSP applications. The goal of the adaptation is to adjust the uniqueness of the filter through an interface with the environment in order to reach the desired values. The operation of adaptive filters is based on the evaluation of the numerical properties of the signal in its environment, whereas amending the value of its parameters with the aim of minimizing a certain criterion function.

The criterion function may be resolute in a number of behavior, depending on the particular purpose of the adaptive filter, but usually it is a role of a few reference signal. The reference signal may be defined as the desired response of the adaptive filter, and in that case the task 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 do the straight form configuration of the filters it leads to long critical path sincethe inner product computation to acquire the filter output. Hence for high sampling rate, the critical path of arrangement should not surpass the sampling period. So, we go for distributive arithmetic (DA).

1.3 DISTRIDUTED ARITHMETIC

DA is essentially a bit serial computational process that forms an inner (dot) product of a pair of vectors in a single straight step. The benefit of DA is its effectiveness of mechanization. One of the major disadvantages of it is the slowness because of its bit serial character This drawback is not factual if the number of elements in each vector is in proportion with the amount of bits in each vector component For example the time required to input eight 8-bit words one at a time in a analogous fashion is precisely the same as the time needed to input all eight words serially.

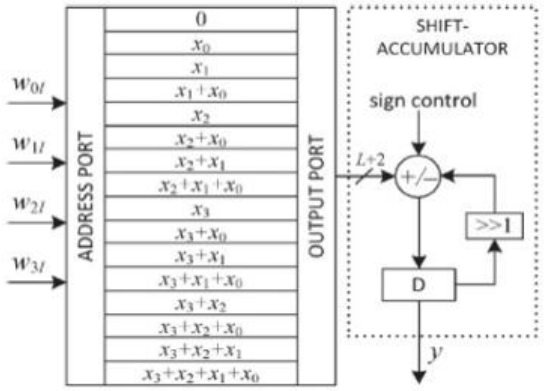
Other modifications to extend the speed can be done by utilizing techniques such as bit pairing or partitioning the input words into the most important half and least importanthalf thereby introducing parallelism in the calculation. One morechiefweakness is the large memory requirement. So when it comes to employ filters with high sampling rate it becomes a solemn matter.

Different methods can be approved to avoid these shortcomings .One amid them is parallel lookup table (LUT) update and synchronized execution of filtering and weight-update actions. The DA-based inner-product computation can be made by provisional signed carry-save accumulation as an alternative of conventional adder-based shift accumulation. A fast bit clock for carry-save accumulation but a large amountof deliberate clock for all other operations can also be approved. The coding is simulated using MODELSIM SE 10.1C.

2. EXISTING SYSTEM

In current years, the multiplier-less distributed arithmetic (DA)-based technique [7] has gained substantial popularity for its high-throughput dealing potentiality and reliability, which result in cost-effective and area–time efficient computing arrangements. Hardware-efficient DA-based devise

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com
International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 3, Special Issue 1, March 2016



of adaptive filter has been suggested by Allred *et al.* [1] using two partition lookup tables (LUTs) for filtering and weight updation. Guo and DeBrunner [2], [3] have improved the design in [1] by using merely one LUT for filtering and weight updating. However, the structures in [1]–[3] do not support high sampling rate as they engage numerous cycles for LUT updates for each new sample. In a recent paper, we have proposed an competent architecture for high-speed DA-based adaptive filter with very low adaptation delay [5].

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

Adaptive digital filters have incredible relevance in signal processing. According to the LMS algorithm, there is a holdup in the feedback fault for updating the weights which does not favor the pipeline implementation, when the sampling rate is high. [2] have projected the deferred LMS algorithm for pipeline application of LMS based ADF. In LMS algorithm, the alteration step can be executed after a fixed delay only, in some practical situations. In such cases the executed algorithm is a customized version of LMS algorithm known as the delayed LMS (DLMS) algorithm. In DLMS the coefficient adaptation is executed following a delay. The work shows the conditions for convergence and approximates the convergence rate, for the mean of the DLMS filter coefficients and for its excess mean square fault. The solitary variation between the LMS and DLMS algorithm is that the modification term for updating the filter weights of the existing iteration are calculated from error equivalent to the past iteration. Many methods have been proposed to execute BLMS based adaptive digital filters competently in systolic VLSI with minimum adaptation delay [2].

