Adaptable, fast, area-efficient architecture for logarithm approximation with arbitrary accuracy on FPGA

Dimitris Bariamis, Dimitris Maroulis*, Dimitris K. Iakovidis
Department of Informatics and Telecommunications, University of Athens, Panepistimiopolis, Ilisia, 15784 Athens, Greece

***Abstract* — This paper presents ALA (Adaptable Logarithm Approximation), a novel hardware architecture for the approximation of the base-2 logarithm of integers at an arbitrary accuracy, suitable for fast and area-efficient FPGA implementation. It is based on a piecewise linear approximation methodology, implemented so that an arbitrary number of linear segments approximate the logarithm function. The achieved approximation accuracy depends on the number of segments used, which also affects the size of a ROM used for storing the parameters that control the computation. The implementation of the ROM using an FPGA BlockRAM allows the parameters to be updated without reconfiguration of the FPGA core. This provides the considerable advantage of data set adaptability to the proposed architecture over the other relevant architectures, as the parameters can be easily updated to minimize the approximation error for different data sets. Both real and synthetic datasets have been used for evaluation purposes. The results show that ALA adapts well to all data sets used and requires significantly less FPGA slices than the CORDIC architecture to achieve the same or higher approximation accuracy. Moreover, it provides a throughput of one result per cycle and up to four times lower latency than the CORDIC core.**

***Keywords* — Field Programmable Gate Arrays, Digital Design**

I. Introduction

The utility of the logarithm in computer science spans a broad spectrum of application domains, including telecommunications, signal and image processing, industrial and biomedical. Many of these applications require a vast number of logarithmic operations per second. Usually, the logarithmic operations should not only be performed fast, but also be performed accurately. For example, in the biomedical domain, inaccurate computations can affect diagnostic processes. In [1,2] the logarithm is involved in the estimation of an entropy feature used for the characterization of the texture of the human colon from endoscopic video frames; in [3] the logarithm is involved in

* Corresponding author
The authors are with the Real Time Systems and Image Analysis Laboratory, Department of Informatics and Telecommunications, University of Athens, 15784 Panepistimiopolis, Athens, Greece; phone: +30-210-7275317; e-mail: {d.bariamis, dmaroulis, diakov, rtsimage}@di.uoa.gr

the estimation of the index of hemoglobin used for color enhancement of the endoscopic video frames; whereas in [4] logarithmic transformations are massively applied for the pre-processing of thousands of microarray data samples.

Hardware architectures that have been proposed for the approximation of the base-2 logarithm include implementations of power series methods [5], polynomial methods [6], high-radix algorithms [7], lookup tables [8], the CORDIC (Coordinate Rotation Digital Computer) algorithm [9] and linear approximation methods [10-14]. Most commonly, CORDIC has been the algorithm of choice for logarithm approximation in state of the art processing units [15,16]. It was originally proposed by Volder [9] for the approximation of trigonometric functions. It was later generalized [17] for the approximation of several other functions, such as the hyperbolic, the exponential, and the logarithm. CORDIC is an iterative algorithm that uses shift, add and table lookup operations; therefore it is suitable for devices that lack multipliers, such as some FPGAs. In applications where resource utilization is more important than throughput, a word-serial CORDIC implementation is used. For throughput sensitive applications, a pipelined implementation of CORDIC that achieves a throughput of one result per clock cycle is usually preferred [18].

Alternatively, many hardware architectures aiming at crude, however fast approximation of the logarithm, implement the piecewise linear approximation approach based on Mitchell's method [10]. According to this approach the logarithm function is approximated by a small and fixed number of consecutive linear segments. In [1],[2] and in [11] this method was implemented by using only two segments; in [12] and [13] four segments were used; whereas in [14] a VLSI architecture that uses two, three and six segments was proposed.

Inspired by Mitchell's method, and motivated by the need of combining both computational efficiency and approximation accuracy, we propose ALA (Adaptable Logarithm Approximation), a novel architecture for the approximation of the base-2 logarithm in a fast and area-efficient way. ALA provides the versatility of using an arbitrary number of segments, rather than a small, fixed number of segments used in the state of the art architectures. The achieved approximation accuracy depends on the number of segments used, which also affects the size of a ROM that is used for storing the parameters that control the computation. These parameters can be adapted to any data set, further enhancing the achieved accuracy. The implementation of the ROM using an FPGA BlockRAM allows the parameters to be updated without reconfiguration of the FPGA core, thus providing the advantage of data set adaptability. In short, the ALA architecture provides the following novel features compared to other state of the art architectures:

- Efficient FPGA implementation that allows a large number of linear segments for the approximation of the logarithm, leading to a high accuracy while having low FPGA resource requirements

- Adaptability to the input data set without requiring reconfiguration of the FPGA, further enhancing the approximation accuracy
- Exploitation of the available multipliers and BlockRAMs of the FPGA that can significantly reduce the slice requirements

The rest of this paper is organized in four sections. Section II describes the logarithm approximation approach implemented in the proposed architecture. The architecture itself is described in Section III, and the results of the comprehensive experiments conducted are presented in Section IV. The conclusions of this study are summarized in Section V.

II. Logarithm Estimation via Piecewise Linear Approximation

In order to be able to control the approximation accuracy of the base-2 logarithm of integers, we considered a piecewise linear approximation approach that involves three steps: 1. calculation of the integral part of the logarithm, 2. linear approximation of the fractional part of the logarithm, and 3. piecewise linear approximation of the fractional part by splitting the linear approximation produced in the second step into *s* consecutive linear segments. The steps are described in detail below:

A. Step 1

The integral part of the logarithm, $l_i(x) = \lfloor \log_2(x) \rfloor$, is determined by the position of the Most Significant Bit (MSB) of the input number *x*, where

$$2^n \le x < 2^{n+1} \Rightarrow l_i(x) = n,\ n \ge 0 \quad (1)$$

B. Step 2

The fractional part of the logarithm i.e. $l_f = \log_2(x) - l_i(x)$, is estimated by linear approximation between the points $(2^n, n)$ and $(2^{n+1}, n+1)$, *n*=1,2,…, of the function $\log_2(x)$ plotted in Fig. 1. The approximated fractional part is

$$l_f(x) = \frac{x - 2^{l_i(x)}}{2^{l_i(x)+1} - 2^{l_i(x)}} = \frac{x}{2^{l_i(x)}} - 1 \quad (2)$$

From Eq. 2 it is obvious that $0 \le l_f(x) < 1$ for every integer *x*.

