Low Cost Hardware Implementation of Logarithm Approximation

R. Gutierrez and J. Valls

***Abstract*—A low cost, high-speed architecture for the computation of the binary logarithm is proposed. It is based on the Mitchell approximation with two correction stages: a piecewise linear interpolation with power-of-two slopes and truncated mantissa, and a LUT-based correction stage that correct the piecewise interpolation error. The architecture has been implemented in an FPGA device and the results are compared with other low cost architectures requiring less area and achieving high-speed.**

***Index Terms*—Logarithm approximation, Mitchell's error correction, piecewise linear approximation.**

I. Introduction

Real-time digital signal processing applications like digital communication systems require high-speed, low-power logarithm operators with low or moderate accuracy. Many methods have been proposed to calculate the logarithm with high accuracy, i.e., LUT-based [1], polynomial approximation [2], digit-recurrence methods [3]. These methods require complex hardware or have long latencies, so they are not suitable for high-speed, low-power applications. The simplest method to approximate the logarithm was proposed by Mitchell [4]. He divides the logarithm into intervals of powers-of-two integers and uses the approximation $\log(1 + \mathrm{x}) \approx \mathrm{x}$ for each interval. The drawback of the Mitchell method is its poor maximum approximation error of 0.08639, which is only 3.53 bits of accuracy. Several methods were developed to improve the accuracy of Mitchell's approximation by adding error correction techniques. These techniques are implemented using piecewise linear interpolation or LUT-based methods.

The most efficient error correction techniques in terms of area cost are the ones based on piecewise linear interpolation [5]–[9], specifically those that only use the operation shift-and-sum for the computation [5]–[9]. [5] and [6] divide the interval into four subintervals and select the coefficients to minimize the absolute error or the mean square error, respectively; Sangregory *et al.* [7] divide the power-of-two interval into two regions and, in order to reduce the hardware complexity, it only uses the mantissa's four more significant bits (MSB) in the interpolation; Abed *et al.* [8] divided the interval into two, three and six regions, and their method also use a truncated mantissa, and select the slopes of the straight-lines as power of two to reduce the hardware cost of the interpolation. The six region approach achieves the best accuracy of 6.2 bits with lower hardware complexity. Juang *et al.* [9] divides the exact logarithmic curve into two symmetric regions obtaining an error of 0.045.

On the other hand, methods based on LUT were also developed to improve the accuracy of Mitchell's approximation. In [10] it was proposed storing the error that occurs due to Mitchell's approximation in a table and adding it to the Mitchell approximation. To improve the accuracy while keeping the hardware cost low, in [11] Mitchell's error is stored in a LUT, whose values are interpolated, and added to its approximation.

In this paper the Mitchell's error is approximated by a four-region piecewise linear interpolation with power-of-two slopes to avoid multiplications. A new set of coefficients and slopes are proposed that improve the accuracy with respect to the previous linear interpolation techniques keeping low area. Furthermore, to achieve a higher accuracy, a small LUT that stores the residual error is added.

This paper is organized as follows: Section II summarizes the piecewise linear interpolation and LUT-based methods to correct Mitchell's approximation of the logarithmic function. In Section III the proposed method is explained. Section IV depicts the architecture for computing the proposed method. Section V shows the implementation results and the comparisons with other methods. Finally, the conclusions are exposed in Section VI.

Manuscript received March 25, 2010; revised July 05, 2010; accepted September 14, 2010. Date of publication October 25, 2010; date of current version October 28, 2011. This research was supported by FEDER, the Spanish Ministerio de Ciencia e Innovación, under Grant No. TEC2008-06787.

R. Gutierrez is with the Departamento de Física y Arquitectura de Computadores, Universidad Miguel Hernandez, Alicante 03202, Spain (e-mail: roberto.gutierrez@umh.es).