In order to evade adaptation delay [3] has projected a adapted DLMS algorithm. In some of the applications of the adaptive finite impulse response filtering, the adaptation algorithm be capable of applied with a delay only in the coefficient update. This has a disparate result on the convergence actions of the algorithm. In this work it is shown how the delayed LMS algorithm is transformed into the distinctive LMS algorithm at simply slight increase in the computational expense. The adapted DLMS used by [4] to obtain a systolic design but it requires large amount of hardware resources as compared to the former one. BLMS is helpful for rapid and computationally-competent implementation of adaptive digital filters.

The convergence recital of BLMS ADFs and LMS ADFs are similar, but for block length L BLMS ADFs offers L fold higher throughput. Taking into account numerous BLMS algorithms like time and frequency domain block filtered-X LMS (BFXLMS) has been planned for explicit applications. Computationally more efficient BFXLMS using FFT and fast Hartley transform (FHT). A waited block LMS algorithm and a parallel multiplier-based design for high throughput pipeline execution of BLMS ADFs have been projected. In a block LMS algorithm with holdup weight adaptation for hardware execution of FIR adaptive filters has been projected. The delayed block least mean square algorithm take a block of L input samples and produces block of L output, in each training cycle. The recreation result shows that the DBLMS algorithm has convergence performance equal to that of the DLMS algorithm. A greatly coordinated systolic architecture for FIR adaptive filters has been derived. The optional architecture can bear L time higher sampling rate when compared with the other pipelined designs and thus comprise 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 employ a lofty throughput ADF by quick Block Least Mean Squares (FBLMS) adaptive algorithm.

FPGA realization in [7] matches that the optional DA based adaptive filter can implement with expressively lesserregionpractice about 52% less than that of the presented 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 calculation of filter production and the weight increment term. DA uses bit-serial operations and LUTs to realize high throughput filters which uses merely one cycle per bit of resolution irrespective of filter length. Although, constructing

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com
International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 3, Special Issue 1, March 2016

adaptive DA filters needs recalculation of the LUTs for every adaptation which can deny any performance benefits of DA filtering. With the help of an supplementary LUT with distinctive addressing, the efficiency and throughput of DA adaptive filters can be completed as similar order as permanent DA filters. In this paper, a new hardware adaptive filter configuration has been recommended for extremely high throughput LMS adaptive filters have described the development of DA adaptive filters and explained that practical finishings of DA adaptive filters have very high throughput comparative to multiply and accrue designs also showed that DA adaptive filters have a potential area.

3 .PROPOSED SYSTEM

3.1 ADAPTIVE FIR FILTER

An adaptive filter is a filter that regulates 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 definite output of the filter. Majority of adaptive filters are digital filters due to the complexity of the optimization algorithm. A non-adaptive filter has a preset transfer function. Adaptive filters are necessary for some appliances because some constraints of the preferred processing action are not known in advance. Due to raise in the control of digital signal processors, adaptive filters include much more common and are now regularly used in devices for instance cellphones and extra communication devices, digital cameras, and medical monitoring equipment.

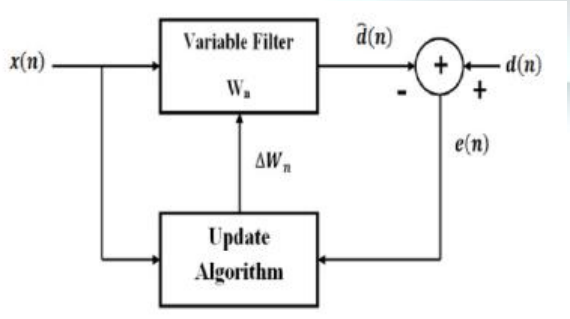


Fig 3.1. Adaptive FIR Filter Structure

The block diagram, shown above, is an instance of a basis for fastidious adaptive filter realizations, such as Least Mean Squares (LMS).The block diagram here specifies that a patchy filter can be made to extract an estimate of the desired signal.

