Review scientific paper

Informacije MIDEM
Journal of Microelectronics, Electronic Components and Materials
Vol. 43, No. 4 (2013), 203 – 211

# Fixed-point Multiplication and Division in the Logarithmic Number System: a Way to Low-Power Design

Patricio Bulić

University of Ljubljana, Faculty of Computer and Information Science, Ljubljana, Slovenia

**Abstract:** In this article we present the use of the logarithmic number system (LNS) to implement fixed-point multiplication and division. LNS has recently attracted the interest of researchers for its low-power properties. The reduction of power dissipation in LNS arises from the simplification of basic arithmetic operations.

In this paper we give a survey of the recently proposed digital circuits for logarithm and anti-logarithm conversion and multiplication and division in LNS. We also compare these methods in terms of accuracy, area, time and power. Finally, we give an overview of the real world applications that benefit form the use of LNS arithmetic.

**Keywords:** computer arithmetic, logarithm number system, power dissipation

# Zmanjševanje porabe v vezjih  z uporabo celoštevilskega množenja in deljenja v logaritemskem številskem sistemu

**Izvleček:** V tem preglednem članku predstavimo uporabo logaritemskega številskega sistema za implementacijo množenja in deljenja v fiksni vejici. Logaritemski številski sistem zadnje čase vabi pozornost raziskovalcev zaradi lastnosti nizke disipacije moči. Glavni razlog za manjšo porabo energije leži v lastnosti logaritemskega številskega sistema, da poenostavlja osnovne aritmetične operacije. V članku podamo pregled predlaganih nenatančnih digitalnih vezij za množenje in deljenje v logaritemskem sistemu ter jih primerjamo glede na natančnost ter porabo prostora, časa in moči. Na koncu podamo pregled vsakdanjih aplikacij v katerih lahko učinkovito uporabimo nenatančno logaritemsko aritmetiko in tako zmanjšamo porabo energije, ne da bi bistveno vplivali na natančnost in zanesljivost algoritmov.

**Ključne besede:** računalniška aritmetika, logaritemski številski sistem, poraba moči

* Corresponding Author's e-mail: patricio.bulic@fri.uni-lj.si

## 1 Introduction

Real time digital signal processing applications often use data from acquisition devices, which are corrupted with noise. If in such applications area, power or processing speed are more important than accuracy, then faster, less-power and less-hardware consuming approximate solutions can be used. Multiplication and division are among the most used arithmetic operations in digital signal processing, neural networks and adaptive systems. A great number of repeated multiplications and divisions impose a significant power and time consumption. An approximation of these operations should not lead to considerable degradation of applications' performance; therefore the introduced error should be as low as permitted by an application.

The choice of the number system affects the power dissipation, since the number system has an effect on several levels of the design abstraction. In particular, the appropriate selection of the number system can reduce power dissipation, because it can reduce the number of the operations, the strength of the operators and the activity of the data.

This review paper is organized as follows: in the following subsections we give an overview of the switching power consumption in digital circuits and discuss how the choice of the number system can affect power dissipation. In Section 2 various methods for multiplication in LNS are presented. Section 3 gives an overview of the algorithms used in binary logarithmic and antilogarithmic approximations. A general logarithmic multiply/divide unit is also discussed. In section 4 we give an overview of the real world applications that benefit from the usage of the arithmetic in the logarithm number system. We conclude the paper in Section 5.

### 1.1 Power dissipation

Power dissipation is a prime design issue, mainly due to the growing need for portable electronic devices. Low-power design requires optimization at all levels of abstraction. Dynamic power consumption is due to charging and discharging of capacitance. The energy consumed for N clock cycles is

$$E_N = n(N) \cdot C \cdot V_{DD}^2 \tag{1}$$

where *n(N)* is the number of 0 to 1 transitions in *N* clock cycles, *C* is switching capacitance and $V_{DD}$ is supply voltage. The switching power is given as energy per transition and can be expressed as:

$$P_{avg} = \lim_{N\to\infty} \frac{E_N}{N} \cdot f = \lim_{N\to\infty} \frac{n(N)}{N} \cdot f \cdot C \cdot V_{DD}^2 \tag{2}$$

Where *f* is the clock frequency. The term $\lim_{N\to\infty} \frac{n(N)}{N}$ represents the switching activity on a signal line, and is denoted as $\alpha_{0\to1}$:

$$\lim_{N\to\infty} \frac{n(N)}{N} = \alpha_{0\to1} \tag{3}$$

The switching power dissipation in a circuit is then given as:

$$P_{avg} = f \cdot C \cdot V^2 \cdot \alpha_{0\to1} \tag{4}$$

Three main principles of power reductions are:
- reduction of voltage,
- reduction of the switching activity, i.e. minimize spurious glitches, and
- reduction of the area complexity, i.e. reduce the switching capacitance.

