# Digital Signal Processing Applications with Iterative Logarithmic Multipliers

[1]Aleksej Avramović, [2]Patricio Bulić, [3]Zdenka Babić

[1](aleksej@etfbl.net), [2](patricio.bulic@fri.uni-lj.si), [3](zdenka@etfbl.net)

**Contribution to the State of the Art**

DOI: **10.7251/JIT1102083A** UDC **621.391:004**

**Abstract:** Many digital signal processing applications demand a huge number of multiplications, which are time, power and area consuming. But input data is often corrupted with noise, which means that a few least significant bits do not carry usable information and do not need to be processed. Therefore, approximate multiplication does not affect application efficiency when approximation error is less than noise introduced during data acquisition. This fact enables usage of faster and less power-consuming algorithms that is important in many cases where processing includes convolution, integral transformations, distance computations etc. This paper discusses logarithm-based approximate multipliers and squarers, their characteristics and digital signal processing applications based on approximate multiplications. Our iterative multipliers and squarers contain arbitrary series of basic blocks that involves only adders and shifters; therefore, it is not power and time consuming and enables achieving arbitrary accuracy. It was shown that proposed approximate multipliers and squarers can be used in several signal processing applications without decreasing of application efficiency.

**Keywords:** Approximate multiplication, Digital signal processing

## Introduction

Digital signal processing (DSP) applications often involve algorithms, which demand a huge number of multiplications, which can be time, power and area consuming. Multipliers often process a large amount of data corrupted with noise, which is unnecessary consumption of power and time. For example, many applications involve calculations of integral transformations, such as Fast Fourier Transform (FFT), Discrete Cosine Transform (DFT) and Discrete Wavelet Transform (DWT), after which quantization is applied, like in algorithms for compression [3], [20]. Similarly, frequency leakage, which is common during spectrum analysis, may lead to the estimation of harmonics with certain amount of error. In such applications, which involve error due to quantization or other quantization, sometimes it is more efficient to calculate multiplication results without least significant bits, instead of calculating full-precision results. In other DSP applications convolution or correlation between two signals has to be calculated. Calculation of correlation may involve a large number of multiplications, but it is important to notice that only the maximum or its approximation, of correlation is used; therefore approximated multiplication will not decrease application efficiency. The similar is with noise filtering and other applications that include convolution. Other applications that involve a significant number of multiplications are found in cryptography, object matching and recognition, video and image processing, etc. In applications where the speed of the calculation is more important than accuracy, truncated or logarithm multiplications seem to be suitable methods [6], [12], [18].

## Integer, truncated and logarithmic multipliers

Integer multiplier is one of the simplest methods for computation of the product, but it requires *n* multiplication steps for two *n*-bits unsigned numbers [10]. Such an integer multiplication, where the least-significant bit of the multiplicator is examined, is known as the radix-2 multiplication. Shorter time delay can be achieved by examining *k* lower bits of the multiplicand in each step. Usually, the radix-4 multiplication is used, where two least-significant bits of the multiplicand are examined. This kind of approach usually requires significant hardware resources. The well-known implementation of such a multiplier is an array multiplier, where *n*-2 *n*-bits carry-save adders and one *n*-bits carry-propagate adder is used to implement the *n*-bits array multiplier.

Truncated multipliers are extensively used in digital signal processing where the speed of the multiplication and the area- and power-consumptions are important [11], [18], [22]. However, as mentioned before, there are many applications in DSP where high accuracy is not important. By discarding some of the less significant bits, which can be corrupted with noise, multiplier is less hardware and time consuming. If it is necessary, simple compensation circuits can be applied to reduce the approximation error [6], [14], [15].

