IJIEMR Transactions, online available on 29th Jan 2019. Link :
http://www.ijiemr.org/main/index.php?vol=Volume-08&issue=ISSUE-01

Title: **HIGH SPEED RECURSIVE NOISE CANCELLATION WITH FAST CONVERGENCE DIGITAL SYSTEM**

Volume 08, Issue 01, Pages: 329–331.

Paper Authors

DR.P.SUNEEL KUMAR, P.SWAPNA
sridevi women's engineering college, Hyderabad

International Journal for Innovative Engineering and Management Research
A Peer Revieved Open Access International Journal
www.ijiemr.org

HIGH SPEED RECURSIVE NOISE CANCELLATION WITH FAST CONVERGENCE DIGITAL SYSTEM

[1]DR.P.SUNEEL KUMAR, [2]P.SWAPNA
[1]Professor Department of ECE, Sridevi Women's Engineering College
[2]Assistant Professor Department of ECE Sridevi Women's Engineering College
[1]psunilkumar.phd@gmail.com,[2]swapna.patibandlaa@gmail.com

Abstract- The key objective of this paper is to provide an idea for VLSI Implementation of RLS algorithm for Noise Cancellation with real time analog inputs. In this paper, we present an efficient architecture for the implementation of ANC systems often for high-speed digital signal processors to cancel out disturbing noise. The throughput rate of the proposed design is significantly increased by recursive update and concurrent implementation of filtering and weight-update operations. The conventional LMS inner-product computation is replaced by conditional signed recursive accumulation in order to reduce the sampling period and area complexity. The proposed implementation significantly outperforms the existing implementations in terms of three important key metrics. 1. The least mean squares (LMS) algorithms adjust the filter coefficients to minimize the cost function. Compared to least mean squares (LMS) algorithms, the RLS algorithms achieve faster convergence by variable step size. 2. Proposed RLS algorithms require fewer computational resources and memory than the RLS algorithms. 3. The implementation of the algorithms is less complicated due to lesser tap approach than the all other existing algorithms. Through MATLAB simulation experiments efficiency of RLS over LMS will be proved. The VLSI implementation results show that the proposed algorithm as superior performance in Fast convergence rate, low complexity, and has superior performance in noise cancellation.

Keywords- Active noise cancellation(ANC), least mean square (LMS) ,recursive lease square(RLS)

I. INTRODUCTION

The Least Mean Square (LMS) algorithm is introduced by Hoff in 1960.In diverse fields of engineering Least Mean Square algorithm is used because of its simplicity. It has been used in many fields such as adaptive noise cancellation, adaptive equalization, side lobe reduction in matched filters, system identification etc. By using simple architecture for the implementation of variant Block LMS algorithm in which weight updation and error calculation are both calculated in block wise, Hardware outputs are verified with simulations from FPGA. For the computation efficiency of the LMS algorithm some additional simplification are necessary in some application. There are many approaches to decrease the computational requirements of LMS algorithm that is block LMS algorithm [1]. In Block LMS algorithm technique involves calculation of a block of finite set of output values from block of input values. Efficient parallel processors can be used in block implementations of digital filters which results in speed gains [1]. LMS is one of the adaptive filtering algorithms derived from steepest descent algorithm used in wide variety of applications. Block LMS is one of the variants in which the weights are updated once per every block of data instead of updating on every clock cycle of input data.

II. ALGORITHM FORMULATION

In [1], the adaptation step size is adjusted using the energy of the instantaneous error. The weight update recursion is given by

$$\mathbf{W}(n+1) = \mathbf{W}(n) + \mu(n)e(n)\mathbf{X}(n) \tag{1}$$

and the step-size update expression is

$$\mu(n+1) = \alpha\mu(n) + \gamma e^2(n) \tag{2}$$

The constant umax is normally selected near the point of instability of the conventional LMS to provide the maximum possible convergence speed. The value of is chosen as a compromise between the desired level of steady state misadjustment and the required tracking capabilities of the algorithm. The parameter controls the convergence time as well as the level of misadjustment of the algorithm. The algorithm has preferable performance over the fixed step-size LMS: At early stages of adaptation, the error is large, causing the step size to increase, thus providing faster convergence speed. When the error decreases, the step size decreases, thus yielding smaller misadjustment near the optimum. However, using the instantaneous error energy as a measure to sense the state of the adaptation process does not perform as well as expected in the presence of measurement noise. This can be seen from (3). The output error of the identification system is