Low-power design usually requires operation at lowest possible voltage and clock speed. Glitches are temporary changes in the value of the output and as such they represent unnecessary transitions. They are caused due to the skew in the input signals to a gate. Gate sizing and path balancing techniques like pipelining can reduce glitches.

In this paper we will focus on two techniques that affect all factors in power Eq. (4):
- the choice of the number system,
- optimization of arithmetic circuits.

### 1.2 Logarithmic number system

Stouraitis and Paliouras [1,2] studied the impact of the logarithmic number system on the power dissipation. They showed that, if the data distribution is uniform, the probability of the *i*-th bit transition from 0 to 1 (probability of the bit assertion) is constant in the fixed-point number representation, and is 0.25 for each bit in a word. On the other side they showed that the probabilities of bit assertion in LNS operands are not constant – they depend on the significance of a bit. The probability of bit assertion for the more significant bits is substantially lower than the probability of bit assertion for the less significant bits. This is due to the inherent data compression probability of the logarithm and this behavior leads to a reduction of the average switching activity in the entire word. The study [1] showed that the activity savings percentage can be more than 15%. Paliouras and Stouraitis report that approximately a two-times reduction in power dissipation is possible for operations with word size of 8 to 14 bits.

The logarithmic number system can simplify certain arithmetic operation and can reduce the strength of the operators. For example, the multiplication is reduced to the addition and the division is reduced to subtraction. In order tu use this benefit of the logarithm number system, a conversion circuitry is required to perform the conversion from the fixed-point number representation to LNS and vice-versa. The basic arithmetic operations and their counterparts are presented in Table 1.

**Table 1:** Basic arithmetic operations in the fixed-point number representation and their LNS counterparts.

| Fixed-point operation | Logarithmic operation |
|---|---|
| A = B . C | $\log_2 A = \log_2 B + \log_2 C$ |
| A = B / C | $\log_2 A = \log_2 B - \log_2 C$ |
| $A = B^2$ | $\log_2 A = (\log_2 B) << 2$ |

Paliouras and Stouraitis in [2] claim that LNS due to savings in signal activity and the reduction of the strength of the operators can be a successful candidate for the implementation of low-power arithmetic circuits.

## 2 Multiplication in LNS

Multiplication has always been a hardware-, time- and power consuming arithmetic operation, especially for large-value operands. This bottleneck is even more emphasized in digital signal processing (DSP) applications that involve a huge number of multiplications. In many real-time DSP applications, speed is the prime target and achieving this may be done at the expense of the accuracy of the arithmetic operations. Signal processing deals with signals distorted with the noise caused by non-ideal sensors, quantization processes, amplifiers, etc., as well as algorithms based on certain assumptions, so inaccurate results are inevitable. For example, a frequency leakage causes a false magnitude of the frequency bins in spectrum estimations. The signal-compression techniques incorporate quantization after a cosine or wavelet transform. When transform coefficients are quantized, instead of calculating high-precision coefficients and then truncating them, it is reasonable to spend less resources and produce less accurate results before the quantization. In many signal-processing algorithms, which include correlation computations, the exact value of the correlation does not matter; only the maximum of the correlation plays a role. Additional small errors introduced with multipliers, as mentioned in the application described and others, do not affect the results significantly and they can still be acceptable in practice.

Logarithmic multiplication introduces an operand conversion from integer number system into the logarithm number system. The multiplication of the two operands N1 and N2 is performed in three phases, calculating the operand logarithms, the addition of the operand logarithms and the calculation of the antilogarithm, which is equal to the multiple of the two original operands.

The main advantage of this method is the substitution of the multiplication with addition, after the conversion of the operands into logarithms. 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) [3], although there is a lookup-table approach in some of the MA-based methods [4]. Generally, MA-based methods suppressed lookup tables due to hardware-area savings. However, this simple idea has a significant weakness: logarithm and anti-logarithm cannot be calculated exactly, so there is a need to approximate the logarithm and the antilogarithm.