J. Valls is with Departamento de Ingeniería electronica, Universidad Politécnica de Valencia, 46730 Valencia, Spain (e-mail: jvalls@eln.upv.es).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TVLSI.2010.2081387

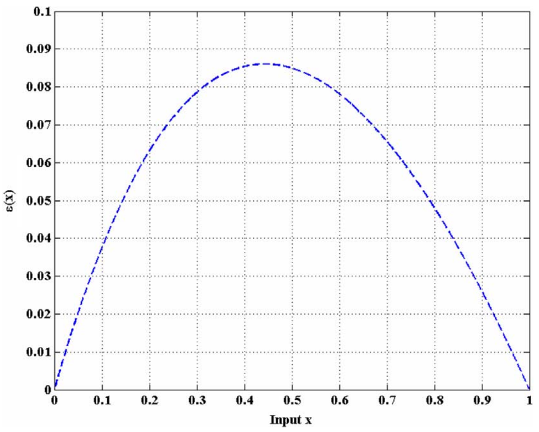

Fig. 1. Mitchell error.

II. Background

Let N be a binary number $z_n \; z_{n-1} \; z_{n-2} \ldots\ldots z_0 . z_{-1} z_{-2} \ldots\ldots\ldots z_{-p}$, and let z_n be the most significant nonzero bit of N. In such a case, N can be expressed as

$$N = 2^n(1 + x) \tag{1}$$

where x complies with $0 \leq \mathrm{x} < 1$. Then the base-2 logarithm of N is:

$$\log_2 N = n + \log_2(1 + x). \tag{2}$$

Therefore, logarithmic value of N can obtained by detecting the position of the most significant nonzero bit of N, and computing the approximate value of $\log_2(1 + x)$.

Mitchell suggested in [4] a linear approximation for $\log_2(1 + x)$, which only uses the linear term in the Taylor series: $\log_2(1 + x) \approx x$. This method is fast and extremely easy to implement in hardware but gives the following absolute error function

$$\varepsilon(x) = \log_2(1 + x) - x \tag{3}$$

whose maximum value is as high as 0.08639, which is only 3.5 bits of accuracy. This error is represented in Fig. 1.

TABLE I
CONSTANTS USED IN TWO REGION APPROXIMATION

	[7]	[8]	[9]
S_0	2^{-2}	2^{-2}	$(2^{-3}+2^{-4})$
S_1	2^{-2}	2^{-2}	$(2^{-3}+2^{-5})$
x_0'	$x_{3:0}$	$x_{2:0}$	$x_{3:0}$
x_1'	$\overline{x}_{3:0}$	$\overline{x}_{2:0}$	$\overline{x}_{3:0}$
b_0	0	0	0
b_1	0	0	$(2^{-3}+2^{-5})$

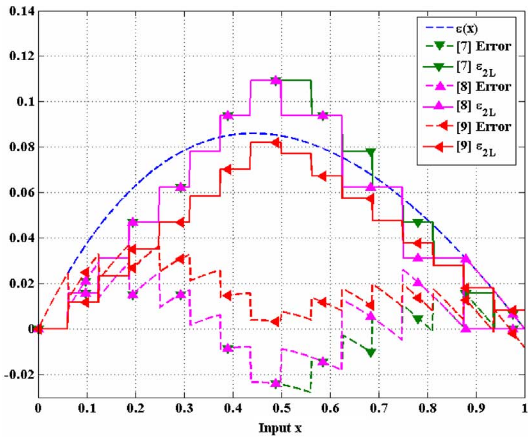

Fig. 2. Two-region linear approximations and their error.

Below, the methods for correcting the error made by Mitchell based on piecewise linear interpolation and LUTs are introduced.

A. Methods Based on Shift-and-Add Piecewise Linear Interpolation

Different methods for dividing the interval into two regions are proposed in [7], [8], [9]. Equation (4) shows the two-region linear interpolation approximation of $\varepsilon(x) \approx \varepsilon_{2L}(x)$

$$\varepsilon_{2L}(x) = \begin{cases} S_0 \cdot x_0' + b_0, & 0 \le x < 0.5 \\ S_1 \cdot x_1' + b_1, & 0.5 \le x < 1 \end{cases}. \tag{4}$$

The differences among the three methods are the values of slopes and constants used in the linear approximation of each region. Table I shows these values, where $x_{n.0}$ denotes the n most significant bits (MSB) of the x and $\bar{x}_{n.0} = 1 - x_{n.0} - 2^{-n}$.

