A New Hardware Implementation of Base 2 Logarithm for FPGA

A. M. Mansour, A. M. El-Sawy, M. S. Aziz, and A. T. Sayed
Wireless Advanced System Innovations and Electronics Art, Wasiela, Cairo, Egypt
Email: {ahmed.mansour, ali.mohamed, mbsami, ahmed.sayed}@wasiela.com

***Abstract*—Logarithms reduce products to sums and powers to products; they play an important role in signal processing, communication and information theory. They are primarily used for hardware calculations, handling multiplications, divisions, powers, and roots effectively. There are three commonly used bases for logarithms; the logarithm with base-10 is called the common logarithm, the natural logarithm with base-e and the binary logarithm with base-2. This paper demonstrates different methods of calculation for $\log_2$ showing the complexity of each and finds out the most accurate and efficient besides giving insights to their hardware design. We present a new method called Floor Shift for fast calculation of $\log_2$, and then we combine this algorithm with Taylor series to improve the accuracy of the output, we illustrate that by using two examples. We finally compare the algorithms and conclude with our remarks.**

***Index Terms*—logarithms, $\log_2$, floor, Taylor series expansion, exponent, mantissa, CORDIC, SQNR, variance, FPGA**

I. Introduction

Logarithm tables have been used extensively in the early 17th century to perform calculations used for many applications in mathematics and science, until replaced in the latter half of the 20th century by electronic calculators and computers. Logarithmic scales reduce wide-ranging quantities to smaller scopes; the binary logarithm is often used in digital communications and information theory because it is closely connected to the binary numeral system, information entropy involves the binary logarithm, this is needed to compare the efficiency of different probable implementation alternatives. If a number n, greater than 1, is divided by 2 repeatedly, the number of iterations needed to get a value at most 1 is the integral part of $\log_2$ (n). This idea is used in the analysis of several algorithms that we present [1].

The effective methods to compute the logarithmic values of data are divided into two main types; one is the look-up table based algorithms and the other is Iterative methods. This paper presents different algorithms that are used to calculate $\log_2$ showing the accuracy and complexity in hardware design.

The remaining of this paper is organized as follows; Section 2 covers calculating $\log_2$ using a look-up table, Section 3 shows an Iterative method for calculating $\log_2$ where accuracy depends on the number of iterations, Section 4 refers to the CORDIC method and how it is used for calculating $\log_2$, Section 5 presents our proposed method for calculating $\log_2$, Section 6 presents our proposed method in calculating $\log_2$ using Taylor series. We compare the algorithms in Section 7 and 8, and finally Section 9 presents our conclusions and future work.

Manuscript received July 26, 2014; revised October 24, 2014.

II. Look-Up Table Method

This algorithm depends on selecting a certain range of interest and storing a LUT in ROM. It is the traditional method for calculating the logarithm for any base, but we must consider memory resources needed for look-up tables and available space. Implementation is direct forward since there are not any decisions taken [2], [3].

III. Iterative Method

The input value X is first divided into mantissa m and exponent e representation as:

$$X = m2^e \tag{1}$$

The logarithmic value of X can be expressed in terms of m and e as:

$$\log_2(X) = \overbrace{\underbrace{e}_{IntegerPart} + \underbrace{\log_2(m)}_{FractionPart}}^{\log_2(X)} \tag{2}$$

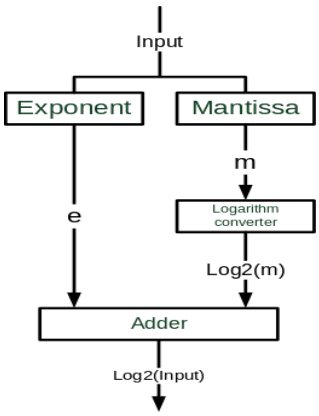

Figure 1. Main block diagram for Iterative algorithm.