The binary representation of the number *N* can be written as:

$$N = 2^k \cdot \left(1 + \sum_{i=0}^{k-1} b_i \cdot 2^{i-k}\right) = 2^k \cdot (1+x) \qquad (5)$$

where k is a characteristic number, i.e. the place of the leading-one bit, $b_i$ is a bit value at the *i*-th position and *x* is the fraction. The logarithm with the basis 2 of the number *N* is:

$$\log_2 N = \log_2(2^k \cdot (1+x)) = k + \log_2(1+x) \qquad (6)$$

The expression $\log_2(1+x)$ is usually approximated and the approximation affects the accuracy. Many methods for the logarithm and anti-logarithm approximation have been proposed in the past [3,4,5,6,7,8,9].

### 2.1 Mitchell's algorithm based multiplier

One of the oldest methods to approximate the multiplication and division in LNS is Mitchell's based logarithm computation [3] that approximates the logarithm with piecewise straight lines:

$$\log_2 N \approx k + x \qquad (7)$$

The Mitchell's based approximation (MA) of the logarithm of the product is:

$$\log_2(N_1 \cdot N_2) \approx k_1 + k_2 + x_1 + x_2 \qquad (8)$$

Mitchell proposed the following approximation of the product:

$$(N_1 \cdot N_2)_{MA} \approx f(x) = \begin{cases} 2^{k_1+k_2} \cdot (1 + x_1 + x_2), & \mid x_1 + x_2 < 1 \\ 2^{k_1+k_2+1} \cdot (x_1 + x_2), & \mid x_1 + x_2 \geq 1 \end{cases} \qquad (9)$$

The approximation of the product requires the comparison of the sum of mantissas, i.e. the product approximation depends on the carry bit from the sum of mantissas. The Mitchell's method computes the product using only shift and add operations. The architecture of the multiplier proposed in [11] is depicted in Figure 1.

The Mitchell's based multiplication produces a significant error. The relative error increases with the number of bits with the value of '1' in the mantissas. The maximum possible relative error for MA multiplication is around 11%, and the average error is around 3.8%. The Mitchell's multiplier consumes only 17% of the power consumed by a standard fixed-point multiplier.

Numerous attempts have been made to improve the MA's accuracy. Hall [10], for example, derived differ-

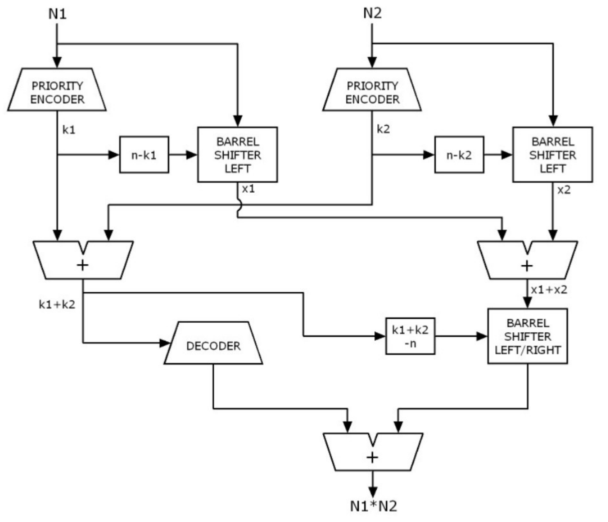

**Figure 1:** The architecture of the Mitchell's multiplier.

ent equations for error correction in the logarithm and antilogarithm approximation in four separate regions, depending on the mantissa value, reducing the average error to 2%, but increasing the complexity of the realization. Among the many methods that use look-up tables for error correction in the MA algorithm, McLaren's method [4], which uses a look-up table with 64 correction coefficients calculated in dependence of the mantissas values, can be selected as one that has satisfactory accuracy and complexity.

## *2.2 Operand decomposition in the Mitchell's algorithm*

Mahalingam and Ranganathan [11,12] proposed a method based on operand decomposition for improving the accuracy of Mitchell's logarithmic multiplication. The operand decomposition is applied to the inputs as a preprocessing step to Mitchell's logarithmic multiplication in order to reduce the number of ones in the input operands. In this method the *n*-bits input operands *X* and *Y* are decomposed into *n*-bits operands:

$$A = X \vee Y$$
$$B = X \wedge Y$$
$$C = \overline{X} \wedge Y$$
$$D = X \wedge \overline{Y}$$

where $\vee$ and $\wedge$ denote bitwise OR and AND operations. The product is then calculated as

$$X \cdot Y = (A \cdot B) + (C \cdot D) \tag{10}$$

and the computation requires two Mitchell-based multipliers. The authors reported that the average relative error has been decreased to 2.1%, while the area is doubled in comparison to MA. It has been reported that operand decomposition based multiplier consumes only 30% of the power consumed by a standard fixed-point multiplier. The architecture of the operand decomposition based multiplier proposed in [11] is depicted in Figure 2.

