SEMARAK ILMU PUBLISHING

Journal of Advanced Research in Applied Sciences and Engineering Technology

Journal homepage:
https://semarakilmu.com.my/journals/index.php/applied_sciences_eng_tech/index
ISSN: 2462-1943

Approximate Square and Square Root Calculation without Multiplication or Division for DSP Applications

Dina Mohamed Ellaithy[1,*]

[1] Department of Microelectronics, Electronics Research Institute (ERI), Cairo, Egypt

ARTICLE INFO

Article history:
Received 26 November 2023
Received in revised form 17 April 2024
Accepted 15 June 2024
Available online 15 July 2024

Keywords:
Square; square root; low power consumption; approximation algorithm; logarithm number system; logic synthesis; energy-efficient processing

ABSTRACT

Square and square root functions are a fundamental step in many real-time digital signal processing applications such as image processing, signal processing, and filtering. The evaluation performance of these applications depends on the hardware algorithm used in the implementation of the square and square-root calculation. In this paper, we present an efficient algorithm that is capable of performing square and square root calculations without multiplication or division. Multiplier-less circuitry is introduced throughout the architecture to avoid the use of costly multipliers and dividers. The proposed square and square-root design exhibit good performance according to accuracy, latency, and power dissipation. Implementation and synthesis using Synopsys Design Compiler with a 90 nm CMOS process, 1.0 V supply voltage standard cell library, demonstrates that the proposed architecture leads to a 45% reduction in power dissipation while achieving higher accuracy when compared to the state-of-the-art approximate architectures. In addition, the proposed architecture is able to process the square and square-root calculation in 3 ns.

1. Introduction and Related Work

The wide range of applications of digital signal processing (DSP) in several fields such as image compression, pattern recognition, demodulation, decoding, adaptive filtering, and least mean squaring has increased the demand for efficient implementations of real-time functions [1,2]. Also, real-time control systems applications in various fields such as motor drivers, power converters, and power factor correctors require energy-efficient function processing [3-7]. The widely exploiting arithmetic functions in digital signal processing (DSP) and control systems are multiplication function, squaring function, division function, square root function, root mean squares and total harmonic distortion. The exact hardware implementation of these functions consumes huge power dissipation, large area, and great delay that slow down the overall computations. Consequently, the exact computation makes limitations on the battery-powered devices due to power, speed and area challenge [8,9]. For example, the exact hardware implementation of multiplication function requires a generation of large number of partial products. Furthermore, a large amount of addition and

* *Corresponding author.*
E-mail address: dina_elessy@eri.sci.eg

https://doi.org/10.37934/araset.48.2.121135

shifting of these partial products are performed to obtain the final results. This cost high power, large area, and long delay. Thus, the major source of overhead of the multiplication function comes from the complete generation of the partial products. However, such mentioned DSP and control systems applications are error resilient, which denotes that accuracy lack are allowed in the processing and computations without affecting the overall performance [10,11]. Therefore, the approximate arithmetic computations scheme might be exploited to relax the partial products generation and so the power, area, and delay overhead. Approximate calculation algorithms are hardware efficient and offer good accuracy solutions [12-23].

The ever-increasing demand for simple hardware implementation with appropriate accuracy for DSP and control systems applications has led to utilize the approximate arithmetic in computation of complex functions. Low power dissipation, less area, and low delay have been achieved according to the algorithm and the method of approximation that being used. Since the square function and square root function are frequently utilized in DSP and control systems applications, several previous research efforts have studied the implementation of these two functions. There are many different approximate algorithms to implement the square and square-root operations, each with trade-off power, delay, and accuracy [1-26]. However, it isn't easy to have good accuracy with high-performance computations. Therefore, these approximation algorithms deal with the design complexity that sacrifices approximated errors.