$$e(n) = d(n) - \mathbf{X}^{\mathbf{T}}(n)\mathbf{W}(n) \tag{3}$$

where the desired signal d(n) is given by,

$$d(n) = \mathbf{X}^{\mathbf{T}}(n)\mathbf{W}^{*}(n) + \xi(n) \tag{4}$$

A. NORMALISED LEAST MEAN SQUARE (NLMS) ALGORITHM

To derive the NLMS algorithm we consider the standard LMS recursion, for which we select a variable step size parameter, □ (n). This parameter is selected so that the error value, $e^+(n)$, will be minimized using the updated filter tap weights, w(n+1), and the current input vector, **x**(n).

$$\mathbf{w}(n+1) = \mathbf{w}(n) + 2\mu(n)e(n)\mathbf{x}(n)$$

$$e^{+}(n) = d(n) - \mathbf{w}^{T}(n+1)\mathbf{x}(n)$$

Next we minimize $(e^+(n))^2$, with respect to □ (n). Using this we can then find a value for □ (n) which forces e^+ (n) to zero.

$$\mu(n) = \frac{1}{2\mathbf{x}^{T}(n)\mathbf{x}(n)}$$

This □ (n) is then substituted into the standard LMS recursion replacing □, resulting in the following.

$$\mathbf{w}(n+1) = \mathbf{w}(n) + 2\mu(n)e(n)\mathbf{x}(n)$$

$$\mathbf{w}(n+1) = \mathbf{w}(n) + 1/\left(\mathbf{x}^{T}(n)\mathbf{x}(n)\right)\mathbf{e(n)x(n)}$$

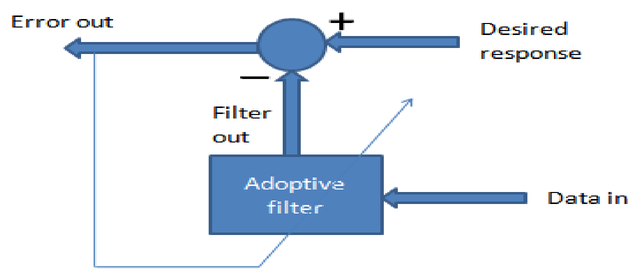


Fig 1.Adoptive filter structure

Often the NLMS algorithm is expressed as equation 5.20; this is a slight modification of the standard NLMS algorithm detailed above. Here the value of is a small positive constant in order to avoid division by zero when the values of the input vector are zero.. The parameter is a constant step size value used to alter the convergence rate of the NLMS algorithm, it is within the range of 0<<2, usually being equal to 1.

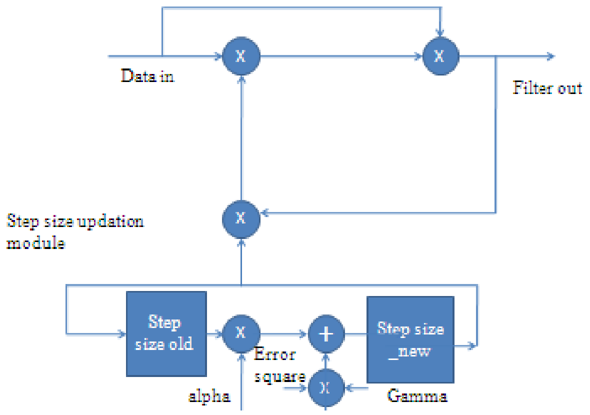


Fig 2 .Coefficient path estimation

$\mathbf{w}(n+1) = \mathbf{w}(n) + e(n)*\mathbf{x}(n)$.