The logarithmic output is simply the sum of the integer part and the fractional part as shown in Fig. 1. The

doi: 10.12720/ijsps.3.2.177-182

exponent e is a signed integer and it is exactly the integer part of the output. Since the exponent is of base-2 and the mantissa m takes values in the range of [1, 2], then its logarithmic value is in the range of [0, 1]. Fig. 2 shows the block diagram for calculating $\log_2$ (m).

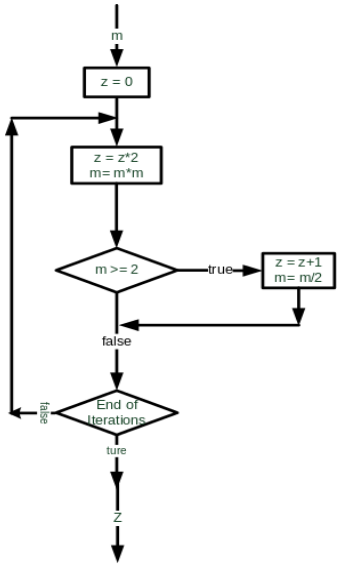

Figure 2. Log_2 (m) calculation.

There are a lot of methods used to implement Log_2 using iterative method as in [1], but here we present more simple and fast method. Fig. 3 presents the exact value and the calculated value of Log_2 (m) with different numbers of iterations, showing that as number of iterations increases accuracy increases and vice versa.

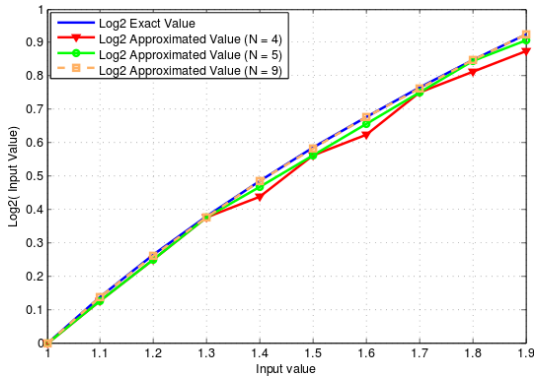

Figure 3. Exact value and calculated value of Log_2 (m).

IV. CORDIC METHOD

The CORDIC algorithm uses only adders and shifters. It provides a relatively high precision output, which is suitable for hardware implementation. Calculating $\log_2$ using the CORDIC algorithm depends on linear and inverse hyperbolic tangent modes of CORDIC, then using (4) for conversion between logarithm base-2 and natural logarithm [4], [5].

$$ln(m) = 2 * \tanh^{-1}\frac{(m-1)}{(m+1)} \quad (3)$$

$$\log_2(m) = \log_e(m) * \log_2(e) \quad (4)$$

The design of CORDIC algorithm for calculating $\log_e$(X) using CORDIC method is presented in Fig. 4.

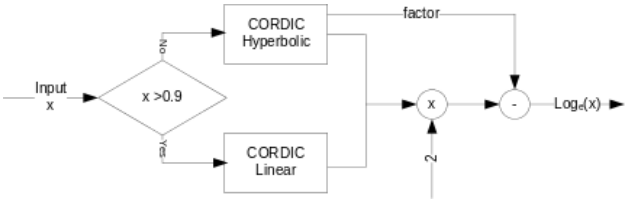

Figure 4. Main block diagram for CORDIC method.

Referring to error analysis methodology presented in [6], it shows that our CORDIC design is more accurate than the design presented in [5] for the same example points that we mentioned in Table I showing our minimized error.

TABLE I. ERROR ANALYSIS

Input Data	0.9375	0.4375	1.3125	3.375
Error	3.67E-005	1.40E-004	8.31E-005	3.53E-005

V. NEW FLOOR-SHIFT METHOD

This section presents our new method of implementing $\log_2$, let X be a binary number

$$X = X_{N-1}X_{N-2}.....X_1X_0 \quad (5)$$

where X_k is the binary digit, X_k={0, 1}. Assume that the binary digits X_{N-1} X_{N-2} ....X_{k+1} of X are all zeros and X_k is 1. Then the leading one is X_k, thus we get:

$$X = \sum_{i=0}^{k} x_i 2^i = 2^k + \sum_{i=0}^{k-1} x_i 2^i \quad (6)$$

Since the logarithmic value of X can be expressed in terms of m and e as in (1), so our method tries to calculate $\log_2$ (X) directly by getting the integer part then add the fraction part.