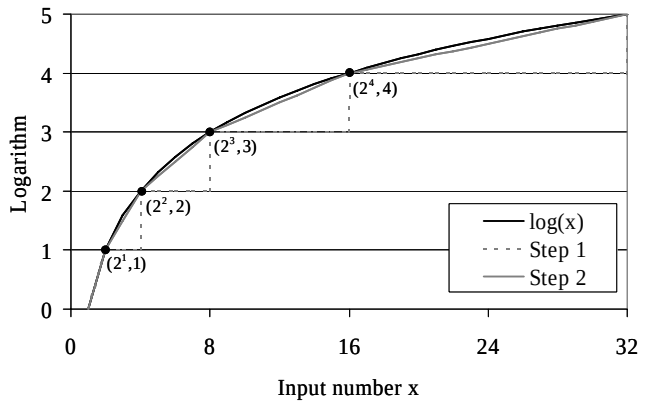


Fig. 1. Steps 1 and 2 of logarithm approximation

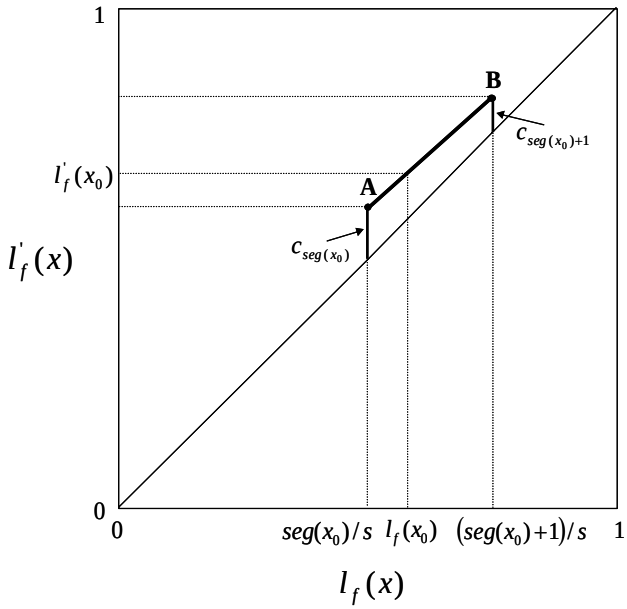


Fig. 2. l_f' as a function of l_f

C. Step 3

In the third step, a new approximation $l'_f(x)$ of the fractional part of the logarithm is derived as a function of $l_f(x)$. It is derived by piecewise linear approximation between the points $(2^n, n)$ and $(2^{n+1}, n+1)$ using *s* linear segments of equal length.

The segment *seg*(*x*) to which *x* belongs is determined by its fractional part $l_f(x)$ according to Eq. 3.

$$seg(x) = \lfloor s \cdot l_f(x) \rfloor \qquad (3)$$

Figure 2 illustrates a schematic representation of the $l_f(x)$ to $l'_f(x)$ transformation for an input number x_0. The endpoints *A* and *B* of the segment $seg(x_0)$ are elevated by $c_{seg(x_0)}$ and $c_{seg(x_0)+1}$ and their coordinates are derived by Eqs. 4 and 5 respectively. The linear segment *AB* is defined by Eq. 6, which is simplified to the equivalent Eq. 7.

$$A \equiv \left(\frac{seg(x_0)}{s}, \frac{seg(x_0)}{s} + c_{seg(x_0)} \right) \qquad (4)$$

$$B \equiv \left(\frac{seg(x_0)+1}{s}, \frac{seg(x_0)+1}{s} + c_{seg(x_0)+1} \right) \quad (5)$$

$$l_f^{'}(x_0) - \left(\frac{seg(x_0)}{s} + c_{seg(x_0)} \right) = \frac{\left(\frac{seg(x_0)+1}{s} + c_{seg(x_0)+1} \right) - \left(\frac{seg(x_0)}{s} + c_{seg(x_0)} \right)}{\frac{seg(x_0)+1}{s} - \frac{seg(x_0)}{s}} \cdot \left(l_f(x_0) - \frac{seg(x_0)}{s} \right) \quad (6)$$

$$l_f^{'}(x) = l_f(x) + \left(c_{seg(x)+1} - c_{seg(x)} \right) \cdot \left(s \cdot l_f(x) - seg(x) \right) + c_{seg(x)} \quad (7)$$

Figures 3 and 4 illustrate two approximation examples of the logarithm function for an input number x_0, using two and four segments (s=2 and s=4) respectively. It can be noticed that in the case of two segments, the articulation point is elevated by c_1, and in the case of four segments, the three articulation points are elevated by c_1, c_2 and c_3.