A. FPGA Realization Issues

To Field programmable gate arrays are ideally suited for the implementation of adaptive filters. However, there are several issues that need to be addressed. When performing software simulations of Adaptive filters, calculations are normally carried out with floating point precision. Unfortunately, The resources required of an FPGA to perform floating point arithmetic is normally too large to be justified, and measures must be taken to account for this. Another concern is the filter tap itself. Numerous techniques have been devised to efficiently calculate the convolution operation when the filters coefficients are fixed in advance. For an adaptive filter whose coefficients change over time, these methods will not work or need to be modified significantly.

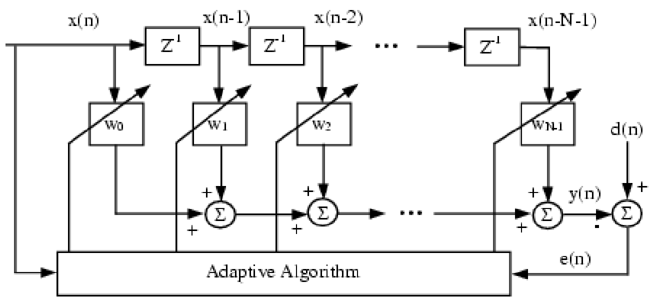


Fig.3 Design of Transversal Filters

III. IMPLEMENTATION

LMS algorithm mainly consists of two basic process

- ➢ Filtering process
- ➢ Adaptive process

Filtering process:

• In filtering process FIR filter output is calculated by convolving inputs and tap weights.

• Estimation error is calculated by comparing the output with desired signal.

Adaptive process

• In adaptive process tap weights are updated based

on the estimation error.

Three Steps Involved

• Calculation of filter output.

• Estimation of error.

• Tap weight up-dation.

A. LMS adaptive filter : Basic Concepts:

In this algorithm filter weights are updated with each new sample as required to meet the desired output. The computation required for weights update is illustrated by equation (1). If the input values u(n),u(n - 1),u(n - 2)....u(n - N + 1) form the tap input vector u(n), where N denotes the filter length, and the weights w^0(n)……w^N-1(n) form the tap weight vector

w(n) at iteration n, then the LMS algorithm is given by the following equations:
y(n)= w^h(n)*u(n)
e(n)= d(n)-y(n)
w^(n+1)=w^(n)+M*u(n)*e(n) (1)

where y(n) denotes the filter output. d(n) denotes the desired output. e(n) denotes the filter error (the difference between the desired filter output and current filter output) which is used to update the tap weights. M denotes a learning rate, and W^(n+1) denotes the new weight vector that will be used by the next iteration.

B. Variable Step Size

$$w_i(n+1) = w_i + 2\mu_i g_i(n)$$

$$\mathbf{g}(n) = e(n)\mathbf{x}(n)$$

Where g is a vector comprised of the gradient terms, g_i(n)=e(n)x(n-i), i=0…N-1, the length corresponds to the order of the adaptive filter. The values for □ (n) can be calculated in either of the methods expressed in equation 3.26. The choice is dependent on the application, if a digital signal processor is used then the second equation is preferred. However, if a custom chip is designed then the first equation is usually utilized. For both the Matlab and real time applications the first equation is being implemented. Here □ is a small positive constant optionally used to control the effect of the gradient terms on the update procedure, in the later implementations this is set to 1

$$\mu_i(n) = \mu_i(n-1) + \rho\, sign(g_i(n))\, sign(g_i(n-1))$$

$$\mu_i(n) = \mu_i(n-1) + \rho g_i(n)\, g_i(n-1)$$

In order to ensure the step size parameters do not become too large (resulting in instability), or too small (resulting in slow reaction to changes in the desired impulse response), the allowable values for each element in the step size are bounded by upper and lower values.

C. Resource usage in implementation

Here the architecture is designed to perform in real time implementation. In input buffering RAMS the continuous incoming data is stored that provide the calculations involved until for the weight updating. From each of the input RAM blocks the data is read out alternatively and passed in to input data memory. In tap weight memory block tap weights are initially stored. Both weights and input data's are passed to the multiplier simultaneously for multiplication which multiplies both weight vector and data vector. This result is passed to adder which adds the output of the multiplier. Then the adder output is equivalent to the FIR filter output,(which ideally requires N multipliers depending on the number of taps which here is reduced to one) is then subtracted from another input sample to calculate the error.