As can be observed; the two-region methods propose the use of simple shifting and addition operations to avoid the multiplication operation required by the linear interpolation. Furthermore, they only operate with the most significant bits to reduce their hardware cost. Fig. 2 shows the Mitchell error, $\varepsilon(x)$, the two-region linear interpolations, $\varepsilon_{2L}(x)$, and their errors, $e = \varepsilon(x) - \varepsilon_{2L}(x)$.

Combet *et al.* [5] and Hall *et al.* [6] divided the interval $0 \le x < 1$ into four different regions. The resulting piecewise linear approximation, $\varepsilon_{4L}(x)$, for $\varepsilon(x)$ is in (5) and the values of the coefficients and slopes are in Table II. The coefficients used in [5] were selected by trial and error and coefficients used in [6] were selected to minimize the mean square error. The slopes and coefficients in [5] can be decomposed in a maximum of two power-of-two terms, and in [6] two slopes (37/128 and 29/128) need three power-of-two terms. On the other hand, the sum operations are performed with full-precision. Fig. 3 shows the Mitchell error, $\varepsilon(x)$, the four-region linear interpolations, $\varepsilon_{4L}(x)$, and their errors, $e = \varepsilon(x) - \varepsilon_{4L}(x)$

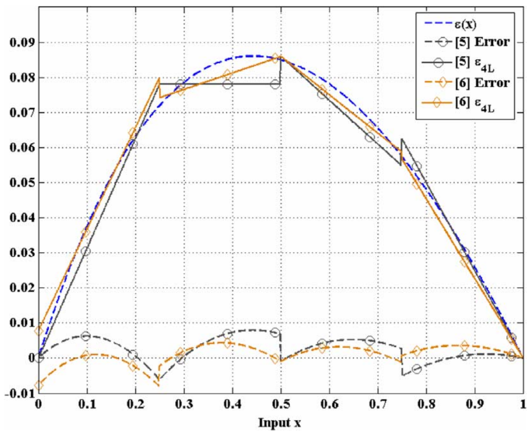

Fig. 3. Four-region linear approximations and their errors.

TABLE II
CONSTANTS USED IN FOUR REGION APPROXIMATION

	[5]	[6]
S_0	(5/16)	(37/128)
S_1	(5/64)	(3/64)
S_2	(1/8)	(7/64)
S_3	(1/4)	(29/128)
x_0', x_1'	x , 1	x
x_2', x_3'	$\overline{x}$	$\overline{x}$
b_0 , b_1	0	0
b_2 , b_3	(3/128) , 0	(1/32) , 0

$$\varepsilon_{4L}(x) = \begin{cases} S_0 \cdot x_0' + b_0, & 0 \le x < 0.5 \\ S_1 \cdot x_1' + b_1, & 0.25 \le x < 0.5 \\ S_2 \cdot x_2' + b_2, & 0.5 \le x < 0.75 \\ S_3 \cdot x_3' + b_3, & 0.75 \le x < 1 \end{cases}. \tag{5}$$

Abed *et al.* [8] proposed in their paper piecewise linear interpolations of $\varepsilon(x)$ for different regions. The two-region interpolation was expounded previously. Now, the six-region approximation, $\varepsilon_{6L}(x)$, of $\varepsilon(x)$ is presented in (6)

$$\varepsilon(x) \approx \begin{cases} 2^{-2} \cdot x_{5:0}, & 0 \le x < 0.0625 \\ 2^{-2} \cdot x_{5:0} + 2^{-6}, & 0.0625 \le x < 0.25 \\ (2^{-4} + 2^{-7} + 2^{-8}), & 0.25 \le x < 0.375 \\ (2^{-4} + 2^{-6} + 2^{-7}), & 0.375 \le x < 0.625 \\ (2^{-4} + 2^{-7}), & 0.625 \le x < 0.75 \\ 2^{-2} \cdot \bar{x}_{5:0}, & 0.75 \le x < 1 \end{cases}. \tag{6}$$

In this case, only the six most significant bits of x are used for the sum operation. Fig. 4 shows the Mitchell error, $\varepsilon(x)$, the six-region linear interpolation, $\varepsilon_{6L}(x)$, and its error, $e = \varepsilon(x) - \varepsilon_{6L}(x)$.