The simple implementation of 2-bit exact multiplication uses four AND gate and two half adder (HA) as shown in Figure 1(a). Squarer is a specific state of multiplier in which both the inputs are same. The circuit implementation of 2-bit multiplier may be used as a 2-bit squarer by replacing the second input operand by the first input operand as shown in Figure 1(b). The four-bit result from the 2-bit multiplication process presented in Figure 1(a) requires four AND gate and two half adder to generate the partial products. While in case of A = B, the output bits of the 2-bit input circuit shown in Figure 1(a) are: P0 (1-bit): A0 AND A0 which becomes A0. P1 (1-bit): (Sum bit of HA) = A1.A0 XOR A0.A1 which results '0'. Carry bit of first HA is redirected as input to the second stage HA. P2 (1-bit): sum of second HA. P3 (1-bit): carry of second HA. The final result is represented as (P3P2P1P0). Thus the 2-bit binary squarer circuit is confirmed in Figure 1(b). This circuit uses one AND gate and one HA. As the number of bits increases, the hardware cost is increases significantly.

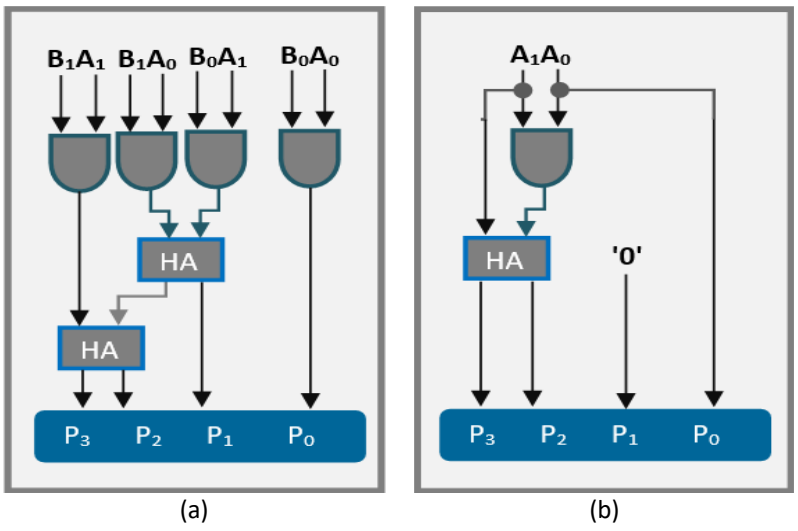

Fig. 1. Hardware architecture of (a) 2-bit exact multiplication process, (b) 2-bit exact square process

Figure 2 presents the partial products of eight-bit squaring process showing the symmetry of partial products. The complete computation of square function is performed by three phases. First phase includes the generation of partial products. Then minimize the number of partial product rows to two rows. At final, using adder for adding these two rows to obtain the final result. The implementation of squarers can be approximated by removing several least significant columns of the partial product matrix. Different error reduction techniques are proposed to control the overall accuracy performance.

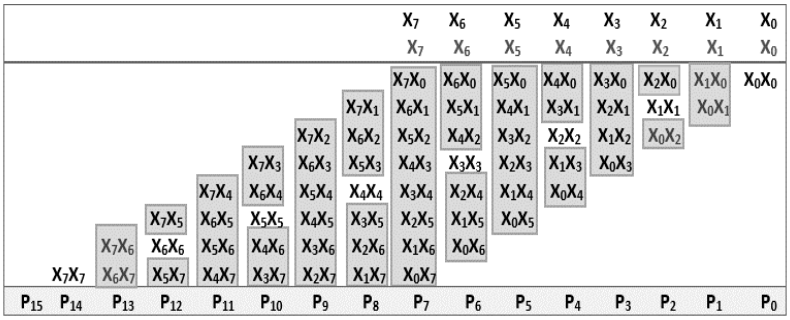

Fig. 2. Exact eight-by-eight multiplication partial products

For the square function approximation computation, previous work can be categorized in, approximate Booth squarer technique, folding technique, Indian Vedic based computation technique by Reddy [15], approximate square-accumulate technique by Gillani *et al.,* [19], array based approximate multiplier or truncated squarer technique, and approximate logarithmic squaring technique [1,2,10-19].