3.2 LEAST MEAN SQUARE (LMS) ALGORITHM

The LMS algorithm is an adaptive algorithm which adjusts the coefficients of FIR filters iteratively. The following steps are pursued to detect the coefficients of an adaptive FIR filter:

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

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

Where,

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

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

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

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

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

$$\mathbf{E\ (n) = d\ (n) – y\ (n)} \quad (4.4)$$

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

3.3. DISTRIBUTED ARITHMETIC

Distributed Arithmetic (DA) is a diverse method for implementing digital filters. The basic idea of this technique is to utilize a DA table and a shift accumulator as a substitute for all multiplications and additions. DA is a bit-serial calculation process which permits 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 couple of vectors which is a generalcalculationtechnique 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 small number of steps by hoarding all probablegroupingsummation of weights in a memory table. The advantage of DA is its competence of automation.

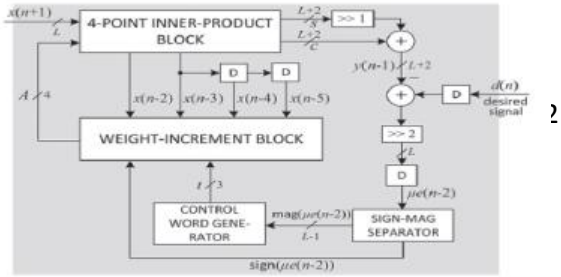
$$y = \Sigma\ (k)\ N{-}1K{=}1x\ (k) \quad (4.5)$$

Where,

w (k) is the fixed co-efficients

x (k) is the input data words

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



ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com
International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 3, Special Issue 1, March 2016

Fig 3.2. DA based LMS Adaptive Filter N=4

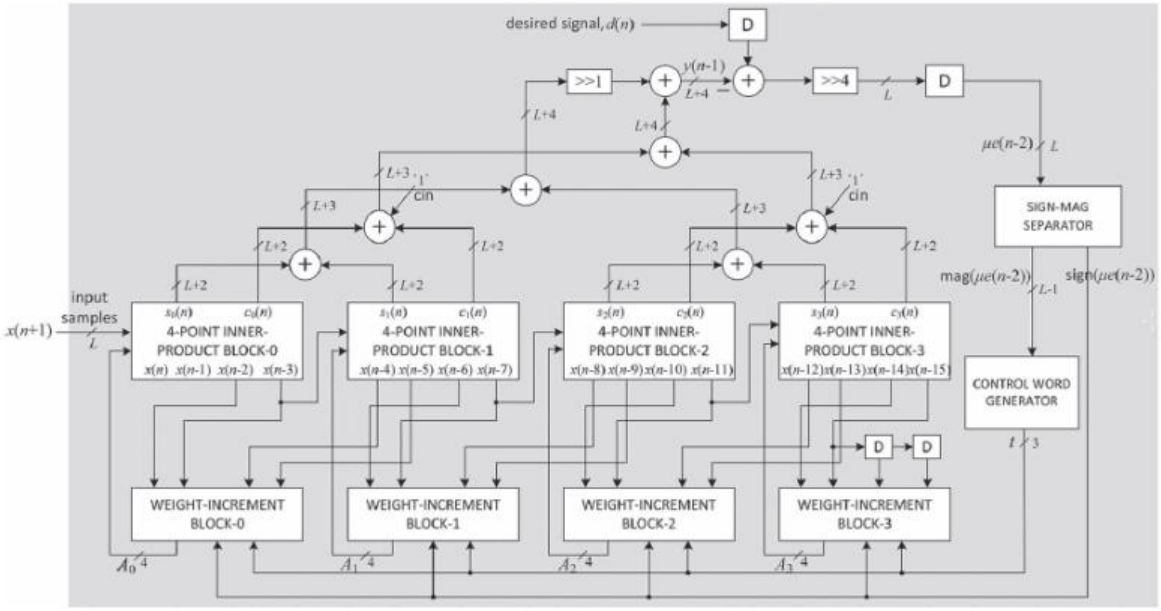


Fig 3.3. DA based LMS Adaptive Filter N=16

The key parts of the block diagram are as follows:

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

3.4. FOURPOINT INNER PRODUCT BLOCK

The early part of the filtering procedure of the LMS adaptive filter, for each cycle, is the need to perform an inner-product calculation. This is the chore that supplies to majority of the critical path. The block diagram of four-point inner product block is as shown in figure 3.4.