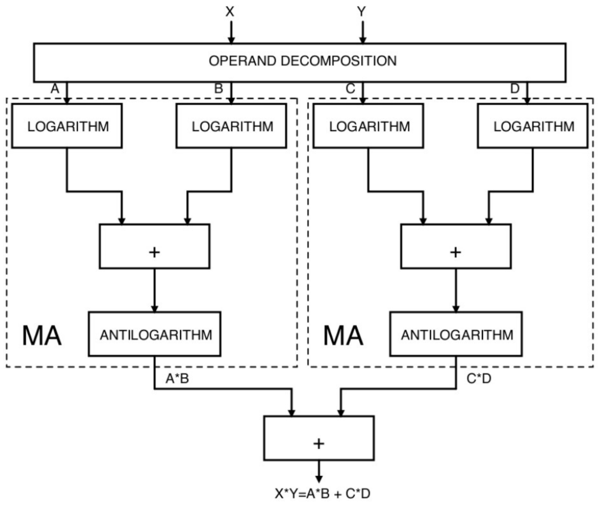

**Figure 2:** The architecture of the operand-decomposition based multiplier.

## *2.3 Babic's multiplier*

Babic et al. [13,14] proposed a solution that simplifies logarithm approximation introduced by Mitchell in Eq. (9) and introduces an iterative algorithm with various possibilities for achieving the multiplication error as small as required and the possibility of achieving the exact result. By simplifying the logarithm approximation introduced in Eq. (9), the correction terms could be calculated almost immediately after the calculation of the approximate product has been started. In such a way, the high level of parallelism can be achieved by the principle of pipelining, thus reducing the complexity of the logic required by Eq. (9) and increasing the speed of the multiplier with error rectification circuits. From the binary representation of the numbers in Eq. (5), we can derive a correct expression for the multiplication:

$$N_1 \cdot N_2 = (2^{k_1} + (N_1 - 2^{k_1}) \cdot (2^{k_2} + (N_2 - 2^{k_2})) \tag{11}$$

$$N_1 \cdot N_2 = 2^{k_1} 2^{k_2} + 2^{k_2}(N_1 - 2^{k_1}) +$$
$$+ 2^{k_1}(N_2 - 2^{k_2}) + (N_1 - 2^{k_1})(N_1 - 2^{k_1}) \tag{12}$$

The last term $(N_1 - 2^{k_1})(N_1 - 2^{k_1})$ requires the multiplication, while all other terms can be calculated using only shift and add operation. Babic et. al proposed the following approximation of the product:

$$N_1 \cdot N_2 \approx P_{approx} = 2^{k_1}2^{k_2} + 2^{k_2}(N_1 - 2^{k_1}) + 2^{k_1}(N_2 - 2^{k_2}) \tag{13}$$

Discarding the term $(N_1 - 2^{k_1})(N_1 - 2^{k_1})$ from Eq. (12) gives an approximate product $P_{approx}$, which can be underestimated for at most 25%. The architecture of the Babic's multiplier is depicted in Figure 3.

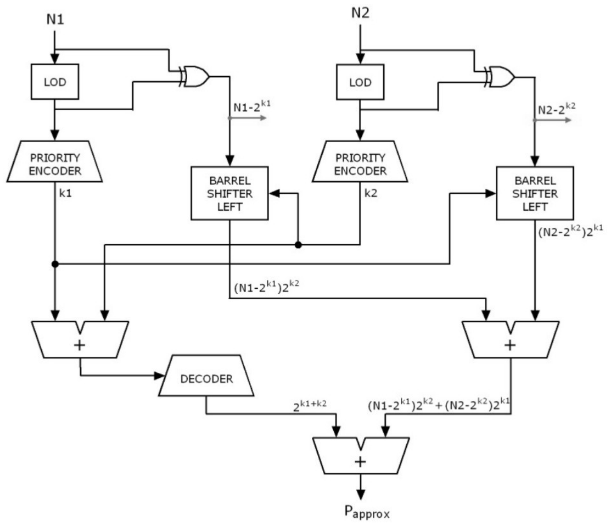

**Figure 3:** The architecture of the Babic's multiplier.

Babic et al. also proposed a simple scheme for error rectification. Instead of neglecting the term $(N_1 - 2^{k_1})(N_1 - 2^{k_1})$ in Eq. (12), we can calculate the product $(N_1 - 2^{k_1})(N_1 - 2^{k_1})$ in the same way as $P_{approx}$, i.e. using Eq. (13). In such a way the error rectification can start immediately after removing the leading ones from the both input operands. By repeating the above procedure we can approximate the product to an arbitrary precision without using the exact multiplier. Babi´c et al. showed that in the worst case scenario the relative error decays exponentially with the rate $2^2$ per pass. The average relative error with one error rectification has been decreased to 0.97%.