Logarithmic multiplication is an approximate multiplication technique that uses the fact that logarithm of the product is a sum of operand logarithms [6], [12], [14], [15]; therefore an operand conversion from integer number system into the logarithm number system (LNS) is used. In more detail, the multiplication of the two operands $N_1$ and $N_2$ is performed in three phases, calculating the operand logarithms, the addition of the operand logarithms and the calculation of the antilogarithm:

$$\log(N_1N_2) = \log(N_1) + \log(N_2) \tag{1}$$

The main advantage of this principle is that multiplication is done by one summation, but approximation of logarithm and antilogarithm conversion introduces error. An iterative approximation of LNS multiplier can be derived from binary representation of a number:

$$N = 2^k\left(1 + \sum_{i=j}^{k-1} 2^{i-j} Z_i\right) = 2^k(1+x) \tag{2}$$

where *k* is place of the most significant bit equals one, so called characteristic number, and *Z* is a bit value at the *i*th position. Because, computers work with binary number system, it is most appropriate to use 2 as logarithm basis, so we can derive:

$$\log_2(N) = \log_2\left(2^k\left(1 + \sum_{i=j}^{k-1} 2^{i-j} Z_i\right)\right) = \log_2\left(2^k(1+x)\right) = k + \log_2(1+x) \tag{3}$$

Previous equation is a basis for Mitchells LNS multiplier approximation first time presented in [16]. Second term in (3) is discarded as an approximation error, but Mitchell also suggested correction term based on if-else logic. Later, several authors tried to simplify correction in various ways. Abed and Sifred [1], [2] derived correction equations with coefficients that are a power of two, reducing the error and keeping the simplicity of the solution. Among the many methods that use look-up tables for error correction in the MA algorithm, McLaren's method [15], which uses a look-up table with 64 correction coefficients calculated in dependence of the mantissas values, could be selected as one that has satisfactory accuracy and complexity. A recent approach for the MA error correction, reducing the number of bits with the value of '1' in mantissas by operand decomposition, was presented by Mahalingam and Rangantathan [14]. LNS multipliers can be generally divided into two categories, one based on methods that use lookup tables and interpolations, and the other based on Mitchell's algorithm (MA) [16], although there is a lookup-table approach in some of the MA-based methods [14].

## Iterative multiplier and squarer

In [5], [6] and [8], algorithm of iterative logarithmic multiplier is presented and analyzed in detail. Iterative calculation of correction terms is one way to deal with LNS multiplier approximation explained in (3). This kind of approach introduces a simple pipelined basic block for calculation of first approximation. Basic block avoids if-else logic, thus significantly reducing the hardware resources. The same basic block can be used for error correction, which represents a significant advantage for simpler hardware implementation. Due to optimal pipelining, correction term calculation may begin before the first approximation is calculated, saving calculation time. The second advantage of iterative approach is the fact that arbitrary number of correction blocks can be added, without increasing the time of delay. From (3), we can derive true value of a product:

$$P_{true} = N_1 N_2 = 2^{k_1}(1+x_1)2^{k_2}(1+x_2) = 2^{k_1+k_2}(1+x_1+x_2) + 2^{k_1+k_2}(x_1 x_2) \quad (4)$$

Combining (4) with (1) it can be shown that:

$$P_{true} = 2^{k_1+k_2} + \left(N_1 - 2^{k_1}\right)2^{k_2} + \left(N_2 - 2^{k_2}\right)2^{k_1} + \left(N_1 - 2^{k_1}\right)\left(N_2 - 2^{k_2}\right) \quad (5)$$

We can see that the last term in (5) demands another multiplication, so by discarding it, we can introduce the first approximation:

$$P_{ap} = 2^{k_1+k_2} + \left(N_1 - 2^{k_1}\right)2^{k_2} + \left(N_2 - 2^{k_2}\right)2^{k_1} \quad (6)$$

which can be implemented easily. In [6], it was proven that adding of finite number of correction terms could reduce an approximation error arbitrary. In Figure 1, the pipelined version of iterative multiplier is shown, while Figure 2 depicts an iterative logarithmic multiplier with one correction circuits.