There are other more complex approximation methods based on piecewise linear approximation that achieve higher accuracy. For example, a non-uniform 15-region linear interpolation is presented in [12]. To bypass the multiplication operation in the linear interpolation $(\mathrm{a} \cdot x + \mathrm{b})$, the coefficient a is computed as a sum of shifts of x. These shifts are stored in a LUT.

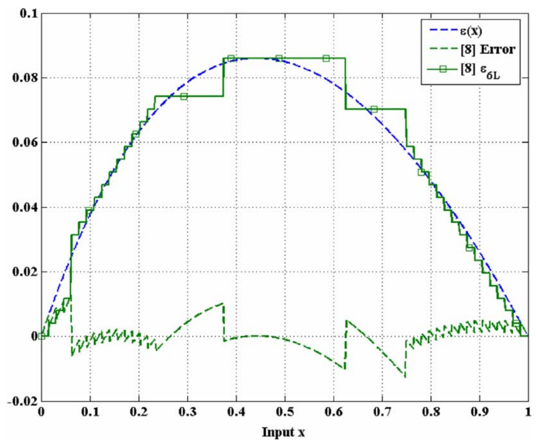

Fig. 4. Six-region linear approximation and its error.

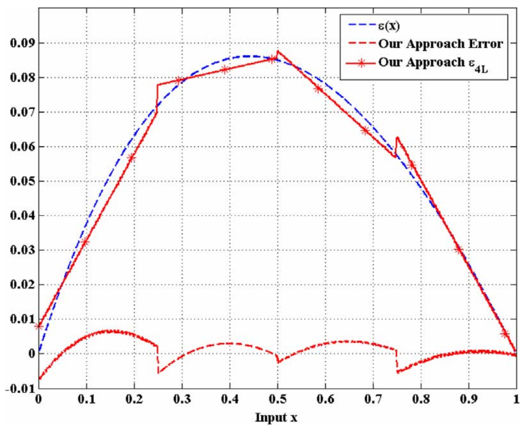

Fig. 5. Proposed 4-region linear approximation and its error.

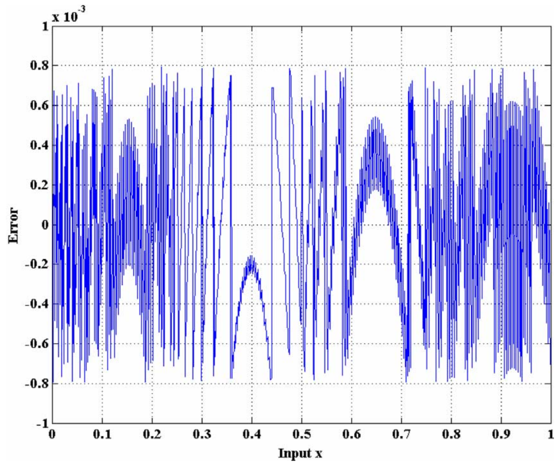

Fig. 6. Error with 4-region linear approximation and the residual error approximated by a 128 × 5 bits ROM.

B. LUT-Based Correction Techniques for Mitchell Method

In [10], a LUT was utilized to store Mitchell's error, which is added to Mitchell's approximation. A hardware efficient architecture based on this method is proposed in [11]. It uses a LUT followed by a multiplierless linear interpolation stage. The interpolation is defined as

$$\varepsilon(x) \approx a + \frac{(b-a)\cdot n_1}{2^{k-t}}, 0 \leq x < 1. \tag{7}$$

Once Mitchell's error is sampled at 2^t points (depending on the required size of the LUT), the parameter a is Mitchell's error from the LUT accessed by t MSB bits of x value, b is the next table value adjacent to a and k is the total number of MSB of x and n_1 is the decimal value of the last $k-t$ bits of x. To avoid the use of multipliers, the product in (7) is transformed to the logarithmic domain following this expression

$$(b-a)\cdot n_1 = \text{antilog}(\log(b-a) + \log(n_1)). \tag{8}$$

The method proposed in [11] computes (7) with three LUTs, to store $\log(\mathrm{a}+\mathrm{b}), \log(\mathrm{n}_1)$ and the antilog function, and two adders. It was compared with other LUT approaches like the symmetric bipartite table method (SBTM) [13], or the methods given by Brubaker [1], Maenner [14], and Kmetz [10], requiring less memory than the others for the same accuracy.