Our proposed method based on that presented in [7] which creates a dependence between accuracy and number of iterations, but here we calculate $\log_2$ (X) directly independent of the number of iterations; getting the floor of $\log_2$ of the input obtaining the integer part then adding $\log_2$ of the mantissa that presents the fraction part. This operation is described as follows:

1. We calculate $\log_2$ (X) by searching for the leading one (i.e., position of X_k) to get the integer part of $\log_2$ (X).
2. We can get the approximated value for $\log_2$ (mantissa) using (7), this operation is described in Fig. 5, noting that we must have the same length N for data after shifting the decimal point [8].

$$\log_2(m) = \frac{x - 2^{\lfloor \log_2(x) \rfloor}}{2^{\lfloor \log_2(x) \rfloor}} \quad (7)$$

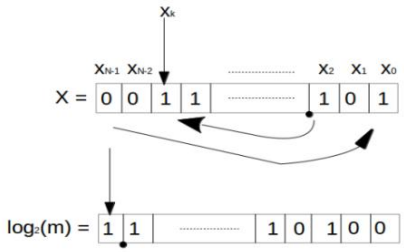

Figure 5. Floor shift method operation.

If we apply (7) directly, we will get a maximum error between exact and calculated values of 0.086, but we combine a simple LUT for certain values range and adding 0.04 as bias to center the error around zero for other values, this modification will minimize the error to half. The next section will present more improvements to this method for calculating mantissa with more accuracy depending on Taylor series.

Ex: $\log_2(0.1) \Rightarrow$ integer part = $\log_2(0.1)$ = -4 and fraction part calculation using (7) equals 0.6, so $\log_2(0.1)$ using this method = -3.4.

VI. Taylor Series Based Improved Method

This method used Taylor series expansion to calculate Log_2 of the mantissa and then add its value to the floor of $\log_2$ of the input as shown in the last section, here we use (8) to get the fraction part using Taylor series as in (9), (10). We will demonstrate how error depends on the point of expansion at x=1 and x=1.5.

$$log_2(m) = \frac{X}{2^{\lfloor log_2(X) \rfloor}} \tag{8}$$

A. *Taylor Series Expansion Centered at x=1*

Taylor series expansion can be written as follows:

$$ln(x) = \sum_{n=1}^{\infty} \frac{(-1)^{n-1}}{n}(x-1)^n \tag{9}$$

Also the conversion equation between ln (x) and $\log_e$ (x) to $\log_2$ (x) can be rewritten as:

$$log_2(x) = \frac{ln(x)}{ln(2)} = \frac{log_e(x)}{log_e(2)} \tag{10}$$

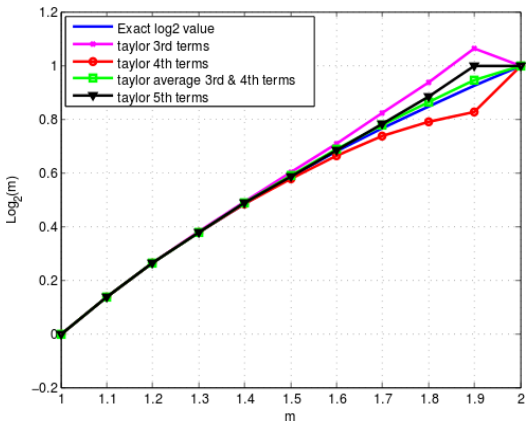

Figure 6. Exact and calculated values for Taylor series algorithm.

Fig. 6 shows $\log_2$ (m) using Taylor series when taking 3rd, 4th terms and taking the average of both will produce approximated value for exact $\log_2$ (m), noting that the accuracy of output value increases as number of terms increases, also noting here the performance of taking average of both 3rd and 4th terms is better than using 5th terms for this case.