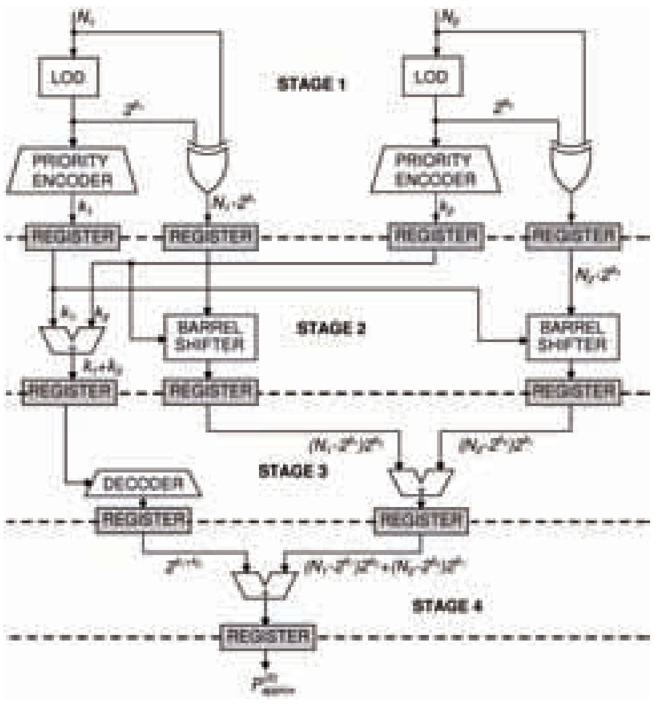

Figure 1. Four stage pipelined version of iterative logarithmic multiplier's basic block.

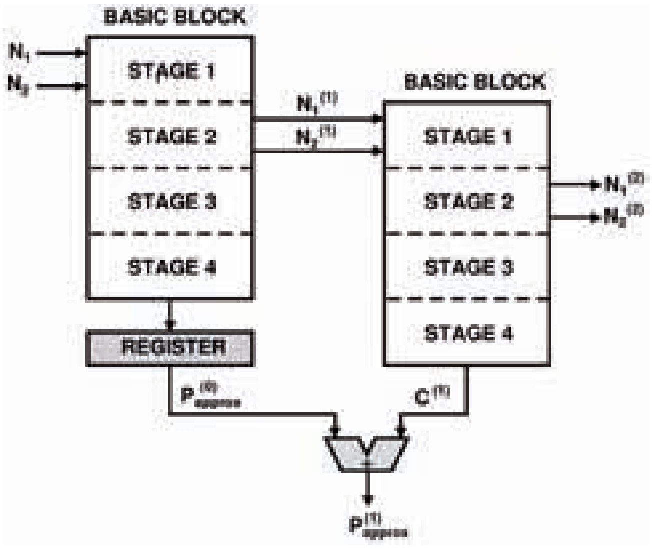

Figure 2. An iterative logarithmic multiplier with basic block and one error correction circuits.

In many digital signal applications, for example, for Euclidean distance calculation, a large number of squaring is employed. For this purpose, an iterative multiplier can be used, but exploiting the fact that the operands are same, can lead to even further hardware simplification. In (7), a simple logarithmic squarer is described. Similar to the multiplier, an approximate equation can be derived for iterative squarer as well. Correct value of a square of $N$ is:

$$S_{true} = \left(2^k + \left(N - 2^k\right)\right)\left(2^k + \left(N - 2^k\right)\right) = 2^{2k} + \left(N - 2^k\right)2^{k+1} + \left(N - 2^k\right)^2 \quad (7)$$

We can see that the last term in (7) demands another square, so by discarding it, we can introduce the first approximation of a square:

$$S_{ap} = 2^{2k} + \left(N - 2^k\right)2^{k+1} \quad (8)$$

Similarly to logarithmic multiplier, in [19] it was proven that finite number of correction circuits could lead to arbitrary small approximation error. The first approximation of square, given by (8), requires one logical shift left (no gates required), one subtraction and one shift by $k$ (Barrel shifter required).

## Motion vector detection

Motion vector is widely used in video compression applications and standards, such as MPEG [23], as well as for moving object location and tracking [7], [21]. In [6] the use of iterative logarithmic multiplier for motion vector detection is described. Direct and most accurate method for motion vector detection is based on technique of block matching, which requires computation of block correlation. The larger is the block, the more multiplications must be calculated for correlation computing. For efficient compression, a compromise between the speed of the calculation and the accuracy of the motion vector is necessary.

