International Journal of Advanced Information Science and Technology (IJAIST) ISSN: 2319:268
Vol.3, No.2, February 2014 DOI:10.15693/ijaist/2014.v3i2.99-104

Low-Power Pipelined Realization of Adaptive Filter Based on Distributed Arithmetic

Rajesh.R
PG Student/Department of ECE
CKCET, Cuddalore, India

Rameshprabu.R
PG Student/Department of ECE
CKCET, Cuddalore, India

Dhanabal.S
PG Student/Department of ECE
CKCET, Cuddalore, India

Ram Kumar.R
PG Student/Department of IT
SMVEC, Pondicherry, India

***Abstract*—The major issues while designing a distributed arithmetic (DA) based adaptive filter are high power and area delay. In general speed and power are the essential factor in VLSI design. So, we have designing a new pipelined architecture for efficient distributed arithmetic (DA) based on LMS adaptive FIR filter. The conventional adder-based shift accumulation for DA-based inner-product computation is replaced by conditional signed carry-save accumulation in order to reduce the sampling period and area complexity. 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. The throughput rate of the proposed design is significantly increased by parallel lookup table (LUT) update and concurrent implementation of filtering and weight-update operations. It involves the same number of multiplexers, smaller LUT, and nearly half the number of adders compared to the existing DA-based design. From synthesis results, it is found that the proposed design consumes less power and less area-delay product (ADP). It is done by using Xilinx tool and it is implemented using FPGA (Field Programmable Gate Array).**

Index terms -Adaptive filter, circuit optimization, distributed arithmetic (DA), least mean square (LMS) algorithm.

I. INTRODUCTION

Adaptive filters are widely used in several digital signal processing applications. The delay line finite impulse response (FIR) filter whose weights are updated by the famous Widrows–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 [1]. The direct form configuration on the forward path of the FIR filter results in a long critical path due to an inner-product computation to obtain a filter output. Therefore, when the input signal has a high sampling rate, it is necessary to reduce the critical path of the structure so that the critical path could not exceed the sampling period.

In recent years, the multiplier-less distributed arithmetic (DA)-based technique 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 using two separate lookup tables (LUTs) for filtering and weight update. To improve the design has using one LUT for filtering as well as weight updating. However, the structures in [2]–[4] 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 filter with very low adaptation delay [5].

This brief proposes a new DA-based architecture for low power, low area delay, and high-throughput pipelined implementation of adaptive filter with very low adaptation delay. The contribution of this brief are as follows.

1) Throughput rate is significantly increased by a parallel LUT update.
2) Further enhancement of throughput is achieved by concurrent implementation of filtering and weight updating.
3) Conventional adder-based shift accumulation is replaced by a conditional carry-save accumulation of signed partial inner products to reduce the sampling period, the bit cycle period amounts to memory access time plus 1-bit full-adder time by carry-save accumulation. The use of the proposed signed carry-save accumulation also helps to reduce the area complexity of the proposed design.
4) Reduction of power consumption is achieved by using a fast bit clock for carry-save accumulation but a much slower clock for all other operations.
5) The existing designs require an auxiliary control unit for address generation, which is not required in the proposed structure.

In the next section, we present a brief review of the LMS adaptive algorithm, followed by the description of the proposed DA-based technique for adaptive filter in Section III. The structure of the proposed adaptive filter is described in Section IV. We discuss the hardware complexity and synthesis

results of the proposed structure in Section V. Conclusions are given in Section VI.

II. Review of LMS Adaptive Algorithm

During each cycle, 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. The estimated error is then used to update the filter weights in every training cycle. The weights of LMS adaptive filter during the *n*th iteration are updated according to the following equations:

$$\mathrm{W(n+1)} = \mathrm{W(n)} + \mu \cdot \mathrm{e(n)} \cdot \mathrm{X(n)} \quad (1a)$$

Where

$$\mathrm{e(n)} = \mathrm{d(n)} - \mathcal{Y}\mathrm{(n)} \quad (1b)$$

$$\mathcal{Y}\mathrm{(n)} = \mathrm{W^{qT}(n)} \cdot \mathrm{X(n)} \quad (1c)$$

The input vector X(n)and the weight vector W(n) at the n^{th} training iteration are respectively given by

$$X(n) = [x(n), x(n-1), .., x(n-N+1)\,]^T \quad (2a)$$

$$W(n) = [w_0(n) w_1(n), .., w_{N-1}(n)]^T \quad (2b)$$

$d(n)$ is the desired response, and $\mathcal{Y}(\mathrm{n})$ is the filter output of the n^{th} iteration. $e(n)$ denotes the error computed during the n^{th} iteration, which is used to update the weights, $\mu$is the convergence factor, and *N* is the filter length.