III. Proposed Logarithm Approximation

The proposed logarithm approximation is based on the Mitchell method with two error correction stages: first a four-region linear interpolation $(\varepsilon_{4L}(x))$ is performed; second, the residual error is approximated $(\varepsilon_R(x))$ by means of LUT-based methods to achieve higher accuracy. Hence, the approximation has these two terms

$$\varepsilon(x) \approx \varepsilon_{4L}(x) + \varepsilon_R(x), 0 \leq x < 1. \tag{9}$$

In our approach, the input interval, $0 \leq x < 1$, of Mitchell's error is divided into four uniform regions that are approximated by straight lines, whose slopes are restricted to a single power-of-two and where just the 8 MSB of x are used to reduce the hardware cost of the additions. The equation for the resulting piecewise linear approximation, $\varepsilon_{4L}(x)$, is shown in (10). The number of bits to truncate x and the coefficients used in these equations were selected by trial and error to minimize the error. The Mitchell error, $\varepsilon(x)$, the proposed four-region linear interpolation and its error are shown in Fig. 5.

Adding just the proposed correction stage, $\varepsilon_{4L}(x)$, to Mitchell's approximation increases the accuracy by up to 7 bits, which is higher than the one achieved by the other correction methods based on piecewise linear interpolation [5]–[9]

$$\varepsilon_{4L}(x) = \begin{cases} 0.008 + x_{7:0}\cdot 2^{-2}, & 0 \leq x < 0.25 \\ 0.07 + x_{7:0}\cdot 2^{-5}, & 0.25 \leq x < 0.5 \\ 0.15 - x_{7:0}\cdot 2^{-3}, & 0.5 \leq x < 0.75 \\ 0.25 - x_{7:0}\cdot 2^{-2}, & 0.75 \leq x < 1 \end{cases}. \tag{10}$$

To improve the accuracy of the obtained approximation a small table, called error-LUT, that stores an approximation of the residual error is added to Mitchell's method corrected with the proposed four-region straight-line approximation. The magnitudes of the values stored in the error-LUT (ε_R) are rather smaller than the magnitude of Mitchell's error values, so the length of the words to be stored is reduced, i.e., using a small LUT of 128 × 5 bits, the accuracy of the approximation can be improved by up to 10.4 bits. Fig. 6 shows the error obtained approximating the Mitchell error with $\varepsilon(x) \approx \varepsilon_{4L}(x) + \varepsilon_R(x)$, where $\varepsilon_R(x)$ is a 128 × 5-bit LUT. If higher accuracy is desired and the LUT size increases, the multipartite table method [15] can be applied to reduce the storage cost and more than 8 bits can be used to generate the proposed 4-region linear interpolation.