It has been reported in [13] that Babic's multiplier consumes only 30% of the power consumed by a standard fixed-point multiplier. Area used by Babic's multiplier is 93% of the area used by the operand decomposition based multiplier.

## 3 Logarithmic converters and division

To improve the error in logarithm and antilogarithm conversion in Mitchell's algorithm many efficient logarithm and antilogarithm converters have been proposed in literature [6] [7] [8] [9]. The logarithm of the binary number *N* is:

$$\log_2 N = k + \log_2(1+x) \tag{14}$$

In order to calculate $\log_2 N$ we have to approximate the term $\log_2(1+x)$

### *3.1 Abed-Siferd logarithm and antilogarithm approximation*

Abed and Siferd [6,7,15] proposed a low-power logarithmic converter where they further divided intervals for piecewise linear curve that is used to approximate the logarithm and anti-logarithm. They proposed a 2-region approximation of $\log_2(1+x)$ as follows:

$$\log_2(1+x) \approx \begin{cases} x + \frac{1}{4}x_{3MSB}, & x \in [0.0, 0.5] \\ x + \frac{1}{4}\bar{x}_{3MSB}, & x \in [0.5, 1.0] \end{cases} \tag{15}$$

Where $\bar{x}_{3MSB} = (1 - x_{3MSB} - 2^{-3})$ and $x_{3MSB}$ represents the tree most significant bits in the mantissa x. The proposed correction equations use only the three most significant bits of the mantissa and all coefficients are restricted to powers of 2. The block diagram of the proposed logarithmic converter is depicted in Figure 4.

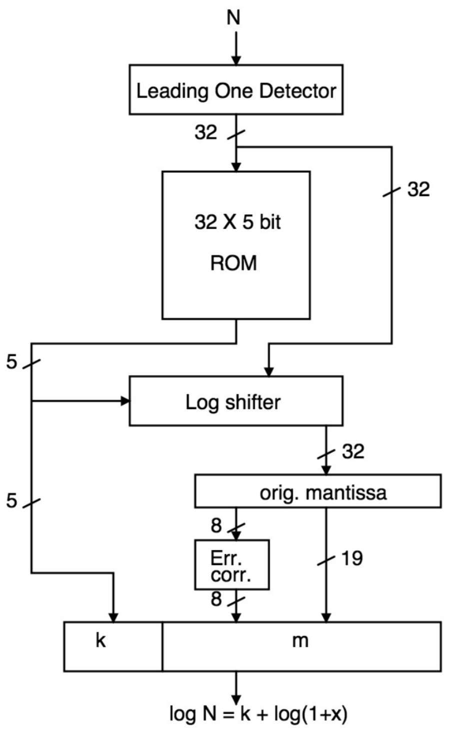

**Figure 4:** A block diagram of the Abed-Siferd low-power logarithmic converter [6].

Abed and Siferd also proposed a low-power antilogarithmic converter [7]. If the logarithm of the number *N* is $\log_2 N = k + \log_2(1+x) = k + m$ then the antilogaritm is

$$N = 2^k 2^m \tag{16}$$

To approximate 2$^m$ Abed and Siferd proposed the following 2-region approximation:

$$2^m \approx f(x) = \begin{cases} m + \frac{3}{16}\bar{m}_{7MSB} + \frac{13}{16} + 2^{-10} + 2^{-11}, & \Big|\quad m \in [0,0.5) \\ m + \frac{3}{16} m_{7MSB} + \frac{13}{16}, & \Big|\quad m \in [0,0.5) \end{cases} \tag{17}$$

Where $\bar{m}_{7MSB} = (1 - m_{7MSB} - 2^{-7})$ and $m_{7MSB}$ represents the seven most significant bits in the number *m*.