We considered matching techniques based on a block correlation. If we take two successive or near video frames and mark them as the reference frame and the observed frame, motion vector technique tries to match blocks from reference frame and observed frame. It is important to find a matching for each block from observed frame (observed block). Motion vector is used as a measure of distance between same object in reference and observed frame. Usually, the difference between successive or near-successive video frames is very small, thus coding that difference may result in faster and more efficient compression. In moving object location and tracking we try to find motion vectors that belong to many objects. If we denote the observed block with $F(i,j)$, where $i$ and $j$ are the pixels' coordinates, and a respective block in the reference frame with $S(i,j)$, assuming the block size is $N$x$N$, then the correlation coefficients $C(x,y)$ are calculated for all positions $(x,y)$ from the reference region as follows:

$$C(x,y)=\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}F(i,j)S(x+i,y+j) \tag{9}$$

As we can see from equation (9), for each pixel in the block, $N$x$N$ multiplications must be calculated, which means, that for the block size is $N$x$N$, $N^4$ multiplications must be calculated. In cases where large frame video stream is processed, this number can be enormous. Hence, other nonlinear methods for block matching can be used, which can introduce matching error [23]. As we can see, calculation of correlation between to blocks can be very computationally expensive, but it is important to notice that only the position of correlation maximum is required for estimation of motion vector. Often we are satisfied with near maximum position, which leads to near accurate motion vector estimation, and will not decrease algorithm efficiency significantly. If approximate multiplication is used instead of full-precision multiplication, most of the correlation coefficients will be decreased for certain percentage. Maximum of correlation function usually will not be different comparing to algorithm with full-precision multipliers. In [6] it was shown that correlation-based block matching technique with iterative logarithmic multiplier and only one error correction circuit introduces mismatch percentage about 3%, while iterative logarithmic multiplier with two correction circuits has negligible mismatch percentage. Mismatch is defined as a difference of maxima of correlation function compared with application with full-precision multipliers. Therefore, approximate multipliers with less power and time consumption can be used in this application.

## SYSTEM IDENTIFICATION

System identification process tries to describe unknown system with linear and time invariant model, which has the same behavioral characteristics as observed unknown system. System identification is usually done with some kind of adaptive filter. Coefficients of model are adapted until the difference between these two systems output becomes arbitrary small. One of the most popular methods for adaptation is based on minimization of mean square difference, and it is called Least Mean Square (LMS) algorithm. LMS algorithm can require a significant number of multiplications, especially for high order system and system with slow convergence. In [4] the fixed-point LMS algorithm, based on iterative logarithmic multiplier is described and tested. If we use $\boldsymbol{h}(n)$ to denote adaptive filter coefficients vector after $n$-th step, we can derive equation for coefficients correction in next step:

$$\mathbf{h}(n+1)=\mathbf{h}(n)-\frac{\mu}{2}\nabla J(n) \tag{10}$$

where $\mu$ is adaptation step unit and $J(n)$ is error cost function for previous step. Error cost function estimates the difference between responses of the unknown system and the model. If we use mean square error for the difference measure, cost function can be estimated as a mathematical expectation of product of adaptive filter response and adaptation error, which yields:

$$\mathbf{h}(n+1)=\mathbf{h}(n)+\mu E\left(\mathbf{x}(n)e(n)\right) \tag{11}$$

where $x(n)$ denotes adaptive filter response and $e(n)$ denotes error in the n-th step. Estimation of mathematical expectation $E$ depends on number of samples and can demand a large number of multiplications, especially if the system order is high. LMS algorithms adapted for fixed-point systems have several rounding error types, for example, input data rounding error, coefficients rounding error, etc. In such algorithm realizations, adaptation time can be decreased with approximate multipliers. An approximation error can be treated as one of the rounding errors. In [4] it was shown that approximation error can be treated as one of rounding errors and that even the logarithmic multiplier with basic block only, would not affect adaptation convergence.

## Context-based image retrieval

It is often necessary to find contextually similar images with query-image, from large number image datasets. Recently, image datasets can consist of several thousands to several hundred thousands images, therefore, searching for visually similar images is likely impossible. Context-based image retrieval (CBIR) system describes every image with appropriate descriptor that is associated to each image. Descriptor has various types of information about image properties, like low level description of color, shapes and textures, and higher-level structures like context. Query is done by calculating distance between query-descriptor and every descriptor from dataset. Descriptors can be relatively high dimensionality, so calculating Euclidean distance between every mage descriptor from database can demand a huge number of squaring.