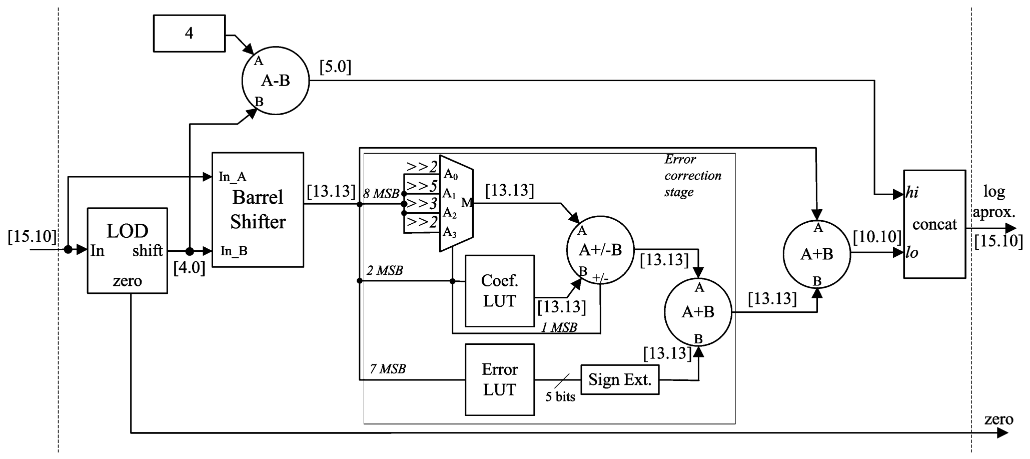

Fig. 7. Proposed architecture for log(N).

TABLE III
HARDWARE RESOURCES USED FOR A LOG(N) FOR A DIFFERENT OUTPUT ACCURACIES

Accuracy (bits)	10	12	14	16	18	20
Slices	90	120	162	180	248	298
LUT4	165	215	302	345	465	560
LUT size (bits)	640	896	2368	5632 (3LUTs)	12800 (3LUTs)	25600 (3LUTs)
Block RAM	0	0	0	1	1	1
Fmax (MHz)	65.8	65.1	63.8	60.8	57.5	54.9
Latency (clock cycles)	1	1	1	1	1	1

IV. ARCHITECTURE

The implementation scheme of the proposed logarithm approximation is shown in Fig. 7. The word width have been indicated in the figure using the format [N.F], where N is the number of bits of the word and F is the number of fractional bits.

The first step consists of locating the leading-one position. This operation is performed by a leading to one detector (LOD) based on the methodology proposed by Oklobdzija [16]. The integer part of the logarithm is obtained by subtracting the leading-one position from the number of integer bits of the input data. The leading-one position is used by a barrel shifter to re-align the data $(1 + x)$.

The content of barrel shifter is the $\log(1 + x) - \varepsilon(x)$ of Mitchell's approximation. The MSB of the barrel-shifter output is ignored; the two more significant remaining bits are used to select one of the four regions of the straight-line error approximation: the multiplexer is used to select a hard-wired right shift that implements a slope of the linear interpolation and selects a coefficient from the LUT; the n MSBs are used to address the error LUT. The value of n is obtained by simulation to achieve the target accuracy. If the size of this LUT increases, multipartite methods [15] can be applied to reduce its cost. The approximation of $\log(1 + x)$ is concatenated with the characteristic to form the output.

V. IMPLEMENTATION RESULTS AND COMPARISONS

The proposed architecture was modelled in VHDL and implemented in a Virtex-II PRO XC2VP30-7 Xilinx FPGA device. Area and maximum frequency were obtained from the Xilinx ISE 10.1 tool. Table III presents the results of the implementation of log(N) for output accuracy between 10 and 20 bits, and where the input and output was generated with 5 integer bits and as many fractional bit as needed to provide the required accuracy. The 4-region linear interpolation uses the 8 MSB and the error LUT was generated with a single table for accuracies from 10 to 14 bits, as the storage requirements are low. The 4-region linear interpolation uses the 14 MSB and the error LUT is decomposed into 3 LUTs in the cases with accuracies from 16 to 20 bits. Two of them (the two larger) are implemented in an embedded dual-port BlockRAM memory and the smallest is implemented with distributed memory based on 4-input LUTs.

TABLE IV
PERFORMANCE OF $\log(1 + x)$ USING THE LINEAR INTERPOLATION APPROACHES

	[5]	[6]	[7]	[8]	[8]	[9]	This work
Partitioned regions	4	4	2	2	6	2	4
Accuracy (Bits)	6.3	6.1	4.1	4	6.2	4.5	7
Slices	57	65	5	5	13	15	18
LUT4	109	128	9	8	24	27	36

Two kind of comparison were done: (1) the comparison among the piecewise linear interpolation methods and (2) the comparison with a LUT-based method. For comparison with the low-cost piecewise linear interpolation methods exposed in Section II.A, the 4-region linear interpolation of the proposed method (just the term $\varepsilon_{4L}(x)$ of (9)) was modeled in VHDL to compute the $\log(1 + x)$ operation with a 32-bit input mantissa and 27-bit result, and implemented in a Virtex-II PRO XC2VP30-7 Xilinx FPGA device. The methods of Section II.A were also modeled to compute the same operation and implemented in the same FPGA device. The results are shown in Table IV. As can be seen, the proposed 4-region linear interpolation improves the accuracy with respect to the other methods and its hardware cost is quite low (just 36 LUTs). Specifically it is lower than the 4-region linear interpolation methods proposed in [5] and [6].