In order to diminish the partial product generation, approximate Booth encoding of input operands has been utilized in the approximate Booth squarer scheme [11,12]. This Booth encoding relaxes the accumulation of the partial products. However, the encoder design becomes more complex. Also, as the radix of the encoder increases, the hardware complexity of the squarer increases. Moreover, the folding techniques exploits the similarity of partial products due to the two inputs are the same [11,13,14]. Thus, the number of partial products for the squarer can be decreased. Combining the prior two techniques, approximate Booth technique and folding of partial products technique, reduces the number of partial products. For example, the Booth folding encoding squarer by Banerjee and Das [13] employs both the radix-4 Booth encoding technique and folding technique. As a result, the number of partial products is decreased by half as compared to simple folding technique. Differently, Vedic by Reddy [15] based computation technique is one of the ancient Indian mathematical that involves mainly 16 sutras that deals with the different branches in mathematics based on arithmetic, algebra, geometry. In this technique, larger calculations are reduced to be simpler in which the large magnitude number is divided into smaller magnitude numbers and concatenated smaller magnitude numbers. Else, the approximate square-accumulate technique that calculates the inner product of a vector with itself to confirm effective quality-efficiency tradeoff by Gillani *et al.,* [19]. Where the approximations are restricted in terms of presenting errors and thus controlling the calculating efficiency. Ordinarily, the squaring operation is carried out by utilizing generic multiplier. To reduce the partial products and hardware, truncated squarer removes the least significant columns of the partial product array [16-18]. Also, approximate generic adders and multipliers are employed in the array multiplier technique.

For extremely low power, approximate logarithmic squaring technique has been proposed as essential technique that substitutes the prior approximate squarer techniques [1,2,10]. All these

approximation techniques to provide contribution to the final result, by reducing area, delay, and power consumption at a price in terms of approximation error.

Similarly, the computation of the square root consumes a large amount of hardware resources and a large delay which makes it necessary to exploit an approximate square root calculation algorithm [3-5,20,21,27]. In order to eliminate the complexity of the implementation of the square root operation, the modified restoring square root architecture by Bandil and Nagar [22], the truncated or pruning scheme by Jiang *et al.,* [23], and the logarithmic-based approximation scheme by Arya *et al.,* [24] have been proposed. These schemes compromise accuracy with hardware cost [22-24]. The modified restoring array-based square root scheme by Bandil and Nagar [22] has been proposed to decrease the power consumption and area with faster computation as compared to the exact square root architecture. This scheme has achieved a simple planned exclusion and enhancement of restoring subtractor cells within the conventional array square root circuit for efficient square root execution. Moreover, an adaptive approximation approach has been achieved to perform square root operation by Jiang *et al.,* [23]. Some less important numbers of input bits are pruned effectively to cut the width of the square root. This scheme reduces the maximum error distance of the approximate square root circuit. In order to decrease the signal activity and the operator's strength, the logarithmic-based approximate scheme has been targeted by Arya *et al.,* [24] to implement the square root operation. The complex computations are converted to simple addition and shifting operations in this scheme. Although the approximations due to the transformation from binary field to logarithmic field produce some errors, the gaining of power saving and speed enhancement is significantly increased.

Thus, this paper proposes approximate square and square root calculation without multiplication or division that are correspondent with error-resilient DSP applications with less hardware resources. The large number of partial products and accumulations in the square and square root operations have been replaced by simple shifting and shifting-and-adding approach with the aid of piecewise linear logarithm and antilogarithm converters. Simple hardware implementation of the piecewise linear converters leads to significant savings of hardware cost and power consumption.

The rest of this paper is organized as follows. Section 2 presents the proposed square and square root hardware architecture design. Also, the proposed logarithm and antilogarithm converters are demonstrated in section 2. Section 3 analyzes the error and the evaluation results. VLSI hardware implementation and synthesis results of the proposed square/square root are discussed in section 4. The conclusion is demonstrated in section 5.

2. Proposed Square/Square Root Hardware Architecture Design

In this section, we present the hardware implementation of the square function, square root function, logarithm converter, and antilogarithm converter.