### 3.2 Multiplication and division using logarithmic converters

Multiplication and division are reduced to addition and subtraction after the input operands have been converted to logarithms. The conversion from LNS to the fixed-point notation is required after the addition or subtraction in LNS. The general architecture of a multiplier and/or divider using LNS is depicted in Figure 5.

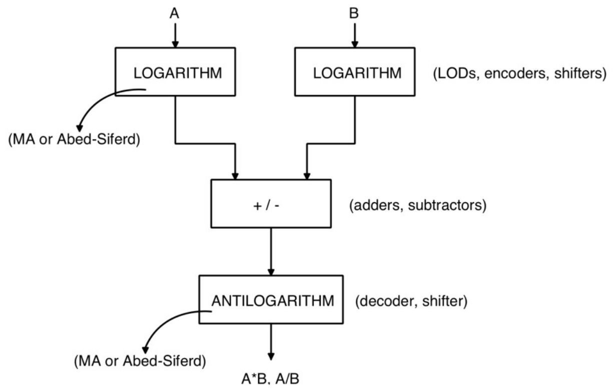

**Figure 5:** General architecture of a multiply/divide unit in LNS.

For logarithm and antilogarithm approximation we can use any of the methods proposed in [6] [7] [8] [9]. Using Abed-Siferd approximation, the relative error is reduced to 2.5% and the multiplier circuit uses only 40% of the area used by the array multiplier.

## 4 The appropriateness of LNS arithmetic in real-world applications

In this section we give an overview of the real world applications that benefit form the use of LNS arithmetic.

### 4.1 3-D graphics systems

In [16] a 32-bit fixed-point logarithmic arithmetic unit (LAU) is proposed for the possible application to mobile three-dimensional (3-D) graphics system. The proposed logarithmic arithmetic unit performs division, reciprocal, square-root, reciprocal-square-root and square operations. The unit is composed of two binary logarithmic converters, a calculation unit and a binary antilogarithmic converter. Instead of general 2-region piecewise-linear interpolation of the logarithm, the authors proposed a new 8-region piecewise-linear interpolation approximation algorithm, which is used in the proposed binary logarithmic converter block. Also, the piecewise interpolation method is used for the binary antilogarithmic converter.

The proposed LAU is implemented with 0.18 um CMOS technology and is verified in the 3-D graphics processing software environment before its chip is implemented. The test model consisted of 1700 polygons with lighting and texture mapping. The screen resolution was 512x512 and the texture size was 256x256. All 3-D graphics operations (vertex matrix transformation, vertex lighting, rendering, and texture mapping are performed by LAU.

The authors [16] claim that no noticeable difference was found by naked eyes between the two images. The small error range was within a tolerable range for the small screen size images of the mobile system. By using the LAU in fixed-point arithmetic, the performance is improved by five times compared with the complex radix-4 method. The power consumption of the LAU is 2.18 mW, while the power consumption of radix-4 unit is 4.29 mW, which is 1.97 times that of LAU's.

### 4.2 Motion vector

The authors in [13] considered the calculation of motion vectors using Babic's logarithmic multiplier. In video compression, a motion vector is used to represent a macro-block in a picture based on the position of this macro-block (or a similar one) in another picture, called the reference picture. For a block from observed frame (observed block), block-matching techniques try to find the best matching block in the reference frame. When the best matching is found, the displacement is calculated and used as a motion vector, in applications like MPEG video compression.

In order to show the usability of the Babic's multiplier for motion-vector calculations, the described block-matching algorithm was applied on a selected region of the successive CT scan frames.

The results obtained with the Babic's multiplier with one error correction term are compared with the results of the regular multiplication. The mismatching percentage for the Babic's multiplier was between 3.17 and 3.9. These results  prove that the percentage of mismatching is very low. Due to the heavy computational requirements, the block matching is often performed in two stages, a rough estimation of a moving vector and then an accurate refinement. In video compression, the errors in motion vectors will only slightly decrease the compression, but will speed up the compression algorithms and minimize power dissipation.