The optimal parameters c_i can be determined upon the data set used in a particular application by minimizing the approximation error E (Eq. 8), which represents the weighted sum of the relative differences between the actual and approximated values of the logarithm. The weight $p(x)$ used for the calculation is the Probability Mass Function (PMF) of the data set.

$$E = \sum_x p(x) \cdot \left| \frac{\left(l_i(x) + l_f'(x) \right) - \log_2(x)}{\log_2(x)} \right| \quad (8)$$

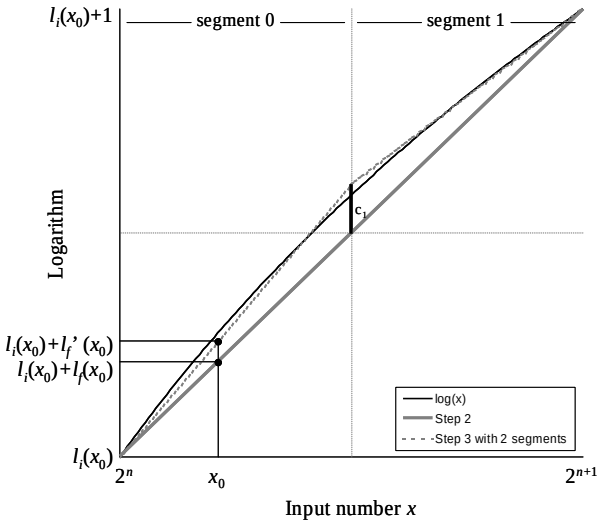


Fig. 3. Step 3 of logarithm approximation with 2 segments

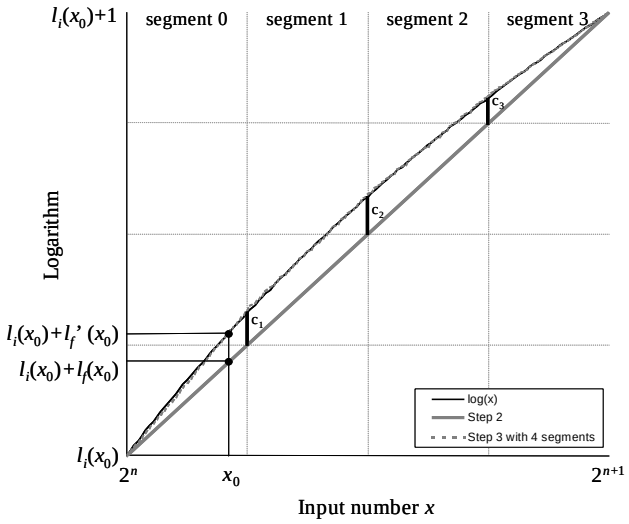


Fig. 4. Step 3 of logarithm approximation with 4 segments

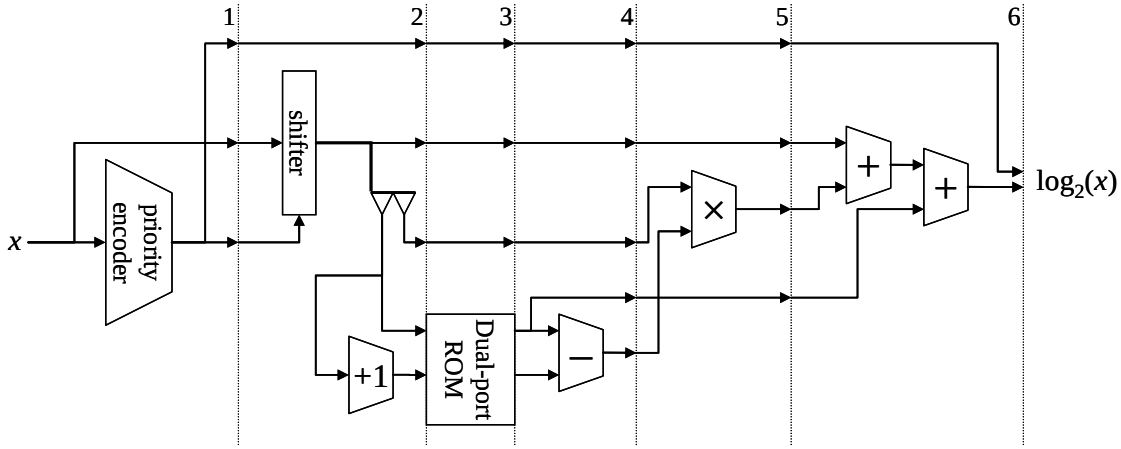


Fig. 5. The ALA architecture

III. Adaptable Logarithm Approximation Architecture

The ALA architecture is implemented as a fully pipelined circuit, in order to achieve a throughput of one result per clock cycle. It consists of 6 pipeline stages and one dual-ported ROM. The ROM stores the parameters c_i in a b-bit wide fixed point representation and can be implemented on FPGA slices or BlockRAMs, depending on which of these resources is more valuable for the specific design. In the proposed implementation, the ROM resides on a single dual-ported BlockRAM. Even though a change in the number of segments s or the parameter width b would

require reprogramming of the FPGA core, the parameters c_i can be updated without reprogramming by transferring them from the host PC to the BlockRAM.

Figure 5 illustrates the structure of the pipelined circuit. In the first pipeline stage, a priority encoder is used to locate the MSB of the input number *x*. The output of the priority encoder is the integral part $l_i(x)$ of $\log_2(x)$. In the second stage a shifter is used to isolate the fractional part $l_f(x)$.

For the calculation of $l_f^{'}(x)$, we have regarded *s* as a power of two. Thus, the product $s \cdot l_f(x)$ (Eq. 3) is equal to $l_f(x)$ shifted left by $\log_2(s)$ bits and the segment *seg*(*x*) is given by $\lfloor s \cdot l_f(x) \rfloor$, which represents the $\log_2(s)$ bits on the left of the radix point of $s \cdot l_f(x)$. The quantity $s \cdot l_f(x) - seg(x) = s \cdot l_f(x) - \lfloor s \cdot l_f(x) \rfloor$ is equal to the fractional part of $s \cdot l_f(x)$, represented by the bits of $s \cdot l_f(x)$ that lie on the right of the radix point. Therefore, the quantities *seg*(*x*) and $s \cdot l_f(x) - seg(x)$, needed for the calculation of Eq. 7, can be calculated from the fixed-point representation of $l_f(x)$ by separating its bits into two parts, without performing any arithmetic or logic operations.