Logarithmic squaring is a simpler version of logarithmic multiplier; therefore it requires less time and power. In [19] CBIR system based on logarithmic squarer is described, and it was proven that system efficiency is not compromised. Images were represented using Gist descriptor, as it was described in [9] and [17]. Gist descriptor tries to describe image at local spatial level. For color images, descriptor dimensionality can be more that 1500, so calculation of Euclidean distance between two images may demand more than 1500 squaring. Modern digital image databases contain more than several hundred thousand images, so it is obvious that squaring represent time bottleneck for large databases. On the other hand, a large dimensionality descriptors are often corrupted with noise, thus approximate squarers probably will not decrease efficiency of CBIR system. In [19] mean average precision (MAP), of CBIR system with full-precision and approximate logarithmic squarers are compared. Seventy queries were performed to find near duplicate images on dataset contained of 10 000 images. Original images were not considered as neither correct nor incorrect retrievals. It was shown that MAP of system with approximate logarithmic squarer with one error correction circuit has same value as MAP of system with full-precision multiplier. Therefore, it was proven that approximate squarer can be used for image retrieval efficiently.

## Neural network application

The hardware implementations of artificial neural network models have found their place in some niche applications like image processing, pattern recognition, speech synthesis and analysis, adaptive sensors with teach-in ability and so on.

Neural networks offer a high degree of internal parallelism, which can be efficiently used in custom design chips. Neural network processing comprises of a huge number of multiplications, i.e. arithmetic operations consuming a lot of space, time and power. In [13] we have shown that exact matrix multipliers can be replaced with approximate iterative logarithmic multipliers with one error correction circuit. As neural networks have highly adaptive nature, which compensated the erroneous calculation, the replacement of the multipliers does not have any notable impact on the NN processing and learning accuracy. Authors in [13] proposed hardware implementation of the multilayer perceptron with on chip learning ability, which confirmed the potential of the proposed approximate multiplier. Authors in [13] performed experiments on Proben1 benchmark dataset, which showed that the adaptive nature of the proposed neural network model enables the compensation of the errors caused by inexact calculations. The iterative logarithmic multipliers require less resource on a chip, which leads to smaller designs on one hand and on the other hand to designs with more concurrent units on the same chip. A consumption of fewer resources per multiplier also results in more power efficient circuits. In [13] we achieved about 20 % of the reduction in power consumption.

## Conclusion

In this paper, possibilities of usage of approximate logarithmic multiplications and squaring, as a special case of multiplying, in various digital signal applications were described. We proposed logarithmic multipliers which belong to a class of approximate multipliers based on a trade-off principle. Trade-off between accuracy and low time and power consuming is performed in algorithms where low consumption is more important than accuracy. In this paper, several such algorithms are described. Examples of use of iterative logarithmic multipliers in various applications, such as motion vector detection, system identification, image retrieval and neural networks were presented. It was shown that approximation of multiplication does not affect digital signal processing application efficiency, especially when application estimations deal with noise-corrupted data.

## References:

[1] Abed, K.H. and Sifred, R.E. (2003). CMOS VLSI Implementation of a Low-Power Logarithmic Converter, IEEE Transactions on Computers, vol. 52, no. 11, pp. 1421-1433.

[2] Abed, K.H. and Sifred, R.E. (2003). VLSI Implementation of a Low-Power Antilogarithmic Converter, IEEE Transactions on Computers, vol. 52, no. 9, pp. 1221-1228.

[3] Agostini, L.V., Silva, I.S. and Bampi, S. (2007). Multiplierless and fully pipelined JPEG compression soft IP targeting FPGAs, Microprocessors and Microsystems, vol. 31, issue 8, pp. 487–497.

[4] Avramović, A., Risojević, V., Babić, Z. and Bulić, P. (2010). System Identification Using Least Mean Square Algorithm with Logarithmic Multiplier, In Proceedings of 8th Symposium INDEL, Banja Luka, BIH, pp. 134-137.