To show the advantage of using the proposed linear interpolation method combined with a LUT-based method to correct the residual error, the proposed architecture was dimensioned to achieve the same accuracy as the one proposed in [11] with an 23-bit input mantissa. To our knowledge, this is the LUT-based architecture that requires the least storage resources, as is shown in [11]. Table IV shows the results of the implementation in a Virtex-II PRO XC2VP30-7 Xilinx FPGA device. Our method uses the 14 MSB for the linear interpolation and requires storing 6272 bits that were decomposed into 3 tables (256 × 13, 256 × 9 and 128 × 5) providing an accuracy of 16.8 bits. On the other

TABLE V
PERFORMANCE OF $\log(1+x)$ USING PROPOSED APPROACH COMPARED TO [11]

	This work	[11]
Slices	165	287
LUT4	200	210
F.F.	265	415
Fmax (MHz)	358	350
LUT size (bits)	6272	6656
(Block RAM)	(1)	(1)
Latency (clock cycles)	8	12
Accuracy (Bits)	16.8	16.7

hand, the method proposed in [11] requires storing 6656 bits that were decomposed into 3 tables (128×18, 128×18 and 128×16) providing an accuracy of 16.7 bits.

As can be seen, the proposed architecture reduces the LUT size by 5.7% generating the logarithm approximation with the same precision. Both methods have been pipelined to have as high a throughput as possible. Our approach exhibits similar working frequency to [11] with lower latency. The critical path in our approach is the delay of Block RAM in the Virtex-II PRO.

VI. CONCLUSION

A low cost, high speed logarithm approximation has been proposed based on Mitchell's method with a correction stage composed of piecewise linear interpolation and a LUT correction. The Mitchell error interval is divided into four regions that are approximated with straight-lines with power-of-two slopes and a truncated mantissa. The accuracy is increased by adding a small LUT or a multipartite table that stores the residual error obtained after the piecewise correction stage. The proposed approach achieves higher accuracy than similar low-cost piecewise linear interpolation methods found in the literature and lower area than LUT-based methods.

REFERENCES

[1] T. A. Brubaker and J. C. Becker, "Multiplication using logarithms implemented with read-only memory," *IEEE Trans. Comput.*, vol. C-24, no. 8, pp. 761–766, Aug. 1975.

[2] M. J. Shulte and J. E. E. Swartzlander, "Hardware designs for exactly rounded elementary functions," *IEEE Trans. Comput.*, vol. 43, no. 8, pp. 964–973, Aug. 1994.

[3] J. A. Piñero, M. D. Ercegovac, and J. D. Bruguera, "High-Radix Logarithm with selection by rounding: Algorithm and Implementation," *J. VLSI Signal Process.*, no. 40, pp. 109–123, 2005.

[4] J. N. Mitchell, "Computer multiplication and division using a binary logarithms," *IEEE Trans. Electron. Comput.*, vol. 11, pp. 512–517, 1962.

[5] M. Combet, H. Van Zonneveld, and L. Verbeek, "Computation of the base two logarithm of binary numbers," *IEEE Trans. Electron. Comput.*, vol. EC-14, no. 6, pp. 863–867, 1965.

[6] E. L. Hall, D. D. Lynch, and S. J. Dwyer, "Generation of products and quotients using approximate binary logarithms for digital filtering applications," *IRE Trans. Comput.*, vol. 19, pp. 97–105, 1970.

[7] S. Sangregory, C. Brothers, D. Gallagher, and R. Siferd, "A fast low-power logarithm approximation with CMOS VLSI implementation," in *Proc. 42nd Midwest Symp. Circuits Syst.*, 1999, vol. 1, pp. 388–391.