In order to proceed with the calculation of $l_f^{'}(x)$, *seg*(*x*) and $s \cdot l_f(x) - seg(x)$ are calculated at the end of the second stage. The third stage involves two concurrent ROM lookups at addresses *seg*(*x*) and *seg*(*x*)+1, retrieving the parameters $c_{seg(x)}$ and $c_{seg(x)+1}$. In the fourth stage, $c_{seg(x)}$ is subtracted from $c_{seg(x)+1}$ and in the fifth stage the result is multiplied by $s \cdot l_f - seg(x)$ using one of the on-chip multipliers. In the last stage, the result of the fifth stage is added to the sum of $l_f(x)$ and $c_{seg(x)}$ to produce $l_f^{'}(x)$, which is concatenated to $l_i(x)$ in order to produce the final $\log_2(x)$ approximation.

IV. Results

Experiments were conducted to evaluate the performance of the ALA architecture with different datasets. The results are accompanied with comparisons with state of the art architectures implementing piecewise linear approaches to logarithm approximation, as well as with the commonly used CORDIC approach.

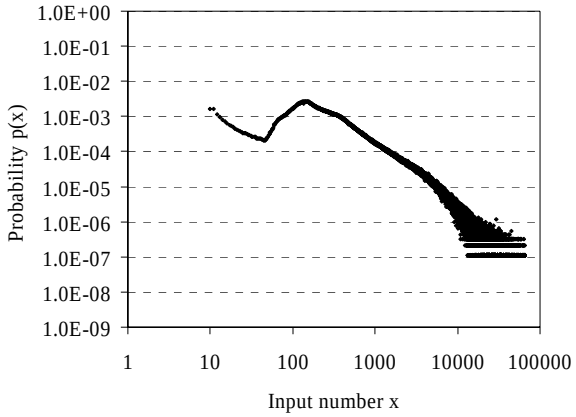
A. Input data sets

There are several applications where the PMF of the input number *x* is not uniform, such as the analysis of medical or biological data. We have considered two data sets, namely Data Set 1 that comprises of microarray data [19] and Data Set 2 that comprises of colonoscopy video streams [2]. The processing task for Data Set 1 (Fig 6a), is the logarithmic normalization of the intensities of more than nine million microarray spots. For Data Set 2 (Fig 6b), the

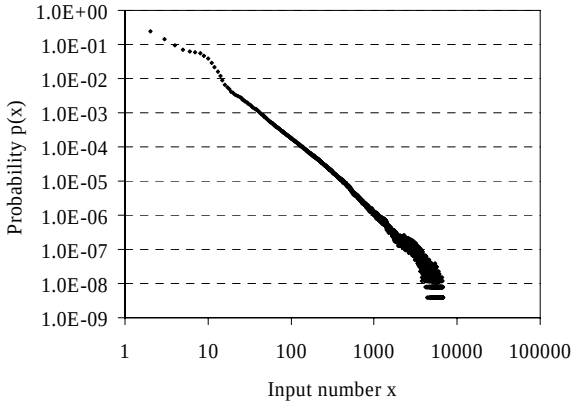
processing task is the estimation of the entropy of grey level co-occurrence matrices of video frame blocks. This task involves more than one billion logarithm calculations.

We have also considered a synthetic data set based on the Gauss-Kuzmin (Fig. 6c) distribution and nine synthetic data sets based on normal distributions, three of which are shown in Fig. 6d. The means and standard deviations of the normal distributions used are: μ=1024 with σ=256, 512, 1024, μ=4096 with σ=1024, 2048, 4096, and μ=16384 with σ=4096, 8192, 16384 respectively.

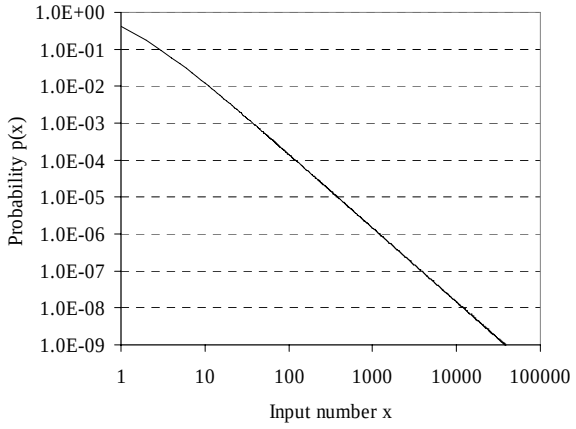
Using the PMFs illustrated in Fig. 6, we calculated an optimal set of the parameters c_i that minimizes the weighted relative approximation error E (Eq. 8) for each data set, by means of a software utility. The c_i parameters were then loaded into the ROM of the ALA architecture.