In general, computations in logarithmic domain are performed in three steps as shown in Figure 3: 1) take the base-2 logarithm of the input operands, 2) operations in the logarithmic arithmetic domain, and 3) take the antilogarithm of the results from 2). Among the existing approaches to the hardware implementation of logarithmic conversion, piecewise linear approximation is usually the most efficient solution.

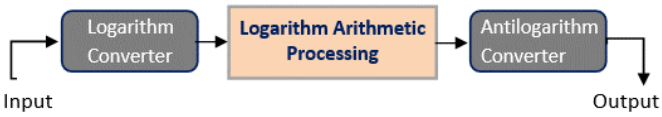

Fig. 3. Logarithm arithmetic unit architecture

Exploiting the approximate logarithm arithmetic in the implementation of square function has its significant improvement on the performance. Great saving in power, area, and delay is achieved at expenses of accuracy. The method of accomplish the complete approximate computation in logarithm domain strongly depends on the scheme that used for performing logarithm and antilogarithm converters. Several prior works have proposed efficient logarithmic and antilogarithmic converters that relay on the principle of Mitchell [28]. In the next subsection, the proposed logarithm and antilogarithm converters are presented.

2.1 Proposed Logarithm and Antilogarithm Converters Implementation Structure

The logarithm arithmetic consists mainly of logarithm converters, antilogarithm converters, and simple arithmetic unit. The logarithm converters are responsible for the transformation from binary number form to logarithm number form. While the antilogarithm converters are responsible for the transformation from logarithm number form to binary number form. Consequently, the hardware implementation of these converters has a great impact on the overall performance. The first investigation by Mitchell's [28] in the transformation method from binary number form to logarithm number form began with representing the input operand x by integer part k and fraction part f. Also, k may be named as the characteristic and demonstrates the position of the most significant bit. Thus, x can be substituted by $2^k(1+f)$. After entering the input operand x into the logarithmic converter, it is transformed to $k+\log_2(1+f)$. Mitchell approximated the transformation curve into segments and represented each segment by straight-line f neglecting the rest of the approximated fraction. Although this approximation is simple and consumes minimum hardware cost, the transformation accuracy is decreased. Based on the piecewise linear approximation $\log_2(1+f)$ may be approximated to $f+\alpha\cdot f+\beta$ in order to correct the error level. In the same way, for the transformation from logarithm number form to binary number form, the input operand x is represented by integer part k and fraction part f, where $x=k+f$. The representation of the input operand x after entering into the antilogarithm converter is $2^k 2^f$. Also, the fraction term 2^f is approximated to $\alpha\cdot f+\beta$ based on piecewise linear approximation. The transformation coefficients α and β for the logarithm or antilogarithm converters are chosen to be power of two for each segment to simplify the hardware implementation. Consequently, a shift-and-add architecture is used in the implementation of the transformation converters as will be discussed in section 4.

In order to enhance the performance of the transformation from binary number form to logarithm number form and vice versa, different prior piecewise linear approximation have been accomplished [29-37]. Several research have been approximated the transformation to logarithm domain into large number of regions to provide higher accuracy. However, as the number of regions increases, the complexity of hardware implementation increases. On the other hand, various researches have proposed the advancements of hardware in terms of power consumption reduction and speed upgrading.

The diversity of applications requires both significantly low power consumption and higher accuracy level. Therefore, two mainly categories are specified to implement the transformation converters from binary number domain to logarithmic domain or vice versa. Piecewise uniform approximation and piecewise nonuniform approximation logarithm and antilogarithm converters implementation are promising alternatives which produce a trade-off between energy consumption and accuracy. In the piecewise uniform approximation converters, the dedicated interval for the converters is divided into segments with the same size. The converters that exploit this scheme consume less hardware implementation cost and consequently less energy dissipation. On the other hand, the piecewise nonuniform approximation converters divide the dedicated interval into

segments with non-equal size. These converters optimize the accuracy level at the expense of hardware cost.

