Journal of
Applied & Computational Mathematics

Joseph and Purushotham, J Appl Computat Math 2015, 4:4
DOI: 10.4172/2168-9679.1000237

Short Communication Open Access

Implementation of LMS-ALE Filter using Vedic Algorithm

Jintu K Joseph[1]* and Purushotham U[2]

[1]Department of VLSI and Embedded systems, Visvesvaraya Technological University, PESIT, Bangalore, India
[2]Department of Electronics and Communications, Visvesvaraya Technological University, PESIT, Bangalore, India

Abstract

ALE or adaptive line Enhancers are special kinds of adaptive filters widely used in noise cancellation circuits. In circuits where we don't have any prior knowledge of signal and noise, fixed filters unit never works good. Among adaptive filter ring algorithms LMS algorithm is very common, in our work also we use LMS algorithm. LMS-ALE filters removes the sinusoidal noise signals present in the channel by calculating the filter coefficients in every iteration. LMS-ALE filter has large number of multiplier units. FFT or Fast Fourier Transform blocks present in LMS algorithm again consist of large array of multiplier units. Optimization of LMS-ALE filter lies must start from optimization of multiplier blocks. Here we use Vedic "Vertical and crosswise algorithm" for multiplier design. When compared to conventional booth multiplier based LMS-ALE filter units, Vedic multipliers gives more performance in areas like resource utilization, power requirement, delay etc. The work includes designing Vedic multipliers, complex Vedic multipliers, redesigning Radix-8 FFT using Vedic multipliers, redesigning LMS block using Vedic FFT, redesigning LMS ALE filter using Vedic multipliers and Vedic LMS blocks. Major part of design is done in Verilog using Xilinx ISE design suite. ADC block present in LMS-ALE filter is done in Matlab version 2013.

Keywords: ALE filter; LMS algorithm; Vedic algorithm; Fast multipliers; Booth multipliers; Radix 8 FFT; Verilog; Xilinx ISE design suite; MatLab

Introduction

In physical environment noise is automatic signal, in all kinds of signal generated there is some kind of unwanted noise signals, when such a signal is amplified in a communication channel both noise and desired signal gets amplified equally, it reduces the clarity of communication system, hence noise cancellation is inevitable. ALE Filter is a most common noise cancellation system. It uses some kind of adaptive algorithm [LMS algorithm]. LMS adaptive block consist of FFT and inverse FFT blocks as it handles the signal in frequency domain. Multipliers are the most repeated block in LMS-ALE filter, to optimize the performance we need to do the optimization from basic multiplier block. In this work we use Vedic algorithm for doing the multiplication.

Compared to conventional booth algorithm Vedic multipliers requires less partial product adders, hence it improves the performance in terms of delay, resource utilization and power requirement [1,2].

Methodology

ALE filter

Adaptive Line Enhancer [ALE] filter optimized to remove sinusoidal noise signals present in the channel. In the Figure 1 (n) represents the desired signal, n (n) denotes the noise signal, z represents de-correlation function, and the block is flowed by an adaptive predictor unit block. De-correlation eliminates any kind of correlation that may exist between the noise samples. Predictor can make prediction on the sinusoidal component of the noise signals and system will adaptively minimize the instantaneous squared error output. Inputs to the adaptive filter or predictor unit are units behind the original input signal. Therefore, in order to time align the enhanced signal, $\hat{s}(n)$ with the input signal, $x(n)$ the adaptive filter must be able to 'predict ahead' in time by optimizing its filter coefficients in a least squares sense, hence the instantaneous squared error is minimized [3].

Vedic maths

Vedic mathematics or else called Indian mathematics originated in Indian sub-continent at 1200BC. Ancient time significant growth to this field is done by great scholars like Aryabhata, rahmagu pta, M ahavira, Bha skara II, Nilakantha Somayaji etc. the decimal number system that we are using today is first record in Indian mathematic books. Vedic maths is the list of mental mathematical calculation techniques describes d in Vedas. Those mental techniques are combined together and described in special text called "vedic mathematics", it is written by Bharati Krishna Tirthaji, and published the same in the year [4].

Vedic multiplication algorithm

Vertical and crosswise algorithm is one among the 16 Vedic sutras mentioned in the Vedic mathematics. Vertical and crosswise algorithm

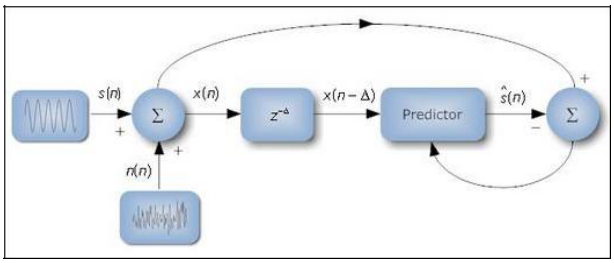

Figure 1: LMS-ALE filter block diagram.