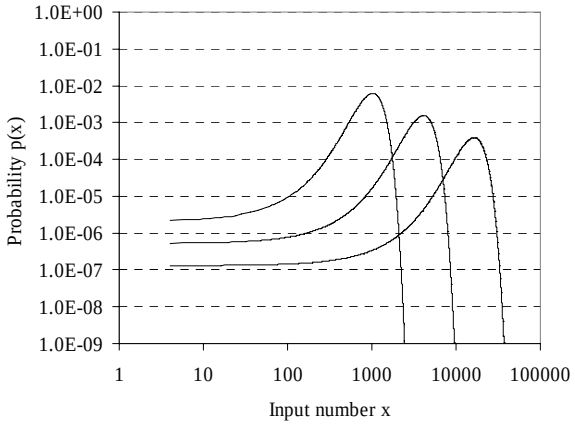
(a) Data Set 1



(b) Data Set 2



(c) Gauss-Kuzmin Distribution



(d) Normal Distributions

Fig. 6. Probability Mass Functions of tested data sets

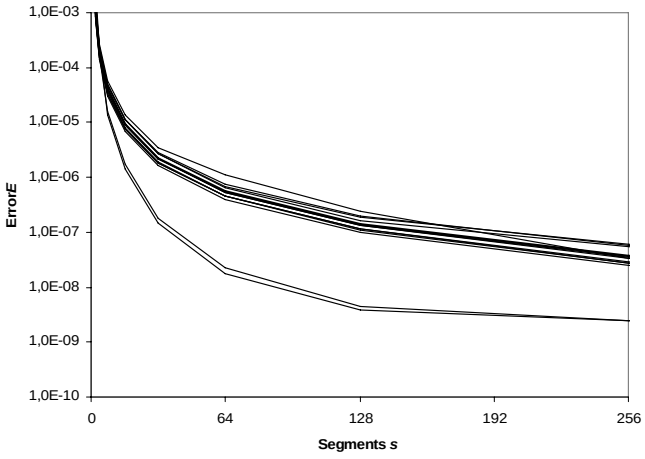


Fig. 7. Approximation error as a function of segments *s*, for *b* = 24

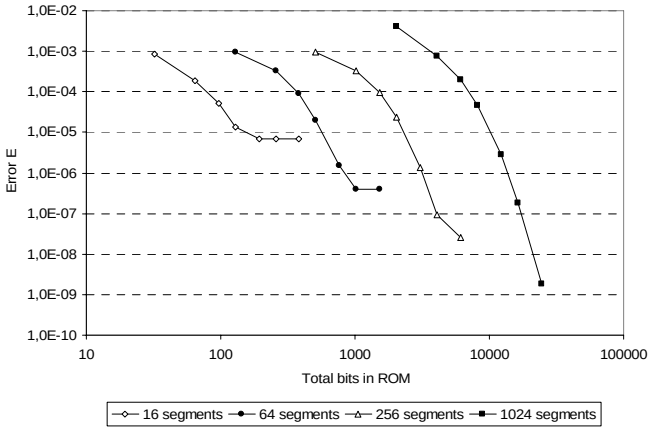


Fig. 8. Approximation error as a function of ROM size for the uniform PMF data set

Figure 7 illustrates the approximation error *E* obtained for each data set, as a function of the number of segments *s*, for *b* = 24. Data Set 1 as well as the normal and uniform data sets form the upper cluster of lines in Fig. 7. The approximation error for Data Set 2 and the Gauss-Kuzmin data set is almost identical to each other and it is up to two orders of magnitude smaller compared to the approximation error obtained by the other data sets.

B. Implemented configurations

The ALA architecture was implemented for the approximation of the base-2 logarithm of 16-bit integers on a Xilinx Virtex-5 FPGA (XC5VLX110T-3) and was tested for several different configurations. The number of linear segments *s* and the width *b* of the parameters c_i are adjusted so that the ROM resides on a single 36kbit BlockRAM and the multiplier is implemented on a single 25×18 bits multiplier of a DSP48E block. Therefore *s* ranges from 2 to 1024, whereas *b* ranges from 2 to 24 bits. The *b*-bit wide fixed point representation of the parameters c_i consists of a mantissa of *b* bits, whereas the exponent *e* is the same for all values of c_i for each configuration and not stored in the ROM. The value *e* of the exponent varies in the range of -4 to -12, depending on the bit width *b*.

The error *E* obtained as a function of the size of the ROM is illustrated in Fig. 8, for the data set with the uniform PMF. The figure shows four distinct configurations with *s* = 16, 64, 256 and 1024. For each one of these configurations, the bit width of the c_i parameters is 2, 4, 6, 8, 12, 16 or 24.

C. Comparison to other architectures implementing piecewise linear approximation approaches

The piecewise linear approximation approaches implemented by the architectures proposed in [11-14] aim to provide a fast approximation of the logarithm that requires few hardware resources. Due to their design considerations, they employ hardwired logic in order to increase the accuracy of [10], resulting in loss of versatility. The ALA architecture yields considerable gains in accuracy and versatility by using the available FPGA resources efficiently, instead of relying on the hardwired error-correcting logic of the aforementioned methods. Emphasis is given to minimizing the slice requirements, therefore the ROM is implemented on a single dual ported BlockRAM and the multiplier on one dedicated FPGA multiplier block. The accuracy of ALA can be increased by using more linear segments for the approximation and by adapting the parameters of approximation to the input data set. The versatility of ALA relies on the capability of the architecture to load updated approximation parameters into the ROM at runtime without altering the implemented logic.

TABLE I
Approximation error percentage range and error *E* of the proposed architecture when adapted to Data Set 1 and Data Set 2

Method	Segments	Error Percentage Range (%)	Error *E*
ALA adapted to Data Set 1 using Data Set 1	2	2.2975	1.203E-03
	4	0.4835	2.601E-04
	8	0.1025	5.756E-05
	16	0.0219	1.344E-05
	32	0.0051	3.507E-06
	64	1.26E-03	1.090E-06
	128	1.51E-04	2.380E-07
	256	3.04E-05	3.409E-08
	512	6.87E-06	5.189E-09
	1024	1.56E-06	1.837E-09
ALA adapted to Data Set 2 using Data Set 2	2	1.2682	2.060E-03
	4	0.2826	2.323E-04
	8	0.0611	1.371E-05
	16	0.0131	1.431E-06
	32	0.0028	1.501E-07
	64	6.18E-04	1.802E-08
	128	1.36E-04	3.813E-09
	256	3.04E-05	2.424E-09
	512	6.87E-06	2.307E-09
	1024	1.56E-06	2.298E-09

TABLE II
Approximation error percentage range and error *E* of architectures implementing piecewise linear approximation approaches, for data with a uniform PMF, Data Set 1 and Data Set 2. The ALA architecture is adapted to the uniform PMF