Among of the different approaches in the implementation of the transformation from binary number form to logarithm number form or vice versa, the prior research for the transformation that is found to be a compromise between accuracy and hardware cost are the works of Ellaithy *et al.,* [29,32], Kuo and Juang [30,34], Loukrakpam and Choudhury [31], and Lyu *et al.,* [33]. We study the trade-off for the transformation converters to propose the appropriate converters depending on the requirements of the applications. The power consumption has become highly important metric than chip area and accuracy. According to this demand, the proposed transformation converters has been used to implement the square operation and the square root operation.

Thus, in this paper we have proposed a new piecewise eight-segment uniform approximation converters for simpler hardware design and minimum power consumption. The proposed converters exhibit efficient power saving with good accuracy level. The approximation coefficients of each segment of the transformation converters are chosen in order to enhance the overall performance. The selection of the transformation coefficients is based on the trade-off between accuracy and hardware complexity. The transformation coefficients for each segment of the logarithm and antilogarithm converters have been chosen by the aid of a linear programming according to the algorithm demonstrated in Figure 4. The transformation coefficients of each segment are attained using MATLAB software by compare the approximation coefficients to the true amount. The achieved transformation coefficients enhance the hardware implementation with good accuracy level. Table 1 and Table 2 include the uniform eight-segment in column 1 and the proposed transformation coefficients α and β in column 2-3, for each segment, for the logarithm and antilogarithm converters, respectively. The proposed converters are able to decrease power consumption by more than 5% in logarithm converter and more than 37% in antilogarithm converter as compared to prior work with comparable accuracy level in section 4. The accuracy analysis and comparison with the previous work have been presented in the next section 3.

Algorithm: Generation of the Transformation Coefficients for 8-Segment Logarithm and Antilogarithm Converters

[Input] the actual logarithm/antilogarithm equation in the entire range
[Input] the approximation equation
[Input] the eight-subdivisions range [0,1/8,2/8,3/8,4/8,5/8,6/8,7/8,8/8]
Find the best transformation coefficients for the straight line approximation with good accuracy level for each subdivision
Try to make trade-off between the accuracy and the hardware implementation by choosing the transformation coefficients as combinations of power of two
[Output] the transformation coefficients for each subdivision

Fig. 4. Proposed algorithm for the logarithm transformation converters

Table 1
Proposed uniform eight-region logarithmic converter with its transformation coefficient

	Subdivisions	α	β x 2-14
1.	[0 , 1/8)	1 + 92/256	0
2.	[1/8 , 2/8)	1 + 55/256	296
3.	[2/8 , 3/8)	1 + 24/256	792
4.	[3/8 , 4/8)	1	1,380
5.	[4/8 , 5/8)	1 – 20/256	2,032
6.	[5/8 , 6/8)	1 – 36/256	2,668
7.	[6/8 , 7/8)	1 – 52/256	3,432
8.	[7/8 , 8/8)	1 – 64/256	4,096

Table 2
Proposed uniform eight-region antilogarithmic converter with its transformation coefficient

	Subdivisions	α	β x 2-14
1.	[0 , 1/8)	1 – 71/256	16,386
2.	[1/8 , 2/8)	1 – 54/256	16,252
3.	[2/8 , 3/8)	1 – 36/256	15,966
4.	[3/8 , 4/8)	1 – 16/256	15,489
5.	[4/8 , 5/8)	1 + 6/256	14,786
6.	[5/8 , 6/8)	1 + 30/256	13,826
7.	[6/8 , 7/8)	1 + 56/256	12,578
8.	[7/8 , 8/8)	1 + 84/256	11,008

2.2 Proposed Approximate Square/Square Root Function Implementation Structure

An optimal solution of the circuit hardware implementation of complex function in digital signal processing is the transformation from the binary number computing domain to the logarithm number computing domain by Ellaithy *et al.,* [38]. The signal activity and the strength of the operations are reduced. Consequently, the cost of hardware is reduced.