The error in this case of algorithm (x=1) can be minimized if we add certain values to the diverging ones.

Ex: add 0.0119 to values greater than 1.6, the error distribution is shown in Fig. 7.

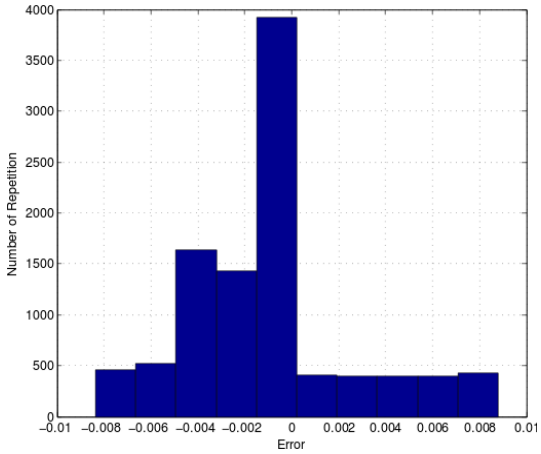

Figure 7. Difference between exact and calculated values for Taylor series algorithm with average Taylor terms with modification.

B. *Taylor Series Expansion Centered at x=1.5*

Taylor series expansion can be written as follows:

$$ln(x) = ln(1.5) + \sum_{n=1}^{\infty} \frac{(-1)^{n-1}}{n*(1.5)^n}(x-1.5)^n \tag{11}$$

The fraction part here of ($\log_2$ (m)) is calculated by using Taylor series expansion taking 4-terms as in (12):

$$log_2(m) = \frac{ln(1.5)}{ln(2)} + \frac{1}{ln(2)} \sum_{n=1}^{4} \frac{(-1)^{n-1}}{n*(1.5)^n}(x-1.5)^n \tag{12}$$

This case of algorithm does not need any modification or biasing as shown in the error distribution in Fig. 8.

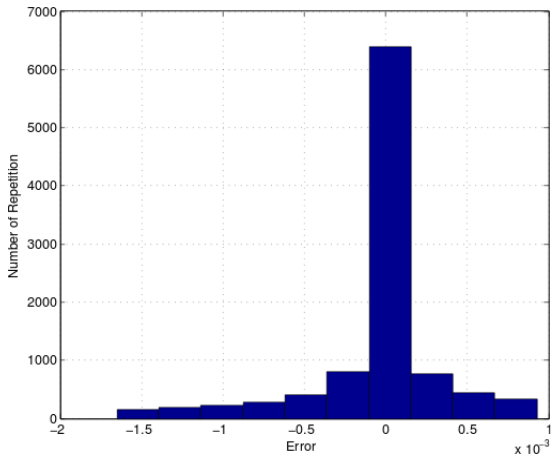

Figure 8. Difference between exact and calculated values for Taylor series algorithm with 4 terms Taylor series centered at x=1.5.

VII. Comparison

Fig. 9 presents the error analysis of each algorithm as a function in SQNR and shows the maximum error that can be obtained. It is clear that as SQNR increases the error decreases and vice versa, also we can observe the curves are overlapping together at specific SQNR values giving the same error for all algorithms. If we increase the number of bits, the additional bits will have different effect in reducing the error and increasing SQNR of each algorithm. For example the gain of increasing SQNR of CORDIC method is less than Taylor which is also less than both, the Iterative and LUT methods, also the gain of CORDIC method is higher than floor shift method. Table II presents the maximum value of SQNR that we can obtain to get minimum error for each algorithm.

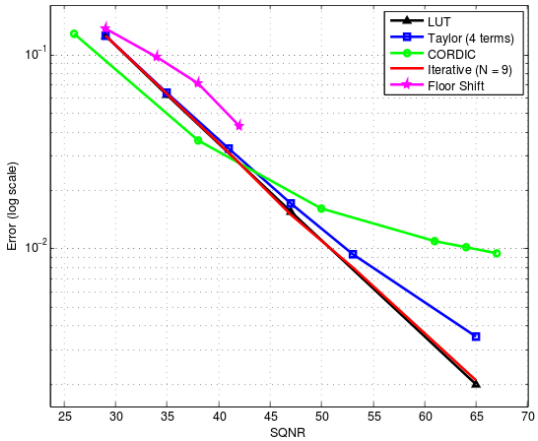

Figure 9. SQNR vs. Error of each algorithm.