Method	Segments	Error Percentage Range (%)	Error *E*		
			Uniform PMF	Data Set 1	Data Set 2
Mitchell et al. [10]	1	5.3605	3.992E-03	6.825E-03	1.645E-02
SanGregory et al [11]	2	1.9717	1.888E-03	3.411E-03	5.894E-03
Combet et al. [12]	4	2.4399	1.106E-03	1.974E-03	2.849E-03
Hall et al. [13]	4	0.9071	1.650E-04	2.854E-04	3.383E-04
Abed et al. [14]	2	1.4843	1.106E-03	1.986E-03	2.469E-03
	3	0.6998	6.748E-04	1.213E-03	1.638E-03
	6	0.3067	2.277E-04	3.950E-04	3.336E-04
ALA adapted to uniform PMF	2	2.2407	6.539E-04	1.205E-03	4.013E-03
	4	0.4750	1.370E-04	2.603E-04	8.055E-04
	8	0.1020	2.970E-05	5.761E-05	2.161E-04
	16	0.0220	6.826E-06	1.346E-05	5.485E-05
	32	0.0052	1.634E-06	3.509E-06	1.389E-05
	64	0.0012	4.011E-07	1.124E-06	3.454E-06
	128	2.94E-04	1.001E-07	3.717E-07	8.455E-07
	256	6.84E-05	2.530E-08	1.089E-07	1.977E-07
	512	1.50E-05	6.551E-09	2.868E-08	4.150E-08
	1024	2.89E-06	1.829E-09	6.027E-09	4.115E-09

Table I shows the error percentage range and the error E of ALA when adapted to each data set. The error percentage range is defined as the difference of the maximum relative error to the minimum relative error produced for each experiment. If no prior knowledge about the input data is available, ALA can be adapted to the case of the uniform PMF. Table II includes ALA when adapted to data with a uniform PMF, showing the error percentage range and the error *E* of all piecewise linear approximation methods under the assumption of no prior knowledge. When using four segments, the ALA architecture obtains a smaller error range than any other four segment approach for data of uniform PMF or Data Set 1, moreover, when using 8 or more segments, ALA significantly outperforms the other methods for all data sets evaluated. It is worth noting that all approaches display a higher approximation error for Data Set 1 and Data Set 2 compared to the case of the uniform PMF. However, the comparison of the ALA results in Table I and Table II reveal that by adapting to each data set, ALA manages to reduce the approximation error by up to two orders of magnitude. The greatest benefit of data set adaptation can be observed for Data Set 2 when s=128, where the approximation error of ALA adapted to Data Set 2 is more than 200 times smaller than that of ALA adapted to the uniform PMF. Furthermore, the results shown in Table II illustrate that ALA is an architecture of generic use, as it achieves a very low approximation error for input data with an unknown PMF.

D. Comparison to CORDIC and other architectures

In order to evaluate the performance of the ALA architecture, we implemented a CORDIC core for the calculation of the hyperbolic arctangent (arctanh) function using the Xilinx Core Generator. The base-2 logarithm function can be calculated through the arctanh function by using Eq. 9.

$$\log_2(x) = \frac{\ln x}{\ln 2} = \frac{2}{\ln 2} \cdot \arctan h(\frac{x-1}{x+1}) \quad for\ -1 < x < 1 \tag{9}$$

The input number *x* must be scaled to $0 \le x < 1$ before it is fed to the CORDIC core and the output of the CORDIC core must be multiplied by the $2/\ln 2$ constant, therefore additional circuitry has been implemented. The additional circuitry requires nearly 100 slices and can achieve a maximum frequency higher than that of the CORDIC core, hence it does not create a bottleneck. Table III shows the most characteristic of the implemented configurations of the proposed architecture and the CORDIC core, sorted by the obtained approximation error when used on the uniform PMF data set. The slices for the CORDIC core do not include the additional circuitry. Table III also includes the implementation results for the simple linear approach proposed by Hall et. al. [13], which has a low slice utilization but can only achieve an approximation error in the order of 10^{-4}.

TABLE III
Results for uniform PMF using ALA and CORDIC architectures

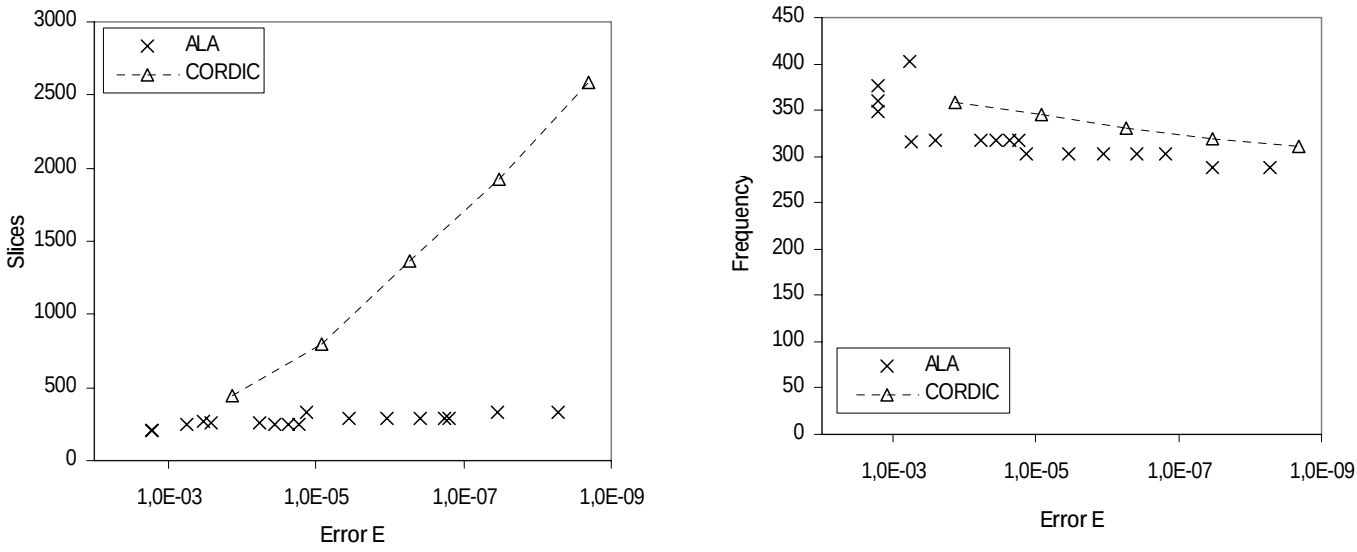
	E	*s*	*b*	Slices	Frequency	Latency	
					MHz	cycles	ns
Hall et. al. [18]	1.65E-04	4	-	202	322.79	4	12.39
ALA	3.55E-04	4	2	249	401.78	6	14.93
	1.37E-04	4	8	254	318.02	6	18.87
	3.02E-05	8	8	254	318.02	6	18.87
	2.03E-05	64	8	241	318.02	6	18.87
	6.83E-06	16	16	295	302.57	6	19.83
	1.63E-06	32	16	290	302.57	6	19.83
	4.08E-07	64	16	285	302.57	6	19.83
	1.31E-07	128	16	291	302.57	6	19.83
	8.95E-08	512	16	293	302.57	6	19.83
	6.55E-09	512	24	333	288.27	6	20.81
	1.83E-09	1024	24	329	288.27	6	20.81
CORDIC	1.36E-04	-	-	445	358.42	12	33.48
	8.47E-06	-	-	793	344.83	16	46.40
	5.35E-07	-	-	1363	329.83	20	60.64
	3.33E-08	-	-	1927	319.12	24	75.21
	2.07E-09	-	-	2587	310.89	28	90.06