[8] K. H. Aded and R. E. Siferd, "CMOS VLSI implementation of low-power logarithmic converter," *IEEE Trans. Comput.*, vol. 52, no. 9, pp. 1221–1228, 2003.

[9] T. B. Juang, S. H. Chen, and H. J. Cheng, "A lower error and ROM-free logarithmic converter for digital signal processing applications," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 56, no. 12, pp. 931–935, Dec. 2009.

[10] G. L. Kmetz, "Floating Point/Logarithmic Conversion Systems," U.S. patent 4583180, Apr. 15, 1986.

[11] S. Paul, N. Jayakumar, and S. P. Khatri, "A fast hardware approach for approximate, efficient logarithm and antilogarithm computations," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 17, no. 2, pp. 269–277, Feb. 2009.

[12] B. G. Nam, H. Kim, and H. J. Yoo, "Power and area-efficient unified computation of vector and elementary functions for handheld 3d graphics systems," *IEEE Trans. Comput.*, vol. 57, no. 4, pp. 490–504, Apr. 2008.

[13] M. J. Schulte and J. E. Stine, "Approximating elementary functions with symmetric bipartite tables," *IEEE Trans. Comput.*, vol. 48, no. 8, pp. 842–847, Aug. 1999.

[14] R. Maenner, "A fast integer binary logarithm of large arguments," *IEEE Micro*, vol. 7, no. 6, pp. 41–45, 1987.

[15] M. J. Shulte and J. E. Stine, "The symmetric table addition method for accurate function approximation," *J. VLSI Signal Process.*, vol. 11, pp. 1–11, 1999.

[16] V. G. Oklobdzija, "An algorithmic and novel design of a leading zero detector circuit. Comparison with logic synthesis," *IEEE Trans. Very Large Scale Integr. VLSI) Syst.*, vol. 2, no. 1, pp. 124–128, 1994.

Defect-Oriented LFSR Reseeding to Target Unmodeled Defects Using Stuck-at Test Sets

Xrysovalantis Kavousianos, Vasileios Tenentes, Krishnendu Chakrabarty, and Emmanouil Kalligeros

***Abstract*—Defect screening is a major challenge for nanoscale CMOS circuits, especially since many defects cannot be accurately modeled using known fault models. The effectiveness of test methods for such circuits can therefore be measured in terms of the coverage obtained for unmodeled faults. In this paper, we present a new defect-oriented dynamic LFSR reseeding technique for test-data compression. The proposed technique is based on a new output-deviation metric for grading stuck-at patterns derived from LFSR seeds. We show that, compared to standard compression-driven dynamic LFSR reseeding and a previously proposed deviation-based method, higher defect coverage is obtained using stuck-at test cubes without any loss of compression.**

***Index Terms*—Defect-oriented testing, dynamic reseeding, embedded testing, linear decompressors, static reseeding.**

I. INTRODUCTION

A key objective of manufacturing testing is defect screening. In addition to the single stuck-at fault model, delay faults must be targeted since many defects in newer technologies lead to timing failures. Some defects for nanometer technologies cannot even be accurately modeled using known fault models [13]. The need for new and complex fault models is therefore leading to a rapid increase in test-data volume (TDV) for integrated circuits. Hence, defect-oriented test-data compression is needed to cope with these challenges.

Manuscript received January 30, 2010; revised May 12, 2010; accepted August 28, 2010. Date of publication October 28, 2010; date of current version October 28, 2011. The work of K. Chakrabarty was supported in part by the Semiconductor Research Corporation under Contract no. 1588.

X. Kavousianos and V. Tenentes are with the Computer Science Department, University of Ioannina, GR 45110, Greece (e-mail: kabousia@cs.uoi.gr; tenentes@cs.uoi.gr).

K. Chakrabarty is with the Department of Electrical and Computer Engineering, Duke University, Durham, NC 27708 USA (e-mail: krish@ee.duke.edu).

E. Kalligeros is with the Information and Communication Systems Engineering Department, University of the Aegean, GR 83200 Samos, Greece (e-mail: kalliger@aegean.gr).

Digital Object Identifier 10.1109/TVLSI.2010.2079961