TABLE II. Maximum SQNR

Algorithm	LUT	Iterative (N=9)	CORDIC	Floor-Shift	Taylor
SQNR	107	71	67	42	75

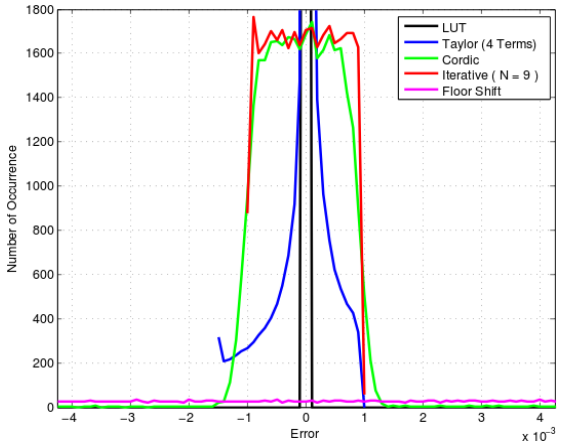

Figure 10. Error distribution of each algorithm.

Fig. 10 shows that the error distribution at maximum value of SQNR for CORDIC and Iterative algorithms has a uniform distribution around zero, while the mean of the errors in Taylor and LUT algorithms are at zero with a little variance. Floor-Shift algorithm has a uniform distribution of error for the whole range.

Based on error distribution curves in Fig. 10, and the values of SQNR in Table II. We chose the LUT, CORDIC, Taylor and Iterative as the best algorithms for our next clarifying comparison shown in Fig. 11.

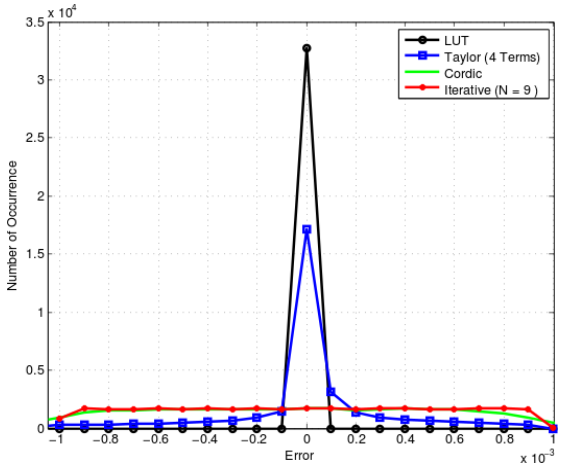

Figure 11. Error distribution of efficient algorithms.

Table III shows that the Taylor algorithm performance with variance = 1.5e-07 and can be reduced if we increase the number of terms, also the CORDIC algorithm has a variance = 9.8e-07 compared to the variance of the LUT algorithm which is 7.6e-11. Next section presents another comparison from hardware metrics showing the difference of each algorithm in (area, latency, power, speed...etc.).

TABLE III. Variance of Each Algorithm

Algorithm	LUT	Iterative (N=9)	CORDIC	Floor-Shift	Taylor
Variance	7.60E-011	3.20E-007	9.80E-007	2.80E-004	1.50E-007

VIII. Hardware Implementation

According to Table II and Fig. 9, we fix the SQNR to 65 for the LUT, Iterative with number of iterations equals 9, CORDIC and our new method combined with Taylor series for hardware implementation and comparison. All the hardware designs are pipelined to support high throughput. We targeted FPGA Xilinx platform with its device xc6vlx240t, package ff784, and speed grade-3. The synthesis of the chosen algorithms is done with clock constraint equals 7.5ns (≈133.33NHz) and the results are summarized in Table IV, Fig. 12, and Fig. 13, also show the main resources distribution for the implemented designs, their speed and power.