To calculate the weight updates the obtained error value is multiplied with data vector and step size to reduce the error from each calculation. Then updated weights are added to previous weights and written back to weight memory. Then updated weights are ready for another data set. As mentioned earlier ideally FIR filter structure requires N multiplier depending on the number of weights considered for implementation. Here multiplier used is reduced to one so architecture presented consumes minimum hardware which is convenient for FPGA implementation.

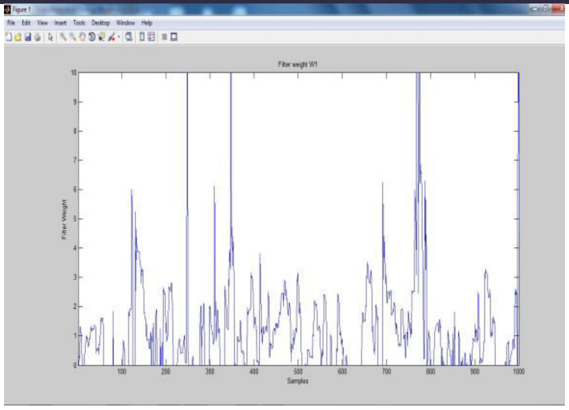


Fig 4. Filter weight updates using VSS-LMS

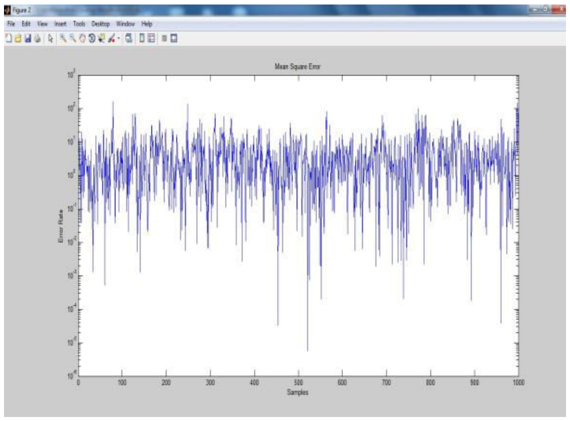
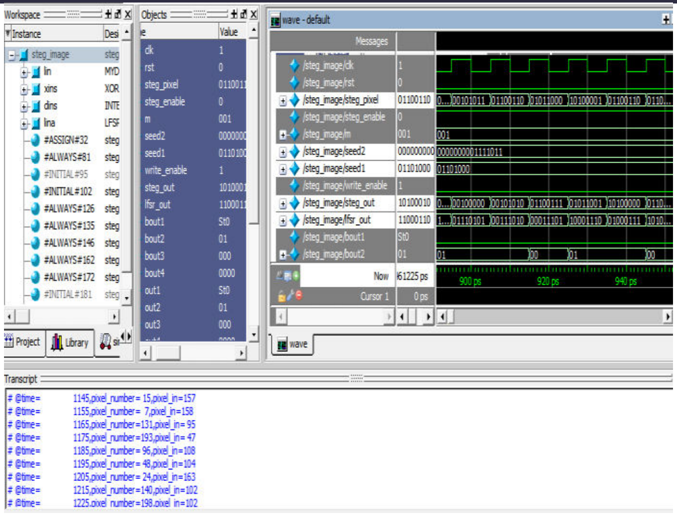