As shown in Table III, ALA has a latency of 6 clock cycles for all configurations, which is smaller than the minimum latency of CORDIC, which depends on the desired accuracy. Due to the significant difference in pipeline stages, the latency of ALA ranges from 14.93 ns to 20.81 ns, whereas the latency of the CORDIC core for the same approximation error is up to four times higher, ranging from 33.48 ns to 90.06 ns. The throughput of both architectures is one result per clock cycle, but ALA is able to provide advantageous performance when used in latency-sensitive applications.

The proposed architecture can be implemented in significantly fewer slices than the CORDIC core for the same approximation error. The difference in required slices is illustrated in Fig. 9a. The pipelined CORDIC core requires additional and wider pipeline stages in order to decrease the approximation error, therefore the number of required slices for the CORDIC core increases more than linearly for each increase of accuracy. In contrast to the CORDIC core, ALA does not require a significant increase in slices in order to decrease the approximation error, as the resulting error mostly depends on the ROM size. In order to minimize the usage of FPGA resources, all the implemented configurations require only one BlockRAM for the storage of the ROM. This results in 329 slices for the proposed architecture when $E = 1.83 \cdot 10^{-9}$, compared to 2587 slices for the CORDIC core for a slightly higher error of $E = 2.07 \cdot 10^{-9}$. The maximum operating frequency of ALA is 401.78 MHz, while the CORDIC core achieves a maximum frequency of 358.42 MHz. In the case of the uniform PMF data set and the normal distributions, the results resemble closely those achieved when the architecture is adapted to Data Set 1.

When using Data Set 2 or a Gauss-Kuzmin distribution, ALA provides up to two orders of magnitude lower approximation error than the uniform and normal distributions. Fig 10a illustrates the required slices as a function of the error *E* while Fig. 10b illustrates the achieved frequency. It is evident from Fig. 10 that the adaptation to Data Set 2 or other data sets that resemble a Gauss-Kuzmin distribution results in a smaller achieved approximation error than the other distributions and enhances the slice utilization advantage of the ALA architecture over the CORDIC core.

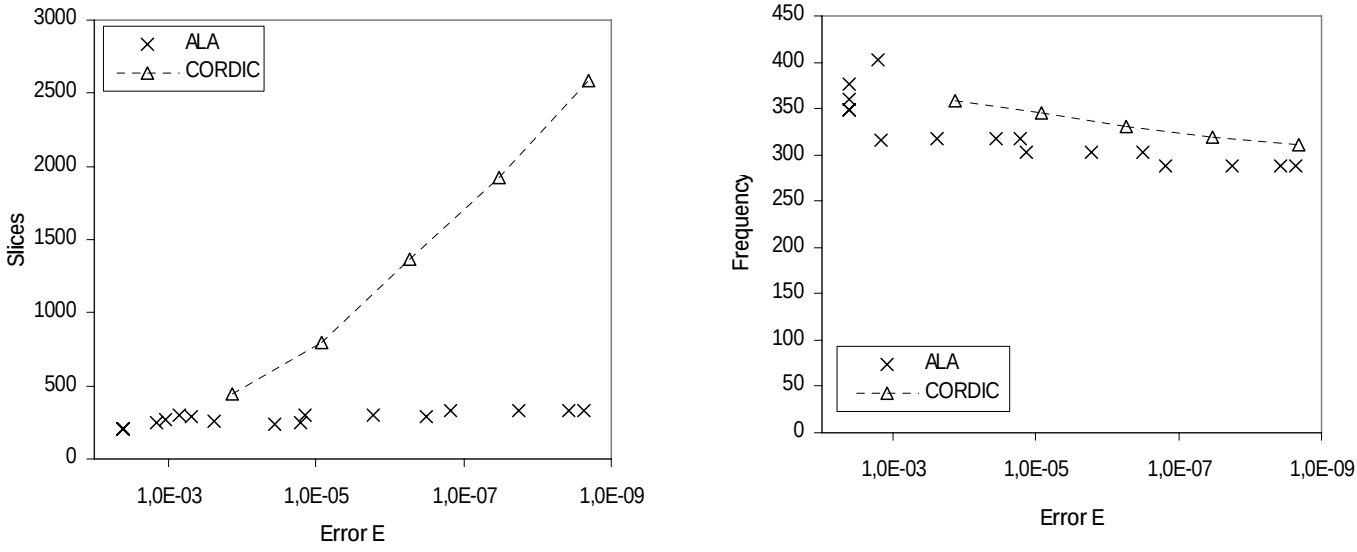
The floating point logarithm approximation architecture presented in [6] has also been synthesized on the same FPGA. It achieves a lowest approximation error in the order of $E=10^{-8}$ for the case of a 23-bit mantissa, which is higher than that of the proposed architecture, while requiring an almost equal number of slices and 6 times more multipliers.