The main factors that are used in comparing two digital designs are speed, area and power, regarding the synthesis results in Table IV. Considering the speed, Table IV shows that the LUT is the fastest one with maximum possible frequency (≈324.254MHz) utilizing 13 RAMB36E1/FIFO36E1 blocks, a dynamic power (≈57.32mW) and a latency of 2 clock cycles. The Iterative algorithm follows the LUT with a maximum frequency (≈211.73MHz), utilizing 9 DSP48E1s which is smaller in the area than RAMB36E1/FIFO36E1 but it uses more flip flops than the LUT, and it consumes less

dynamic power (≈57.32mW) but with a higher latency equals 20 clock cycles. The CORDIC comes next on the list with a maximum frequency (≈194.288MHz), utilizing 2 DSP48E1s with more logic slices, above 1000 slices, and hence it is a power hungry design, it consumes a dynamic power (≈84.29mW) and it has a latency of 21 clock cycles that is higher than the LUT and Iterative latencies. The Taylor based design comes in last according to speed which is (≈151.286MHz), it uses 7 DSP48E1s with less logic slices than the CORDIC and Iterative, it is the most power efficient design as it only consumes a dynamic power (≈13.64mW), it has a latency of 9 clock cycles that make it a little worse than the LUT.

TABLE IV. SYNTHESIS RESULTS

	LUT	Iterative (N=9)	CORDIC	Taylor
FF	15	393	1170	505
LUT Slices	1	206	4509	598
Slices	6	119	1387	205
RAMB36E1/FIFO36E1	13	0	0	0
DSP48E1	0	9	2	7
Dynamic Power (mW)	57.32	29.55	84.29	13.64
Latency (C.C)	2	20	21	9
Max. Frequency (MHz)	324.254	211.73	194.288	151.286

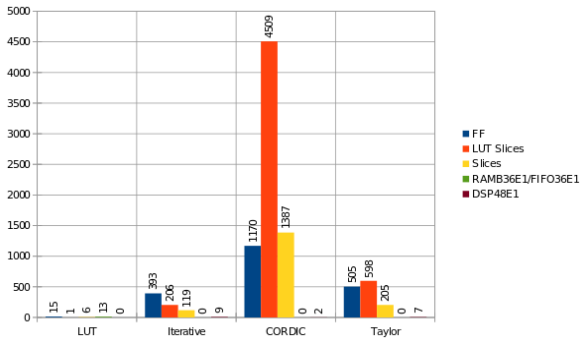

Figure 12. Hardware resources distributions chart.

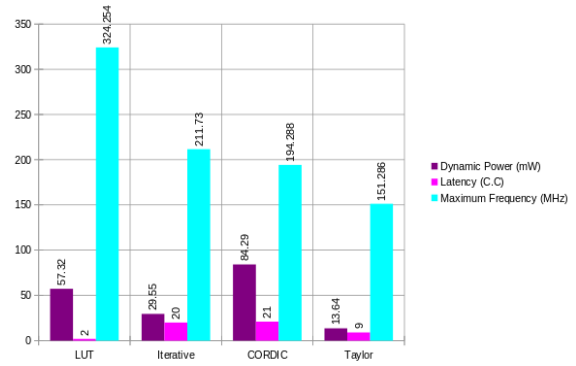

Figure 13. Speed and power chart.

IX. CONCLUSION

This paper presents the most common algorithms that are used for the calculation of $\log_2$. It also proposes a new technique for $\log_2$ calculation that exhibits good tradeoff between speed, power, area and accuracy. It is similar to the LUT in that the accuracy is configurable, viz. more bits in LUT provides the same effect on variance as adding terms to the Taylor series or increase number of iterations. Our hardware analysis shows that the LUT algorithm is the fastest one but it is not area and power efficient. The Iterative has average speed, area and power but its latency is high compared to the LUT and Taylor. The CORDIC has average speed, but it consumes the highest power compared to the others. For the same speed, Taylor based design is the most power efficient design compared to the others, it also has a reasonable latency. Our future work will focus on verifying the performance of each algorithm using different number of bits at different SQNR values and demonstrate its effect on error analysis and hardware metrics such as (speed, power, frequency,…etc.), we will also try to optimize the draw-backs of each algorithm, then combining the advantages of each algorithm to obtain a fully optimized one.