In the case of pipelined 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

$$\mathrm{W(n+1)} = \mathrm{W(n)} + \mu \cdot \mathrm{e(n-m)} \cdot \mathrm{X(n-m)} \quad (3)$$

III. Proposed DA Based Approach for Inner product Computation

The LMS adaptive filter, in each cycle, needs to perform an inner-product computation which contributes to the most of the critical path. For simplicity of presentation, let the inner product of (1c) be given by

$$\mathcal{Y} = \sum_{K=0}^{N-1} w_k \cdot x_k \quad (4)$$

where$w_k$and$x_k$for $0 \le k \le N-1$ fromthe N point vectors corresponding the current weights and most recent $N-1$ input, respectively. Assuming L to be the bit width of the weight, each component of the weight vector may be expressed in two's complement representation

$$w_k = -w_{k0} + \sum_{l=0}^{L-1} w_{kl} \cdot 2^{-l} \quad (5)$$

Where w_{kl}denotes the l^{th} bit ofw_k. Substituting (5), we can write (4) in an expanded form

$$\mathcal{Y} = -\sum_{k=0}^{N-1} x_k \cdot w_{k0} + \sum_{k=0}^{N-1} x_k \cdot \left[\sum_{l=0}^{L-1} w_{kl} \cdot 2^{-l}\right] \quad (6)$$

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

$$\mathcal{Y} = -\sum_{k=0}^{N-1} x_k \cdot w_{k0} + \sum_{l=1}^{L-1} 2^{-l} \cdot \left[\sum_{k=0}^{N-1} x_k \cdot w_{kl}\right] \quad (7)$$

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

$$\mathcal{Y} = \left[\sum_{l=1}^{L-1} 2^{-l} \cdot \mathcal{Y}_l\right] - \mathcal{Y}_0,$$

$$\text{where} \mathcal{Y}_l = \sum_{k=0}^{N-1} x_k \cdot w_{kl} \quad (8)$$

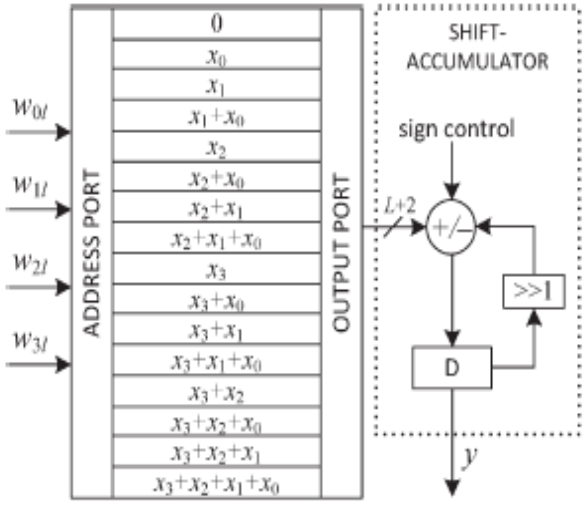

Fig 1 Conventional DA-based implementation of four-point inner product

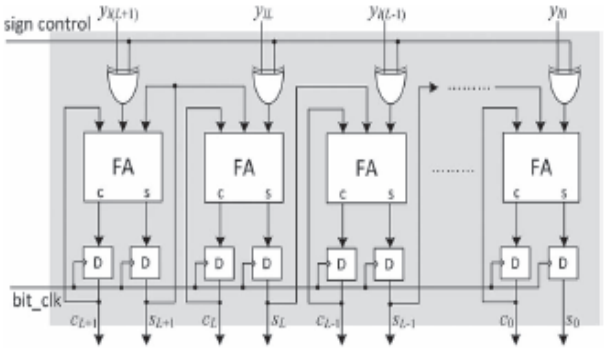

Fig 2 Carry-save implementation of shift accumulation

Since any element of the *N*-point bit sequence $\{w_{kl}$for $0 \le k \le N-1\}$can either be zero or one,the partial sum $\mathcal{Y}_l$ for $l = 0,1,\ldots,L-1$ can have 2^Npossible values. If all the 2^Npossible values of $\mathcal{Y}_l$are pre computed and stored in a LUT, the partial sums$\mathcal{Y}_l$can be read out from the LUT using the bit sequence $\{w_{kl}\}$as address bits for computing the inner product.