### *4.3 Multilayer perceptron with a highly parallel neural unit*

Neural network processing comprises a huge number of multiplications. To gain as much as possible from the custom design, multiplications must be performed in parallel. However, multiplication circuits consume a lot of resources, time and power. Since the resources on a chip are limited, different strategies are applied to overcome the limitations. The first idea is to replace the floating-point arithmetic with fixed-point arithmetic. However, to further increase the performance the exact fixed-point matrix multipliers must be replaced with some approximate solutions. The hardware neural network presented in [17,18] is built around an iterative logarithmic multiplier, which can use many levels of correction circuits to iteratively approximate a product to the arbitrary precision. It also enables the pipelined design of correction circuits, which significantly reduce the propagation time of a signal through a circuit. The logarithmic multiplier with only one correction circuit is enough to reduce the multiplication error, on average, to less than 1%. In contrast to the majority of the proposed designs, where a special hardware unit is used for each neuron, the design presented in [17,18] contains only one highly parallel neural unit, which is capable of the fast parallel calculation of a neuron output. Since the same circuit can be used in forward and backward passes, it is more suitable for hardware neural network designs targeting small FPGA chips. The performance of the proposed hardware neural network with iterative logarithmic multipliers was compared to the usual software models and hardware neural network with exact matrix multipliers.
The neural-network models were tested on the PROBEN1 benchmark dataset, consisting of classification and approximation problems. Due to the highly adaptive nature of neural network models, which compensated the erroneous calculation, the replacement of the multipliers did not have any notable impact on the models' processing and learning accuracy. Furthermore, the consumption of fewer resources per multiplier also results in more power efficient circuits. The power consumption, which was reduced by roughly 20%, makes the hardware neural network models with logarithmic multipliers favorable candidates for battery-powered applications.

### *4.4 Adaptive Control Systems*

The authors in [19] implemented the Kalman filter for object tracking using the logarithmic multipliers and reciprocal units. Object-tracking systems require a large number of complex arithmetic operations, which impose a significant power dissipation.  Since adaptive algorithms can adjust to changes in the environment, they are also able to compensate for the computational errors, internally produced by the arithmetic units in LNS. The Kalman filter is able to iteratively update and estimate the underlying system state given a series of inaccurate and uncertain measurements. It can grasp the dynamics of a given system solely by observing its location over time. The Kalman filter equations comprise a large number of matrix multiplications and a matrix inverse.

To make these computations effective, the authors in [19] proposed  two special purpose hardware units, a dot product unit and a multiply-and-divide unit. The dot product unit exploits the fact that any matrix multiplication can be decomposed to a series of independent dot products. Each dot product unit consists of four logarithmic multipliers and three adders. Four dot-product units are integrated the larger multiply-and-divide (MAD) unit that is capable of computing four dot products in parallel. The MAD unit also includes two logarithmic reciprocal units that serve in computation of the inverse.

The authors in [19] showed that the application of approximate arithmetic units importantly reduces power dissipation of the Kalman filter circuitry. The approximate units dissipate less power than the exact ones, but they usually need more clock cycles to get the result. Nevertheless, the implementations with logarithmic multipliers and reciprocal units are smaller and more energy efficient than the implementations with exact arithmetic. In such a way, the energy consumption per one iteration in the Kalman filter reduces to 70% - 80% of exact arithmetic. Moreover, the logarithmic arithmetic units reduce the critical path for up to 40%.

## 5 Conclusion

In this study we discuss the use of logarithmic number system to reduce the power in digital systems. The choice of LNS can lead to substantial savings in power dissipation, since LNS affects the signal activity and the strength of operators. The replacement of multipliers and dividers in adaptive systems will not have any notable impact on the systems' processing and learning accuracy. The same is true for signal processing systems, as signal processing deals with signals distorted with the noise caused by non-ideal sensors, quantization processes, amplifiers, etc., as well as algorithms based on certain assumptions, so inaccurate results are inevitable.

## 6 Acknowledgements [11]

This study was supported by Slovenian Research Agency (ARRS) under grant P2-0359 (National research program Pervasive computing).