[5] Babić, Z., Avramović, A. and Bulić, P. (2008). An Iterative Mitchell's Algorithm Based Multiplier, In Proceedings of The IEEE Symposium on Signal Processing and Information Technology, Sarajevo, BIH, pp. 303-308.

[6] Babić, Z., Avramović, A. and Bulić, P. (2011). An Iterative Logarithmic Multiplier, Microprocessors and Microsystems, vol. 35, issue 1, pp. 23-33.

[7] Babić, Z., Ljubojević, M. and Risojević, V. (2011). Indoor RFID Localization Improved by Motion Segmentation, In Proc. 7th International Symposium on Image and Signal Processing and Analysis, pp. 271-276.

[8] Bulić, P., Babić, Z. and Avramović, A. (2010). A Simple Pipelined Logarithmic Multiplier, In Proceedings of 28th International Conference on Computer Design ICCD, Amsterdam, Netherlands, pp. 235-240.

[9] Douze, M., Jegou, H., Sandhawalia, H., Amsaleg, L. and Schmid, C. (2009). Evaluation of gist descriptors for web-scale image search, in International Conference on Image and Video Retrieval, ACM.

[10] Hennessy, J.L. and Patterson, D.A. (2007). Computer Architecture: A Quantitative Approach, fourth ed., Morgan Kauffman Pub.

[11] Kidambi, S.S., El-Guibaly, F. and Antoniou, A. (1996). Area-efficient multipliers for digital signal processing applications, IEEE Transactions Circuits and Systems II: Analog and Digital Signal Processing, vol. 43, no. 2, pp. 90–95.

[12] Kong, M.Y., Langlois, J.M.P. and Al-Khalili, D. (2008). Efficient FPGA implementation of complex multipliers using the logarithmic number system, In IEEE International Symposium on Circuits and Systems, ISCAS, pp. 3154–3157.

[13] Lotrič, U. and Bulić, P. (2011). Logarithmic multiplier in hardware implementation of neural networks, in: A. Dobnikar, U. Lotric, B. Ster (Eds.), ICANNGA (1), volume 6593 of Lecture Notes in Computer Science, Springer, pp. 158–168.

[14] Mahalingam, V. and Rangantathan, N. (2006). Improving Accuracy in Mitchell's Logarithmic Multiplication Using Operand Decomposition, IEEE Transactions on Computers, vol. 55, no. 2, pp. 1523-1535.

[15] McLaren, D.J. (2003). Improved Mitchell-based logarithmic multiplier for low-power DSP applications, In Proceedings of IEEE International SOC Conference, pp. 53-56.

[16] Mitchell, J.N. (1962). Computer multiplication and division using binary logarithms, IRE Transactions on Electronic Computers, pp. 512–517.

[17] Oliva, A. and Torralba, A. (2001). Modeling the shape of the scene: a holistic representation of the spatial envelope, International Journal of Computer Vision, vol. 42, no. 3, pp. 145–175.

[18] Rais, M.H. (2009). Efficient hardware realization of truncated multipliers using FPGA, International Journal of Applied Science, vol. 5, no. 2, pp. 124–128.

[19] Risojević, V., Avramović, A., Babić, Z. and Bulić, P. (2011). A Simple Pipelined Squaring Circuit for DSP, In Proceedings of 29th International Conference on Computer Design ICCD, Amherst, MA, USA, pp. 162-167.

[20] Srot, S. and Zemva, A. (2007). Design and implementation of the JPEG algorithm in integrated circuit, Electrotechnical Review, vol. 74, no. 4, pp. 165–170.

[21] Tekalp, A. M. (1995). Digital Video Processing, Prentice Hall.

[22] Van, L.-D. and Yang, C.-C. (2005). Generalized low-error area-efficient fixed-width multipliers, IEEE Transactions Circuits and Systems I: Regular Paper, vol. 52, no. 8, pp. 1608–1619.

[23] Watkinson, J. (2004). The MPEG Handbook: MPEG-1, MPEG-2, MPEG-4, second ed., Focal Press.

Submitted: December16, 2011
Accepted: December 31, 2011