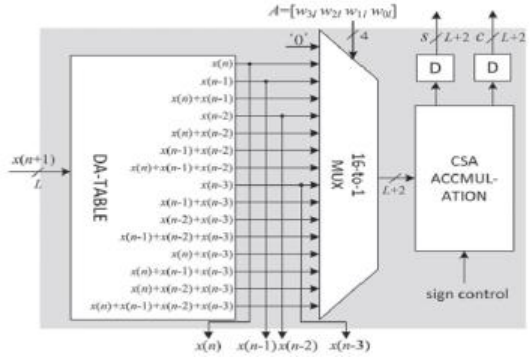


Fig 3.4. Structure of the four-point inner-product block

3.4.1 DA TABLE

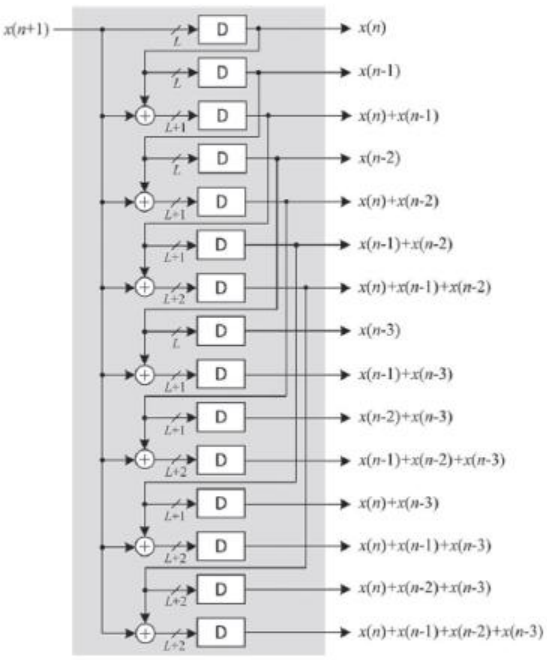


Fig 3.5. DA table for generation of possible sums of input samples

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com
International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 3, Special Issue 1, March 2016

It forms the initial part of the 4-point inner product block, such that it consists of an range of 15 registers. They are used for the principle of storage of partial inner products y. The structure of the DA table is as shown in figure 3.5.

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 the carry save accumulator.

3.4.3. CARRY SAVE ACCUMULATOR

The method of shift accumulation is made in the CSA block. The input of the block is from the 16-to-1 MUX. The bit portions are fed one after the other starting from LSB-to- MSB and to the CSA block. For MSB slices, negative of LUT output accretion is awfullyessential, so the LUT output is passed through XOR gates and sign control input. Hence, when MSB slice happens the XOR gate generate 1's complement as the sign control bit is set to 1. And then the sum and carry are attained behind L clock cycles. The arrangement of carry-save accumulator is as shown in figure 3.6.

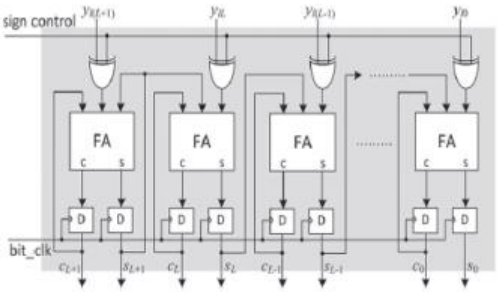


Fig 3.6. Carry-save implementation of shift accumulation

3.5. WEIGHT INCREMENT BLOCK

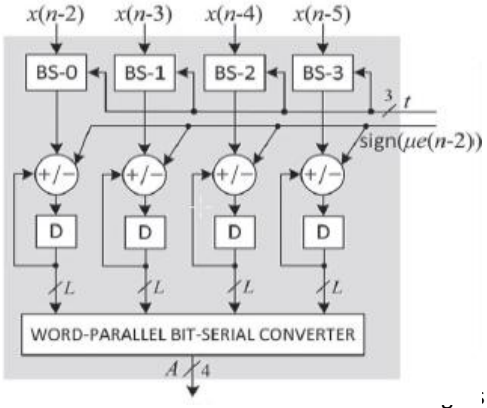


Fig 3.7. Weight Increment Block

It forms one of the vital parts of the filter design structure. The production of the weight for the updation of four-point inner product block is made by the weight increment block. The block diagram of weight increment block is as shown in figure 3.7.