## 7 References

1. T. Stouraitis and V. Paliouras, "Considering the Alternatives in Low-Power Design," *IEEE Circuits and Devices Magazine*, vol. 17, no. 4, pp. 23-29, July 2001.
2. V. Paliouras and T. Stouraitis, "Signal activity and power consumption reduction using the logarithmic number system," in *The 2001 IEEE International Symposium on Circuits and Systems, 2001. ISCAS 2001*, Sydney, NSW, 2001, pp. 653 - 656.
3. J. Mitchell., "Computer Multiplication and Division using Binary Logarithms," *IRE Transactions on Electronic Computers*, vol. 11, no. 4, pp. 512-517, Aug. 1962.
4. D.J Mclaren, "Improved Mitchell-based logarithmic multiplier for low-power DSP applications," in *Proceedings of IEEE International [Systems-on-Chip] SOC Conference, 2003.*, 2003, pp. 53-56.
5. E.E Swartzlander, "Sign/logarithm Arithmetic for FFT Implementation," *IEEE Transactions on Computers*, vol. 32, no. 6, pp. 526-534, Jun 1983.
6. K.H. Abed and R.E. Siferd, "CMOS VLSI implementation of a low-power logarithmic converter," *IEEE Transactions on Computers*, vol. 52, no. 11, pp. 1421-1433, Nov. 2003.
7. K.H. Abed and R.E. Siferd, "VLSI implementation of a low-power antilogarithmic converter," *IEEE Transactions on Computers*, vol. 52, no. 9, pp. 1221 - 1228, Sept. 2003.
8. D. De Caro, N. Petra, and A.G.M. Strollo, "Efficient Logarithmic Converters for Digital Signal Processing Applications," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 58, no. 10, pp. 667 - 671, Oct. 2011.
9. R. Gutierrez and J. Valls, "Low Cost Hardware Implementation of Logarithm Approximation," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 19, no. 12, pp. 2326 - 2330, Dec. 2011.
10. Ernest L. Hall, D.D. Lynch, and S.J., Dwyer, "Generation of Products and Quotients Using Approximate Binary Logarithms for Digital Filtering Applications," *IEEE Transactions on Computers*, vol. 19, no. 2, pp. 97 - 105, Feb. 1970.
11. V. Mahalingam and N. Ranganathan, "Improving Accuracy in Mitchell's Logarithmic Multiplication Using Operand Decomposition," *IEEE Transactions on Computers*, vol. 55, no. 12, pp. 1523 - 1535, Dec. 2006.
12. V. Mahalingam and N. Ranganathan, "An efficient and accurate logarithmic multiplier based on operand decomposition," in *19th International Conference on VLSI Design, 2006. Held jointly with 5th International Conference on Embedded Systems and Design*, 2006, p. 6.
13. Z. Babić, A. Avramović, and P. Bulić, "An iterative logarithmic multiplier," *Microprocessors and Microsystems*, vol. 35, no. 1, pp. 23-33, Feb. 2011.
14. P. Bulic, Z. Babic, and A. Avramovic, "A simple pipelined logarithmic multiplier," in *2010 IEEE International Conference on Computer Design (ICCD)*, Amsterdam, 2010, pp. 235 - 240.
15. K.H. Abed and R.E. Siferd, "VLSI Implementations of Low-Power Leading-One Detector Circuits," in *Proceedings of the IEEE SoutheastCon, 2006*, Memphis, TN, 2006, pp. 279 - 284.
16. Hyejung Kim, Byeong-Gyu Nam, Ju-Ho Sohn, Jeong-Ho Woo, and Hoi-Jun Yoo, "A 231-MHz, 2.18-mW 32-bit Logarithmic Arithmetic Unit for Fixed-Point 3-D Graphics System," *IEEE EEE Journal of Solid-State Circuits*, vol. 41, no. 11, pp. 2373 - 2381, Nov. 2006.
17. Uroš Lotrič and Patricio Bulić, "Applicability of approximate multipliers in hardware neural networks," *Neurocomputing*, vol. 96, pp. 57-65, Nov. 2012.
18. U Lotric and P Bulic, "Logarithmic Multiplier in Hardware Implementation of Neural Networks," in *Lecture Notes in Computer Science Volume 6593, 2011, Adaptive and Natural Computing Algorithms - 10th International Conference, ICANNGA 2011, Proceedings, Part I Conference, ICANNGA 2011*, 2011, pp. 158-168.

19. S. Skube, P. Bulić, and U. Lotrič, “Logarithmic Arithmetic for Low-Power Adaptive Control Systems,” *IEEE Transactions on Control Systems Technology, under review*, pp. 1-22, 2013.
20. M. Ito, D. Chinnery, and K. Keutzer, “Low power multiplication algorithm for switching activity reduction through operand decomposition,” in *Proceedings of the 21st International Conference on Computer Design, 2003.* , 2003, pp. 21 - 26.
21. E.E. Swartzlander and A.G. Alexopoulos, “The Sign/Logarithm Number System,” *IEEE Transactions on Computers*, vol. 24, no. 12, pp. 1238 - 1242, Dec. 1975.
22. F.J. Taylor, R. Gill, J. Joseph, and J. Radke, “A 20 bit logarithmic number system processor,” *IEEE Transactions on Computers*, vol. 37, no. 2, pp. 190 - 200, Aug. 1988.

Arrived: 18. 10. 2013
Accepted: 08. 11. 2013