Fig 5. MSE graph in VSS-LMS

IV. DISTORTION ANALYSIS

Normalized least mean square algorithm generates less mean square error rate when compared to the variable step size least mean square algorithm and least mean square algorithm. The NLMS algorithm, an equally simple, but more robust variant of the LMS algorithm, exhibits a better balance between simplicity and performance than the LMS algorithm. Finally we carried out hardware implementation of NLMS and the design was initially verified with Modelsim simulator tool and we successfully synthesize the verilog HDL code with QUARTUS II EDA tool. Due to its good characteristics the NLMS has been largely used in real-time applications.



```
# @time=        1145,pixel_number= 15,pixel_in=157
# @time=        1155,pixel_number=  7,pixel_in=158
# @time=        1165,pixel_number=131,pixel_in= 95
# @time=        1175,pixel_number=193,pixel_in= 47
# @time=        1185,pixel_number= 96,pixel_in=108
# @time=        1195,pixel_number= 48,pixel_in=104
# @time=        1205,pixel_number= 24,pixel_in=163
# @time=        1215,pixel_number=140,pixel_in=102
# @time=        1225,pixel_number=198,pixel_in=102
```

Fig 6. Simulated output

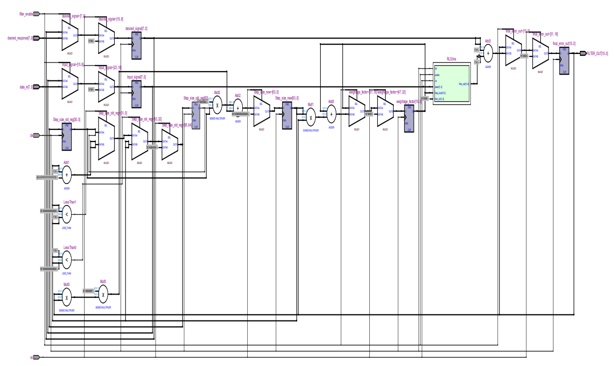
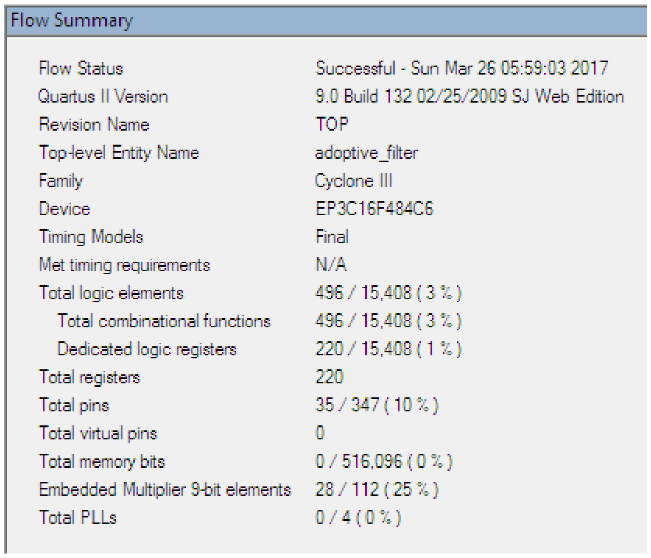


Fig 7. RTL schematic output

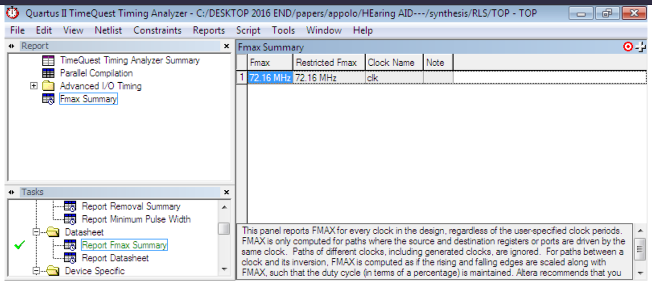


Flow Summary

Flow Status	Successful - Sun Mar 26 05:59:03 2017
Quartus II Version	9.0 Build 132 02/25/2009 SJ Web Edition
Revision Name	TOP
Top-level Entity Name	adoptive_filter
Family	Cyclone III
Device	EP3C16F484C6
Timing Models	Final
Met timing requirements	N/A
Total logic elements	496 / 15,408 (3 %)
Total combinational functions	496 / 15,408 (3 %)
Dedicated logic registers	220 / 15,408 (1 %)
Total registers	220
Total pins	35 / 347 (10 %)
Total virtual pins	0
Total memory bits	0 / 516,096 (0 %)
Embedded Multiplier 9-bit elements	28 / 112 (25 %)
Total PLLs	0 / 4 (0 %)