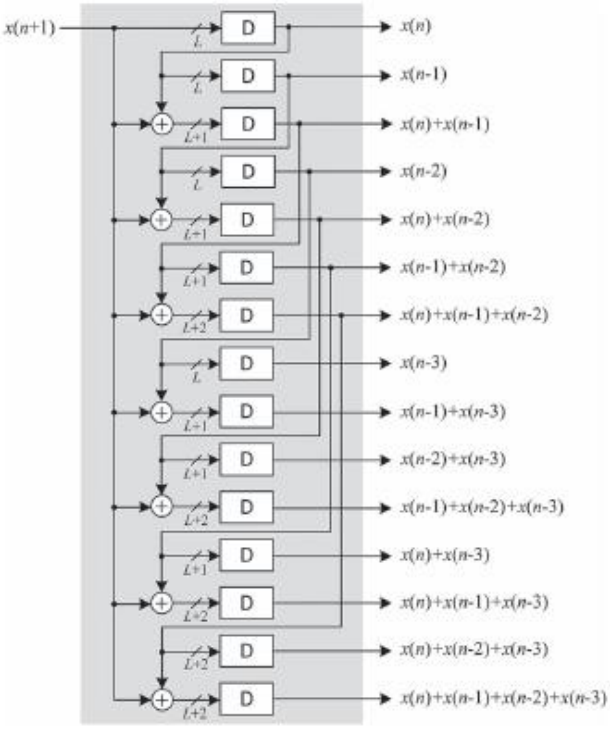

Fig 3 DA table for generation of possible sums of input samples

The inner product of (8) can therefore be calculated in L cycles of shift accumulation, followed by LUT-read operations corresponding to L number of bit slices $\{w_{kl}\}$ for $0 \le l \le L-1$, as shown in Fig. 1. Since the shift accumulation in Fig. 1 involves significant critical path, we perform the shift accumulation using carry-save accumulator, as shown in Fig 2. The bit slices of vector Ware fed one after the next in the least significant bit (LSB) to the most significant bit (MSB) order to the carry-save accumulator. However, the negative (two's complement) of the LUT output needs to be accumulated in case of MSB slices. Therefore, all the bits of LUT output are passed through XOR gates with a sign-control input which is set to one only when the MSB slice appears as address. The XOR gates thus produce the one's complement of the LUT output corresponding to the MSB slice but do not affect the output for other bit slices. Finally, the sum and carry words obtained after L clock cycles are required to be added by a final adder (not shown in the figure), and the input carry of the final adder is required to be set to one to account for the two's complement operation of the LUT output corresponding to the MSB slice.

The content of the k^{th} LUT location can be expressed as

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

Where$k_j$is the $(j+1)^{th}$ bit of *N*-bit binary representation of integer k for $0 \le k \le 2^N - 1$. Note that $c_k$for $0 \le k \le 2^N - 1$ can be pre computed and stored in RAM-based LUT of 2^Nwords. However, instead of storing 2^Nwords in LUT, we store $(2^N - 1)$words in a DA table of $2^N - 1$registers.

IV. PROPOSEDDA-BASED ADAPTIVE FILTER STRUCTURE

The proposed structure of DA-based adaptive filter is shown in Fig. 4. It consists of a four-point inner product block and a weight-increment block along with additional circuits for the computation of error value $e(n)$ and control word t for the barrel shifters.

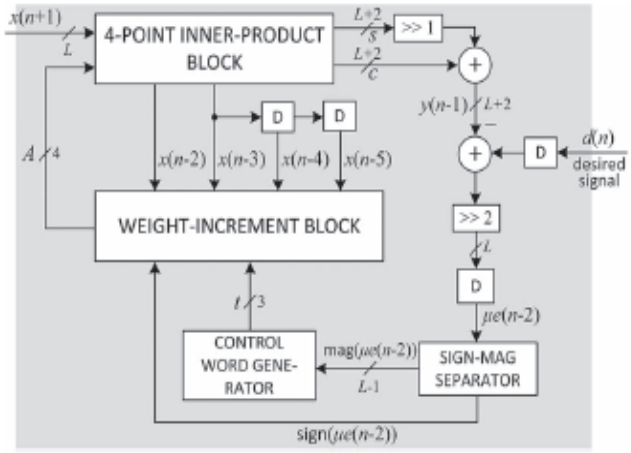

Fig 4 Proposed structure of DA-based LMS adaptive filter of filter length N = 4.