***Corresponding author:** Joseph JK, VLSI and Embedded systems, Visvesvaraya Technological University, PESIT, Bangalore, India, Tel: 0831-2498100; E-mail: Jintuk.joseph@gmail.com

Received July 17, 2015; **Accepted** July 25, 2015; **Published** July 30, 2015

Citation: Joseph JK, Purushotham U (2015) Implementation of LMS-ALE Filter using Vedic Algorithm. J Appl Computat Math 4: 237. doi:10.4172/2168-9679.1000237

is also called as Urdhva Tiryagbhyam. This sutra was used to optimize the multiplier performance. Below given is the multiplication example using vertical and crosswise algorithm. Multiply 33 by 44:

3 3
4 4 ×
1 4 $_2$5 $_1$2 *answer*

Multiplying vertically on the right we get 3 × 4=12, so we put down 2 and carry1 (written$_1$2 above). Then we multiply crosswise and add the two results: 3 × 4 + 3 × 4=24. Adding the carried 1 gives 25 so we put 5 and carry 2 ($_2$5). Finally we multiply vertically on the left, get 3 × 4=12 and add the carried 2 to get 14 which we put down [5].

Complex vedic multiplication

The above flow chart (Figure 2) shows the procedure to compute the addition of ay two signed numbers, Where x1 denotes where both the input say a, b have same sign, if both have same sign then x1 take 1 Else x1=0. When x1 is high it looks like simple unsigned number addition. The result takes the sign of the inputs. If x1=0, we need to compute y1 value. Y1 is equal to 1 if first input say 1>second input b, else it takes 0. If y1=1, then a-b operation was performed as a is larger than b, else b-a operation can be performed, which implies b is larger than a. The output takes the same sign as a larger input operand [6].

The above flow chart (Figure 3) shows the procedure to compute the subtraction of ay two signed numbers, Where x1 denotes where both the input say a, b have same sign, if both have same sign then x1 take 1 Else x1=0. When x1 is high, need to compute y1 value. Y1 is equal to 1 if first input say 1>second input b, else it takes 0. If y1=1, then a-b operation was performed as a is larger than b, else perform b-a operation, which implies b is larger than a. The output takes the same sign as a larger input operand. If x1=0, it looks like simple unsigned number addition. The result takes the sign of the inputs.

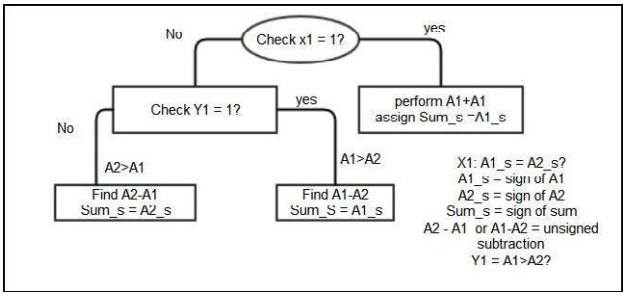

Figure 2: Signed multiplication algorithm [signed addition].

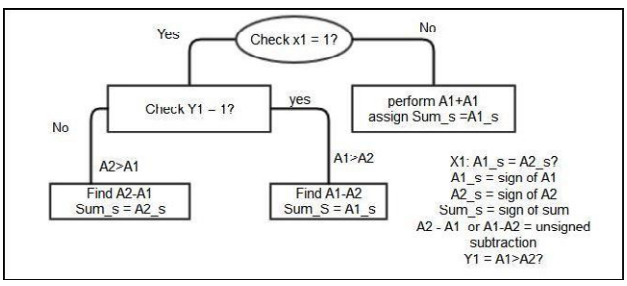

Figure 3: Signed multiplication algorithm [signed subtraction].

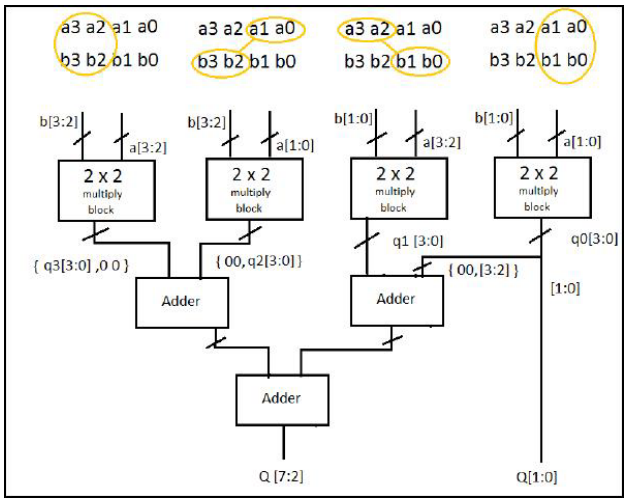

Figure 4: 4x4 Vedic multiplier block diagram.