Fig 8. Area summary

International Journal for Innovative Engineering and Management Research
A Peer Revieved Open Access International Journal
www.ijiemr.org


Quartus II TimeQuest Timing Analyzer - C:/DESKTOP 2016 END/papers/appolo/HEaring AID---/synthesis/RLS/TOP - TOP
File Edit View Netlist Constraints Reports Script Tools Window Help
Report
TimeQuest Timing Analyzer Summary
Parallel Compilation
Advanced I/O Timing
Fmax Summary
Tasks
Report Removal Summary
Report Minimum Pulse Width
Datasheet
Report Fmax Summary
Report Datasheet
Device Specific
Fmax Summary

	Fmax	Restricted Fmax	Clock Name	Note
1	72.16 MHz	72.16 MHz	clk	

This panel reports FMAX for every clock in the design, regardless of the user-specified clock periods. FMAX is only computed for paths where the source and destination registers or ports are driven by the same clock. Paths of different clocks, including generated clocks, are ignored. For paths between a clock and its inversion, FMAX is computed as if the rising and falling edges are scaled along with FMAX, such that the duty cycle (in terms of a percentage) is maintained. Altera recommends that you

Fig 8 . Fmax report.

Table II Hardware complexity report comparison using cyclone III family devices

TYPE	Multiplier used	POWER REPORT(mW)	SPEED (MHz)
LMS	33	68.11	40.66
RLS-PROPOSED	28	67.27	72.16

V. CONCLUSION

Here FPGA-based implementation of both least mean square and recursive approach has been tested for realizing the loss compensation filter for a number of audiograms and coupled with noise attenuation filters. Magnitude responses of these filters, measured using swept sinusoidal tone as input, showed close match with the corresponding desired magnitude responses. And finally an area-and power-efficiency of RLC ANC circuit over LMS has been proved for low power in-ear headphones. The proposed design has been synthesized successfully using QUARTUS II EDA tool.

REFERENCES

[1] G A Clark, S K Mitra and S Parker ''Block implementation of Adaptive Digital Filters'', IEEE transaction on Acoustics, speech and signal processing, 1981.

[2] T.Yoshida, Y. Liguini, H.maeda ''Systolic array implementation of block LMS Algorithm'', Proc of IEEE Electronic letters 1998.

[3] Tian Lan, Jinlin Zhang, ''FPGA Implementation of Adaptive Noise Canceller'' Proc of 2008 International symposiums on information processing .

[4] Maurice G. Bellanger ''An FPGA Implementation of LMS adaptive filters for audio processing'', New York,Dekker,1987.

[5] W. S. Gan, S. Mitra, and S. M. Kuo, "Adaptive feedback active noise control headset: Implementation, evaluation and its extensions," IEEE Trans. Consum. Electron., vol. 51, no. 3, pp. 975–982, Aug. 2005.

[6] Y. Song, Y. Gong, and S. M. Kuo, "A robust hybrid feedback active noise cancellation headset," IEEE Trans. Speech Audio Process., vol. 13, no. 4, pp. 607–617, Jul. 2005.

[7] S. M. Kuo, S. Mitra, and W.-S. Gan, "Active noise control system for headphone applications," IEEE Trans. Control Syst. Technol., vol. 14, no. 2, pp. 331–335, Mar. 2006.

[8] M. Guldenschuh and R. Höldrich, "Prediction filter design for active noise cancellation headphones," IET Signal Process., vol. 7, no. 6, pp. 497–504, Aug. 2013.

[9] L. Zhang, L. Wu, and X. Qiu, "An intuitive approach for feedback active noise controller design," Appl. Acoust., vol. 74, no. 1, pp. 160–168, Jan. 2013. [10] L. Zhang and X. Qiu, "Causality study on a feedforward active noise control headset with different noise coming directions in free field," Appl. Acoust., vol. 80, pp. 36–44, Jun. 2014.