3.5.1. BARREL SHIFTER

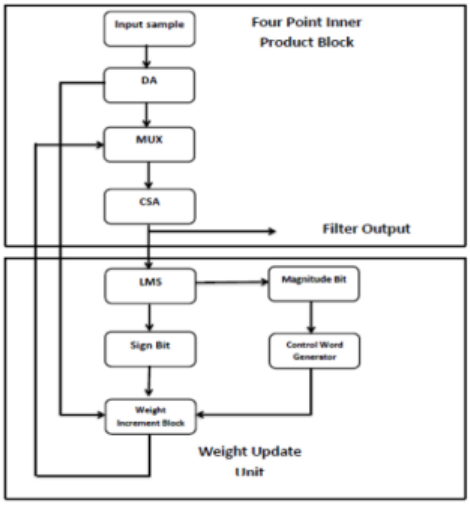
A barrel shifter is a digital circuit that can transfer a data word by a specified number of bits in one clock cycle. The control word for the working of the BS is formed by decoding the magnitude of error. This error that is decomposed is the difference between the desired output of the filter and the 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"
```

3.5.2. ADDER/SUBTRACTOR BLOCK

In adder/subtractor is a digital circuit that is able to add or subtract 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



ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com
International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 3, Special Issue 1, March 2016

Fig 3.8. Filtering process

However an additional delay of XOR gate is involved in carry-save adder. This additional delay affects the overall clock period. So this additional delay has to be reduced which is done by replacing full adder by half-adder in the carry-save adder.

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 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.

MODELSIM RESULTS

DISTRIBUTED ARITHMETIC

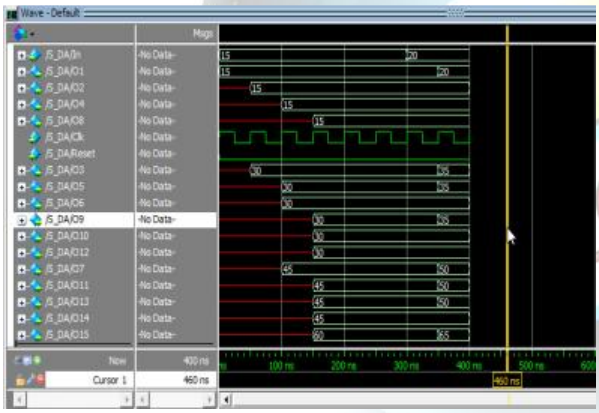


Fig 4.1. Simulation Result of DA

ADAPTIVE FIR FILTER

DESIGN	EXISTING	PROPOSED
Filter Length (N)	16	16
Minimum Sample Period (MSP) (ns)	9.927	6.237
Area Delay Product (sq.µm x ns)	15189	12532
Data Arrival Time (ns)	2.5	1.6
Power (mW)	5.75	4.03
Total delay(ns)	11.124	4.114

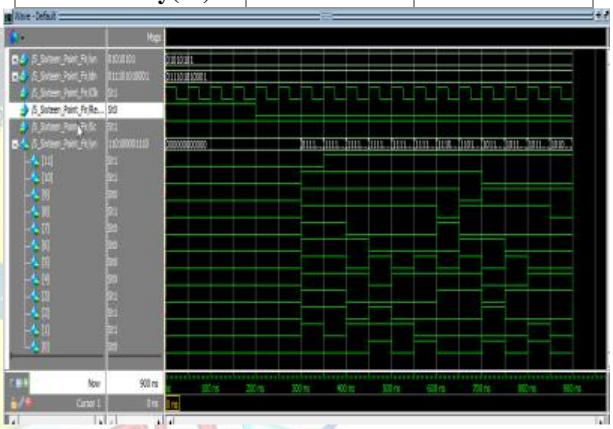


Fig 4.2. Simulation Result of FIR filter

TABLE I
Synthesis result comparison

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 calculation of filter output and additional delay is also reduced by the replacement of full adder by half adder. 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.

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com
International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 3, Special Issue 1, March 2016

REFERENCES

[1] 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.

[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.