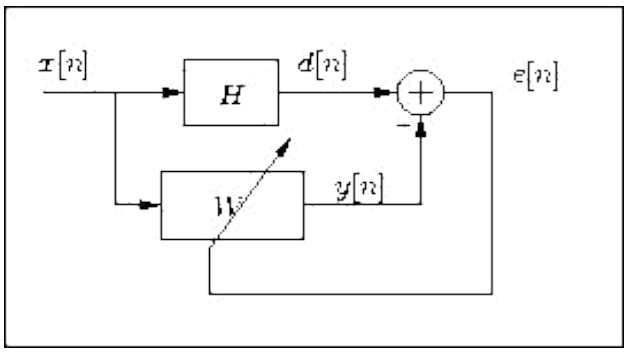

Figure 5: LMS algorithm block diagram.

Vedic 4 x 4 complex multiplier

The above Figure 4 shows the 4 × 4 bit multiplication using Vedic algorithm [7]. It consists of four 2 × 2 Vedic multiplier units, two 4-bit [N bit adder] adders and one 2-bit [N/2 bit adder] units. The orange colored circles indicate the selected operand for multiplication. q [1:0] is equal to Q [1:0], the final result.

LMS

LMS algorithm is the basic adaptive algorithm available, this algorithm helps any system to mimic a desired filter by generating weights adaptively according to the value to error signal (Figure 5) [8].

The LMS algorithm is very useful and easy to compute. The LMS algorithm will perform Ill, if the adaptive system is an adaptive linear combiner as Ill as, if both the n-dimensional input vector *X (k)* and the desire output *d(k)* are available in each iteration, where *X (k)* is

$$X(k)=\begin{bmatrix} x_1(k) \\ x_2(k) \\ . \\ . \\ x_n(k) \end{bmatrix}$$

and the n-dimensional corresponding se t of adjustable weights $W(k)$ is

$$W(k)=\begin{bmatrix} w_1(k) \\ w_2(k) \\ . \\ . \\ w_n(k) \end{bmatrix}$$

By having the input vector $X(k)$, the estimated output y (k), can be computed as a linear combination of the input vector $X(k)$ with the weight vector $W(k)$ as

$$y(k)=X^T(k)W(k)$$

Thus, the estimated error $e(k)$, the difference between the estimated output $y(k)$, and the desired signal $d(k)$, can be computed as

$$e(k)=d(k)-y(k)=d(k)-X^T(k)W(k)$$

Results

The above figure is the simulation result obtained when 2 complex numbers each of 6 bit wide are multiplied, a and b are the input, each has two components a_r, a_i and b_i, b_r. B_r, a_r represents the real part and a_i, b_i represents the imaginary part. A_r_s, a_i_s and b_i_s, b_r_s represents the sign of real part and imaginary part of inputs respectively. The output obtained is c, it also has two components c_r [real part] and c_r (imaginary part).

For smaller modules booth algorithm based multipliers consumes larges resources, for 4 bit multiplier it uses 73 and 19, 4 input LUTs when implemented Booth algorithm and Vedic algorithm. 19 and 42 numbers of occupied slices when implemented using Vedic algorithm and Booth algorithm. When the input gets wider or module becomes larger Vedic algorithm based design consumes larges resources and booth algorithm based design consumes lesser resources [9].

The above Figure 6 shows the power utilization report for Radix 8 FFT [10] implemented using Vedic algorithm and Booth algorithm. These results are obtained from the Xpower analyzer tool of Xilinx ISE software. For radix 8 FFT designed using Vedic algorithm the total power utilization is 194 mw, it consist of three 16 bit Vedic multiplier and two 8 bit multiplier. Hence the total power utilization is sum of power utilization of each multiplier units. For Radix 8 FFT