REFERENCES

[1] D. K. Kostopoulos, "An algorithm for the computation of binary logarithms," *IEEE Transactions on Computers*, vol. 40, no. 11, Nov. 1991.

[2] H. Hassler and N. Takagi, "Function evaluation by table look-up and addition," in *Proc. 12th Symp. on Computer Arithmetic*, Jul. 1995, pp. 10-16.

[3] D. D. Sarma and D. W. Matula, "Measuring the accumcy of ROM reciprocal tables," in *Proc. IEEE 11th Symp. on Computer Arithmetic*, Aug. 1994, pp. 932-940 .

[4] P. K. Meher, J. Valls, T. -B. Juang, K. Sridharan, and K. Maharatna, "50 years of CORDIC: Algorithms, architectures and applications," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 56, no. 9, Sep. 2009.

[5] B. Q. Liu, H. Ling, and Y. Xiao, "Base-N logarithm implementation on FPGA for the data with random decimal point positions," in *Proc. IEEE 9th International Colloquium on Signal Processing and its Applications*, Kuala Lumpur, Malaysia, Mar. 8-10, 2013, pp. 17-20.

[6] C. K. Anand and A. Sharma, "Unified tables for exponential and logarithm families," *ACM Trans. Math. Softw.*, vol. 37, no. 3, Sep. 2010.

[7] D. K. Kostopoulos, "An algorithm for the computation of binary logarithms," *IEEE Transactions on Computers*, vol. 40, no. 11, pp. 1267-1270, Nov. 1991.

[8] S. E. Tropea, "FPGA Implementation of base-N logarithm," in *Proc. 3rd Southern Conference on Programmable Logic*, Feb. 2007, pp. 27-32.

Ahmed M. Mansour was born in Alexandria, Egypt, in 1989. He received the BSc. degree in electrical and communication engineering from Alexandria University, Egypt, in 2011 with honor. He is currently M.Sc. student in electrical and communication department at Alexandria University, Egypt.

In 2012, he joined the Department of Computer and Communication Engineering, University of Alexandria, as a Lecturer. Since January 2013, he joined Wasiela Company as system design engineer. In Sep 2014, he joined VT mena as a researcher. His current research interests include digital signal processing, array signal processing and wireless communication systems.

Ali M. El-Sawy was born in Cairo, Egypt, in 1990, got his BSc. from electronics and electrical communication engineering department, Ain Shams University, in July 2011.

In October 2011, he joined Wasiela Company as a digital design engineer, where he is currently a team leader. His main areas of research interest include hardware implementation and optimization for wireless communication systems and digital signal processing.

Moataz S. Aziz was born in Cairo, Egypt, in 1986. He received the B.S. in electrical and communication engineering from Cairo University, Egypt, in 2008.

He joined an embedded system company in Egypt. In 2010, he joined Wasiela Company as IP design house focusing on physical layer design and implementation of cutting edge digital communication Systems LTE & DVB-T2/C2. He is currently a Team leader. His main areas of research interest include wireless communication systems and digital system architecture and design.

Ahmed T. Sayed was born in Cairo in 1972, got his BSc. From Cairo University from the electronics and communications department, in 1995 with Honors. He received his MSc. in computer engineering in 1997 from the University of Louisiana at Lafayette with Honors.

He worked for Intel Corp. for ten years. He received his PhD from the University of California, Davis in 2007. He has taught at several public and private universities in Egypt. He joined IBM Cairo in Sep. 2008 as a research scientist. He holds one patent with Intel and two with IBM. Joined Varkon Semiconductors as a digital design manager starting Feb. 2011. He is interested in low power digital design and parallel programming for signal processing.