The four-point inner-product block shown in Fig. 5 includes a DA table consisting of an array of 15 registers which stores the partial inner products $Y_l$for $0 < l \le 15$ and a 16 : 1 multiplexor (MUX) to select the content of one of those registers. Bit slices of weights $A = \{w_{3l} w_{2l} w_{1l} w_{0l}\}$for $0 \le l \le L-1$ are fed to the MUX as control in LSB-to- MSB order, and the output of the MUX is fed to the carry-save accumulator (shown in Fig. 2). After L bit cycles, the carry-save accumulator shift accumulates all the partial inner products and generates a sum word and a carry word of size (L + 2) bit each. The carry and sum words are shifted added with an input carry "1" to generate filter output which is subsequently subtracted from the desired output d(n) to obtain the error e(n). All the bits of the error except the most significant one are ignored, such that multiplication of input $x_k$by the error is implemented by a right shift through the number of locations given by the number of leading zeros in the magnitude of the error.

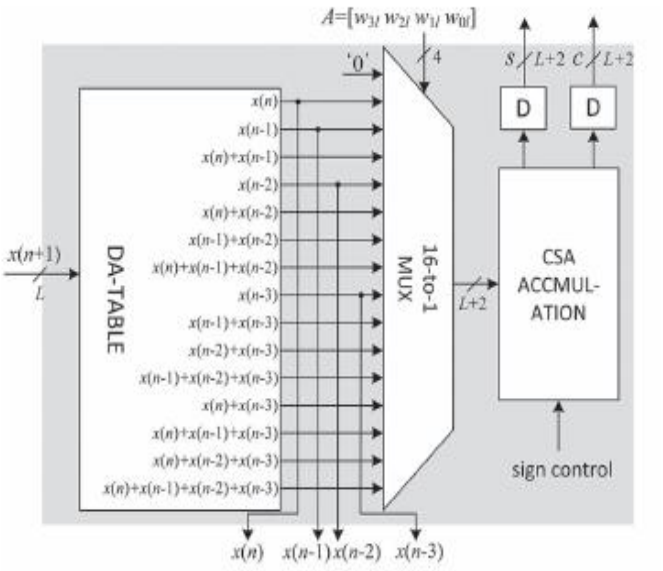

Fig 5 Structure of the four-point inner-product block

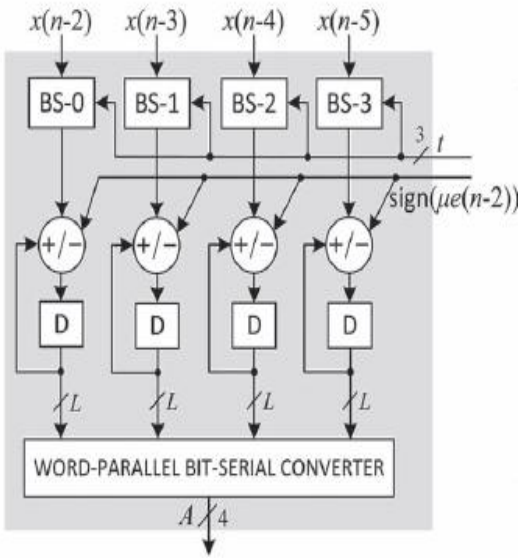

Fig 6 Structure of the weight-increment block

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 to be used for the barrel shifter. The convergence factor $\mu$is usually taken to be$O(1/N)$. We have taken$\mu = 1/N$. However, one can take μas $2^{-i}/N$, where $i$is a small integer. The number of shifts t in that case is increased by$i$, and the input to the barrel shifters is pre shifted by $i$locations accordingly to reduce the hardware complexity.

The weight-increment unit shown in Fig. 6 consists of four barrel shifters and four adder/sub-tractor cells. The barrel shifter shifts the different input values $x_k$for $k = 0,1, \dots, N-1$ by appropriate number of locations. The barrel shifter yields the desired increments to be added with or subtracted from the current weights. The sign bit of the error is used as the control for adder/sub-tractor cells such that, when sign bit is zero or one, the barrel-shifter output is respectively added with or subtracted from the content of the corresponding current value in the weight register.