The implementation of square function (S) and square root function (SR) of input operand A can be transformed to right shift or left shift in the logarithm domain, respectively, according to

$$\begin{aligned} S &= A^2 \\ log_2 S &= 2 \cdot log_2 A \\ S &= 2^{[2 \cdot log_2 A]} \end{aligned} \tag{1}$$

$$\begin{aligned} SR &= \sqrt{A} = A^{1/2} \\ log_2 SR &= 2^{-1} \cdot log_2 A \\ SR &= 2^{[2^{-1} \cdot log_2 A]} \end{aligned} \tag{2}$$

Eq. (1) confirms that the implementation of square function (S) is replaced by right shifting operation after transferring to logarithm domain. In the same way, the implementation of square root function (SR) is replaced by left shifting operation after converting to logarithm domain according to Eq. (2). The implementation of the square and square root operations in logarithm domain is much simpler as shown in Figure 5. After the transformation of the input operand from the binary number form to logarithm number form by using the logarithm converter, only shifting right (RSH) or shifting left (LSH) operation is performed in the logarithm domain for the square operation

or square root operation, respectively. The next step is the transformation from logarithm number form to binary number form by using antilogarithm converter to obtain the final result.

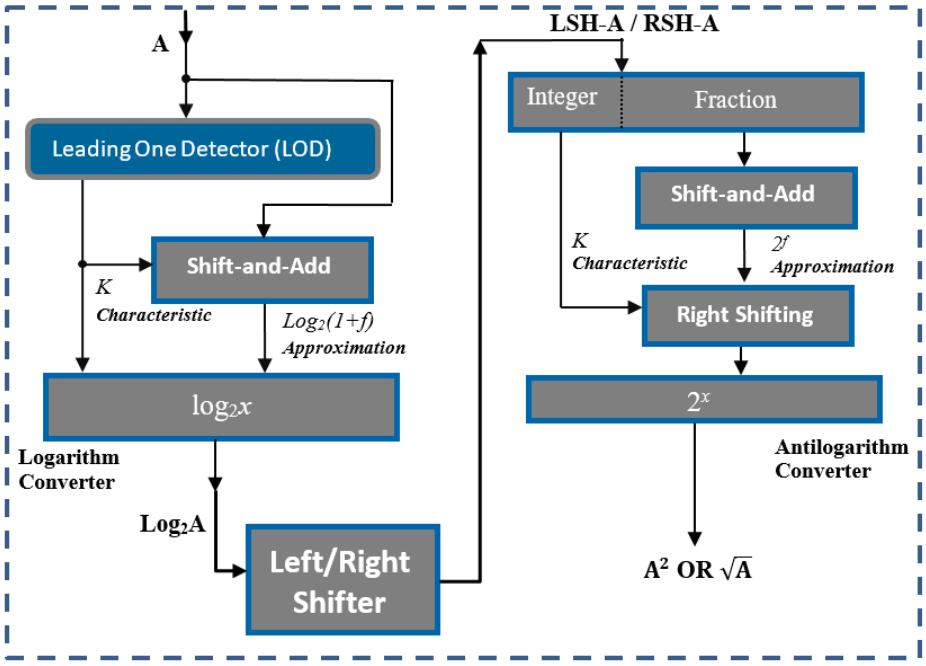

Fig. 5. Proposed architecture of square/square root operation

The hardware implementation of the logarithm converter is composed of leading one detector (LOD) block and sum of shifts block. To compute the characteristic value k, detecting the position of leading one bit, the LOD block is utilized. The circuit implementation of the LOD is based on the scheme by Abed and Siferd [39]. The bits following the leading one bit is considered the fraction value f. The shifts and additions block are utilized to compute the approximated fractional value of $\log_2(1+f)$. The logarithm computation results are generated by concatenating the characteristic value and the approximated fractional part. Likewise, the implementation of the antilogarithm converter is composed of sum of shifts block to compute the approximated fraction value 2^f. The final transformation result is generated by the final shifting by the characteristic value k.