(a) Slices

(b) Frequency

Fig. 9. Comparison between ALA and CORDIC architectures, using Data Set 1



(a) Slices

(b) Frequency

Fig. 10. Comparison between ALA and CORDIC architectures, using Data Set 2

V. Conclusions

We presented the ALA architecture for fast and area-efficient approximation of the base-2 logarithm on FPGA devices. The novel features of this architecture can be summarized in the following:

- It implements a piecewise linear approximation of the logarithm function using an arbitrary number of linear segments, in order to achieve high accuracy.
- It provides adaptability to different data sets without even requiring reconfiguration of the FPGA core.
- Its implementation is designed to exploit the available FPGA resources, such as BlockRAMs and multipliers.

The first feature makes ALA an architecture of generic use. It can be used in a variety of FPGA designs as a generic component that can be easily implemented using a different number of linear segments in order to approximate the logarithm. The second feature makes ALA suitable for applications operating with datasets of different probability mass functions, especially if timely critical alterations of the datasets are involved. For example, the application of different signal transformation processes on an input signal, can lead to different output signals with different probability mass functions. So, the logarithmic processing of these output signals in real-time can be a potential application of the ALA architecture. The third feature enables a significant reduction in FPGA slices by using one BlockRAM for the implementation of the dual-ported ROM and one FPGA multiplier, also allowing ALA to reach a high frequency potential.

Other logarithm approximation architectures implementing piecewise linear approximation [11-14] use up to a maximum of 6 segments, whereas they cannot be implemented using a larger number of segments, or adapted to the dataset of each particular application. Compared with the CORDIC architecture, which is used by state of the art processing systems for logarithm approximation, the ALA architecture requires significantly less FPGA area to achieve the same or higher precision. Moreover, it operates at a similar frequency to the CORDIC core, while providing a throughput of one result per cycle and up to four times lower calculation latency. Thus, the ALA architecture can be embedded into any FPGA-based application that requires fast and accurate logarithm approximation.

Acknowledgements

This work was realized under the framework of the Reinforcement Program of Human Research Manpower (“PENED 2003” – 03ED324), co-funded by the General Secretariat for Research and Technology, Greece, and the European Social Fund.

REFERENCES

[1] D. Bariamis, D.K. Iakovidis, D. Maroulis, “Dedicated hardware for real-time computation of second-order statistical features for high resolution images”, Lecture Notes in Computer Science, Volume 4179 LNCS, Pages 67-77, 2006

[2] S.A. Karkanis, D.K. Iakovidis, D.E. Maroulis, D.A. Karras, and M. Tzivras, “Computer Aided Tumor Detection in Endoscopic Video using Color Wavelet Features,” IEEE Transactions on Information Technology in Biomedicine, vol. 7, pp. 141-152, 2003

[3] K. Nakamura, “Development of real-time endoscopic image processing technology: Adaptive index of hemoglobin color enhancement processing”, Digestive Endoscopy, 14 (Suppl.), S40–S47, 2002

[4] D.M. Rocke and B. Durbin, “Approximate variance-stabilizing transformations for gene-expression microarray data”, Bioinformatics, Vol. 19 no. 8, pages 966–972, 2003

[5] D. M. Mandelbaum, and S. G. Mandelbaum, “A Fast, Efficient Parallel-Acting Method of Generating Functions Defined by Power Series, Including Logarithm, Exponential, and Sine, Cosine,” IEEE Trans. on Parallel and Distributed Systems, vol. 7, no. 1, pp. 33-45, Jan. 1996.

[6] J. Detrey, F. de Dinechin, “Parametrized floating-point logarithm and exponential functions for FPGA”, Microprocessors and Microsystems, Article in Press.

[7] J.-A. Pineiro, M.D. Ercegovac, J. D. Bruguerax, “High–Radix Logarithm with Selection by Rounding”, Proceedings of the IEEE International Conference on Application-Specific Systems, Architectures, and Processors, ASAP02, pp. 101-110, July 2002

[8] J. A. Starzyk, and Y. Guo, "An Entropy-based Learning Hardware Organization Using FPGA," in Proc. Southeastern Symposium on System Theory, pp. 1-5, Athens, OH, 2001.

[9] J. E. Volder, “The CORDIC Trigonometric Computing Technique”, IRE Transactions on Electronic Computers, EC-8:330-334, 1959.

[10] J.N. Mitchell Jr., “Computer Multiplication and Division Using Binary Logarithms,” IRE Trans. Electronic Computers, vol. 11, pp. 512-517, Aug. 1962.

[11] S.L. SanGregory, R.E. Siferd, C. Brother, and D. Gallagher, “A Fast, Low-Power Logarithm Approximation with CMOS VLSI Implementation,” Proc. IEEE Midwest Symp. Circuits and Systems, Aug. 1999.

[12] M. Combet, H. Zonneveld, and L. Verbeek, “Computation of the Base Two Logarithm of Binary Numbers,” IEEE Trans. Electronic Computers, vol. 14, pp. 863-867, Dec. 1965.

[13] E.L. Hall, D.D. Lynch, and S.J. Dwyer III, “Generation of Products and Quotients Using Approximate Binary Logarithms for Digital Filtering Applications,” IEEE Trans. Computers, vol. 19, pp. 97-105, Feb. 1970.

[14] K. H. Abed and R. E. Siferd “CMOS VLSI Implementation of a Low-Power Logarithmic Converter,” IEEE Transactions on Computers, vol. 52, pp. 1421-1433, Nov. 2003

[15] T. Lang, E. Antelo, “High-throughput CORDIC-based geometry operations for 3D computer graphics”, IEEE Transactions on Computers, 54 (3), pp. 347-361, 2005

[16] D.E. Metafas and C.E. Goutis, “A floating-point advanced cordic processor” Journal of VLSI Signal Processing, 10 (1), pp. 53-65, 1995.

[17] J. S. Walther, “A unified Algorithm for Elementary Functions,” in Proceedings of the 38th Spring Joint Computer Conference, pp. 379–385, 1971.

[18] Xilinx LogiCORE CORDIC v3.0 Product Specification DS249, Xilinx Inc., 2005

[19] Stanford MicroArray Database, http://smd.stanford.edu/