V. COMPLEXITY

We have estimated the hardware and time complexities of the proposed design in Table I. The proposed design uses two clocks, namely, the bit clock and the byte clock. The duration of the byte clock is the same as the sampling period. The bit clock is used in carry-save accumulation units and word parallel bit-serial converters, while the byte clock is used in the rest of the circuit. The duration of bit clock is given by $T_{BC} = 4T_M + T_{FA} + T_{XOR} + T_D$, where T_M, T_{FA}, T_{XOR} and $T_D$are the delays of a $2:1$ MUX, a full adder, an XOR gate, and a D flip-flop, respectively, and L is the bit width of inputs as well as coefficients. $4T_M$is the delay of a $16:1$ MUX, since it consists of four stages of B $2:1$ MUXes. The duration of the sample period (byte clock) of proposed design is$L \times T_{BC}$.

The proposed design involves 9 adders/sub -tractor, 4 logarithmic barrel shifters (of three stages each), and 31 registers. The design in [5] on the other hand involves 25 adders, 15 shifters, and 33 registers. The proposed design has an adaptation delay of two clocks, one of which is after the addition of carry and sum words in the inner-product computation blocks and the other is after the error calculation, whereas the designs in [4] and [5] have no such adaptation delay. The adaptation delay of two cycles, however, does not make noticeable degradation of the convergence performance.

The RTL (Register Transfer Level) schematic view DA based adaptive filter architecture is shown in Figure 7.

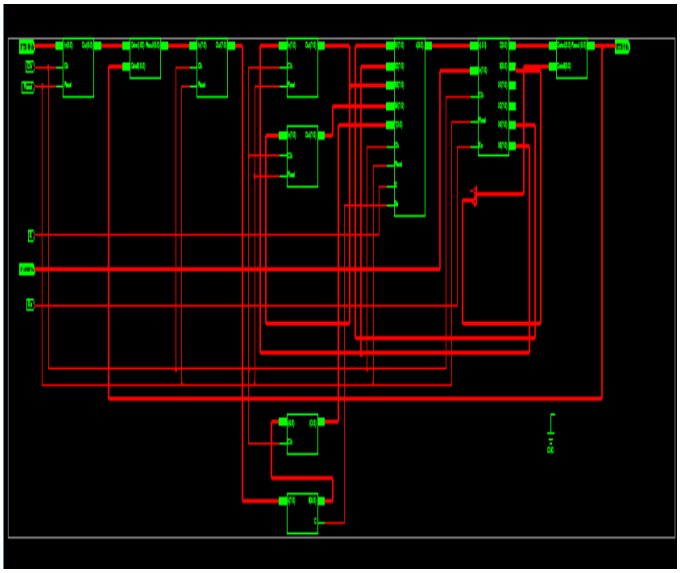

Fig 7 RTL schematic

.

Design	Throughput	Hardware elements				
		Adders	Register	Shifter	LUT	Extra
Allred et al [2]	$T_R + 3T_M + T_A$	4	7	1	2	Control unit
Guo et al [3]	$L(4T_M + T_{FA})$	13	10	4	2	Control unit
Meher et al [5]	$L(4T_M + T_{FA})$	25	33	15	-	16:1MUX
Proposed	$L(4T_M + T_{FA} + T_{XOR} + T_D)$	9	31	3	-	16:1MUX

The comparative output of conventional design and proposed architecture complexity is shown in Table 1. From synthesis results, it is found that the proposed design consumes less power and less area-delay product (ADP) for DA-based adaptive filter shown in Table 2.

Logic Utilization	Used	Available	Utilization
Number of slice Flip Flops	291	1920	15%
Number of 4 inputs LUTs	394	1920	20%
Number of occupied slices	259	960	26%
Number of slices related logic	259	960	26%
Number of slices unrelated logic	0	259	0%
Total number of 4 input LUTs	400	1920	20%
Number of bounded IOBs	32	66	48%
Number of GCLKs	1	24	4%
DELAY	6.404ns		
Power	33.59 mW		

The simulated output of the DA based adaptive filter architecture is shown in Figure 8. The simulation is done by Xilinx Simulator and the result is verified. Finally the delay will be minimized to 6.404 ns and power consumption can be brought to 33.59 mW. This will be obtained by the implementation of adaptive filter technique using Distributed arithmetic.

VI. CONCLUSION

An efficient pipelined architecture for low-power and low-area delay implementation of DA-based adaptive filter is proposed. A carry-save accumulation scheme of signed partial inner products for the computation of filter output was constructed. Also throughput rate is significantly enhanced by parallel LUT update and concurrent processing of filtering operation and weight-update operation.