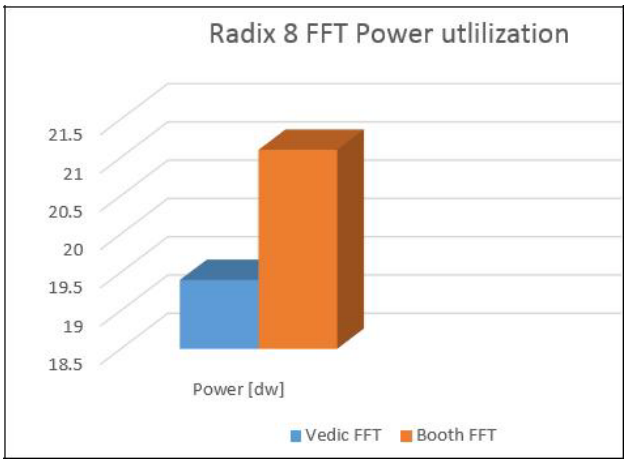

Figure 6: Power utilization comparison graph of radix 8 FFT implemented through Vedic algorithm and Booth algorithm.

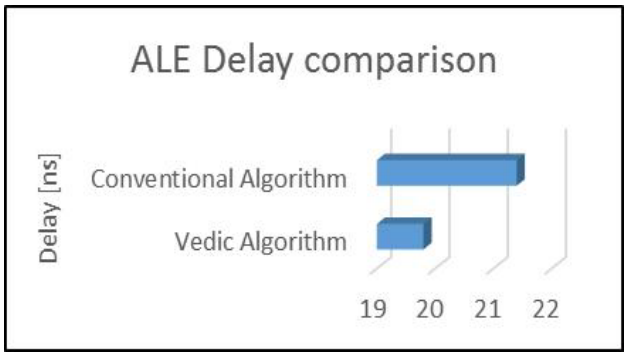

Figure 7: Delay comparison table for ALE implemented using conventional FFT and Vedic FFT.

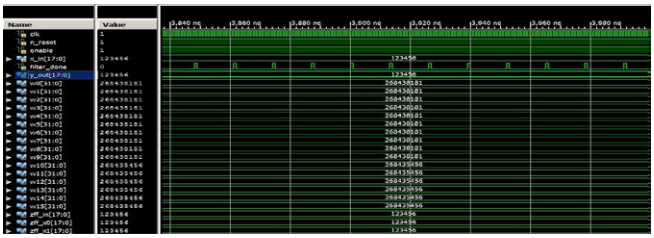

Figure 8: Simulation results for adaptive line enhancing filter using Vedic FFT.

implemented using Booth algorithm the total power utilization is around 211 mw (Figures 7 and 8).

Conclusion

The Fast multiplier design using Vedic algorithm has outstanding performance features in resources utilization, power requirement, delay taken, and area requirement. In this work a generic N x N bit Vedic multiplier which can perform both signed and unsigned multiplication are designed in Xilinx using Verilog. An FFT module which can perform N x N bit Fourier transformation is designed in Xilinx using the Vedic multipliers designed earlier. An ADC module which takes audio input from system, converts the floating point value to 18 bit binary value are modeled in Mat lab. This binary value becomes the input for the ALE. ALE block is designed in Xilinx using Verilog language. The multiplier units in Ale are redesigned using Vedic multipliers. The comparison results between Vedic implementations and conventional implementation are also generated for each stage.

References

1. www.eewiki.net
2. Mehta P, Gawali D (2009) Conventional versus vedic mathematical method for hardware implementation of multiplier. International Conference on Advances in Computing, Control and Telecommunication Technologies. IEEE computer society 640-642.
3. Agrawal J, Matta V, Arya D (2013) Design and implementation of FFT processor using vedic multiplier with high throughput. International Journal of Emerging Technology and Advanced Engineering 3(10): 207-211.
4. Tirtha SBK (1965) Vedic Mathematics. Motilal Banarsidass, India.
5. He Y, He H, Li L, Wu Y, Pan H (2008) The applications and simulation of adaptive filter in noise canceling. IEEE, China.
6. Premananda BS, Pai SS, Bhat SS (2013) Design and implementation of 8 bit vedic multiplier. International journal of advanced research in electrical and electronics and instrumentation engineering 2(3): 5877-5882.

7. Sudeep MC, Sharath Bimba M, Vucha M (2014) Design and FPGA implementation of high speed vedic multiplier. International Journal of Computer Applications 90(16): 6-9.

8. Saha P, Banarjee A, Bhattacharya P, Dandapat A (2011) High speed ASIC design of complex multiplier using vedic mathematics. Proceedings of the 2011 IEEE Students Technical Symposium, IIT Kharagpur.

9. http://www.xilinx.com/

10. Mittal N, Kumar A (2011) Hardware implementation of FFT using vertically and crosswise algorithm. International Journal of Computer Applications 35:17-20.