From the synthesis results, it has been found that the proposed design consumes less power and less ADP. Offset binary coding is popularly used to reduce the LUT size, for obtaining minimized area delay, to half for area-efficient implementation of DA [5], which can be applied to our design as well.

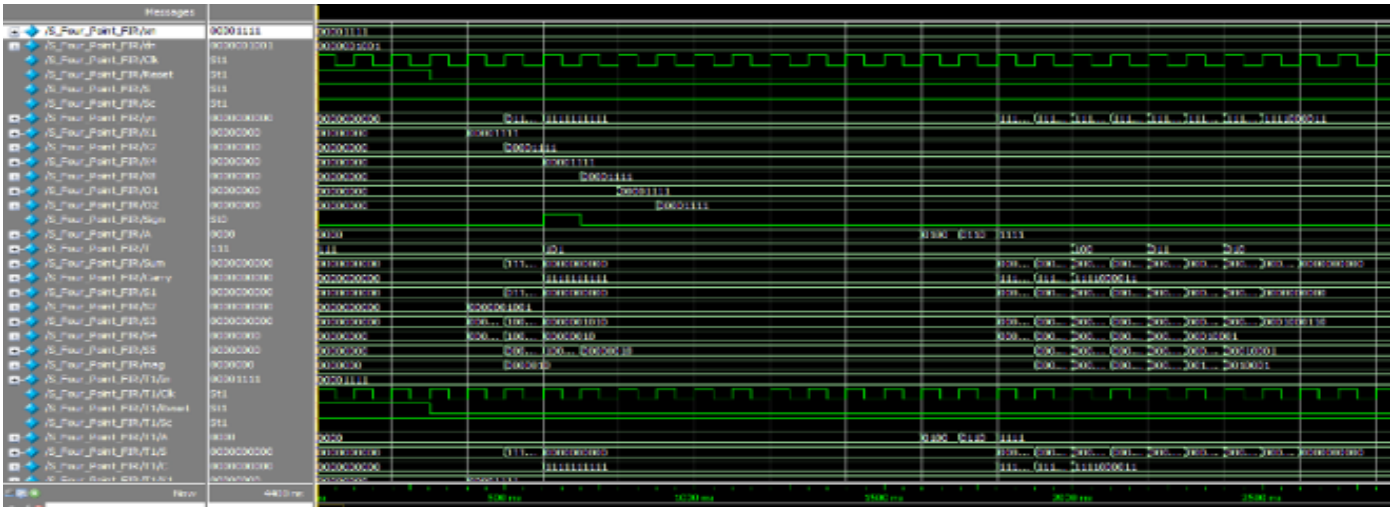

Fig 8 Output of DA based adaptive filter

REFERENCES

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

[2]. [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]. [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]. [4] 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.

[5]. [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.

Authors Profile

R.Rajesh received the **B.E.** degree in electronics and communication engineering from the St.Joseph's College of Engineering and Technology, Thanjavur, Anna University, Chennai, India, in 2012.Currently doing **M.E.** in electronics and communication engineering (Applied Electronics) from the C K College of Engineering and Technology, Cuddalore, Anna University, Chennai, India. His research interest includes VLSI, Digital signal processing, Embedded System, wireless communication.

R.Rameshprabu received the **B.E.** degree in electronics and communication engineering from the Adhiparasakthi Engineering College, Melmaruvathur, Anna University, Chennai, India, in 2010.Currently doing **M.E.** in electronics and communication engineering (Applied electronics) from the C K College of Engineering and Technology, Cuddalore, Anna University , Chennai, India. His research interest includes wireless Network, wireless communication (**WiFi,WiMax**), VLSI, Mobile Communication.

S.Dhanabal received the **B.E.** degree in electronics and communication engineering from the Arunai Engineering College, Thiruvannamalai, Anna University, Chennai, India, in 2010.Currently doing **M.E.** in electronics and communication engineering (Applied electronics) from the C K College of Engineering and Technology, Cuddalore, Anna University, Chennai, India. His research interest includes wireless communication (**WiFi,WiMax**), Mobile Communication, VLSI.

R.Ram Kumar received the **B.Tech.** degree in information technology from the St.Joseph's College of Engineering and Technology, Thanjavur, Anna University, Chennai, India, in 2012. Currently doing **M.Tech.** in information technology (Networking) from the Sri Manakula vinayagar engineering college, Pondicherry University, Pondicherry, India. His research interest includes wireless communication, VLSI, wireless network, networking, Cloud computing.