A step-by-step example demonstrating the above process for employing the proposed architecture of implementing the square and square root operations in the logarithm domain of input operand A = 3 is presented in Figure 6. The first block in the logarithm converter is the LOD. As shown in Figure 6, the position of the leading 1 bit is one (k=1), thus the fraction bits are the remaining bit after the leading one bit. After that simple add and shift operations are performed according to the value of the fraction bits. For the example in Figure 6, f = 0.5, this belongs to the fifth segment of the proposed uniform eight-region logarithmic converter. According to the transformation coefficient in the fifth segment, the logarithm of the fractional part is approximated to [1001 0101 1100]. The last stage is to combine the characteristic and the approximated fractional part. As presented in Figure 6, the logarithm of the input operand A is [0001.1001 0101 1100]. For the implementation of the square process, a right shifter (RSH) is employed while a left shifter (LSH) is employed for the square root implementation as shown in Figure 5. The last step is the transformation from the logarithm domain to the binary domain through the antilogarithm converters. The characteristic and fraction bits are determined as demonstrated in Figure 6. The error analysis of the proposed logarithm

converter, antilogarithm converter, and square and square root operations are presented in the next section.

Let A = 3 = (0011)

Step1: Logarithm Converter:

- $k = 1, f = [1000\ 0000\ 0000]$
- $\log_2(1+f) = [1001\ 0101\ 1100]$
- $\log_2 A = k + \log_2(1+f) = [0001.1001\ 0101\ 1100]$

Step2: Shifting Operation:

- For Square Function: Right Shift - RSH = (0011.0010 1011 1000)
- For Square Root Function: Left Shift - LSH = (0000.1100 1010 1110)

Step3: Antilogarithm Converter:

For Square Function

- $k = 3, f = [0010\ 1011\ 1000]$
- $2^f = [1.0010\ 0000\ 0100]$
- $A^2 = 2^k \cdot 2^f = [1001.0000\ 0010\ 0000] = 9.0078125$

For Square Root Function

- $k = 0, f = [1100\ 1010\ 1110]$
- $2^f = [1.1011\ 1100\ 0011]$
- $\sqrt{A} = 2^k \cdot 2^f = [0001.1011\ 1100\ 0011] = 1.735107422$

Fig. 6. Example of proposed algorithm of square/square root operation

3. Error Analysis and Evaluation Results

MATLAB software is used to evaluate the error. For logarithm and antilogarithm converter, the absolute error and relative error is defined as:

$$\text{Absolute Error} = \text{approximated value} - \text{true value} \tag{3}$$

$$\text{Relative Error}(\%) = \left[\frac{\text{approximated value} - \text{true value}}{\text{true value}}\right] \times 100 \tag{4}$$

Figure 7 and Figure 8 demonstrate the absolute error graphs for the proposed piecewise eight-segment uniform approximation logarithm and antilogarithm converters according to the transformation coefficient listed in Table 1 and Table 2, respectively.

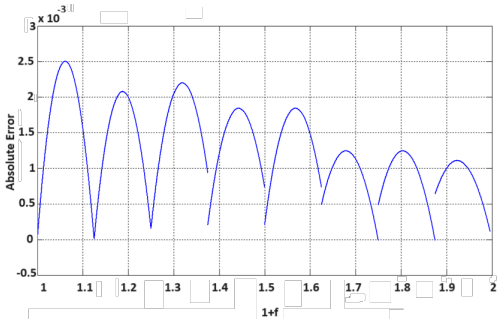

Fig. 7. Absolute error for the proposed logarithm converter

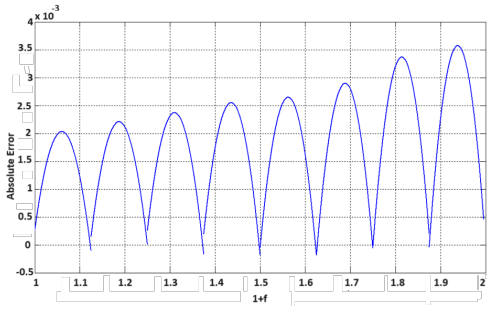

Fig. 8. Absolute error for the proposed antilogarithm converter

Also, Table 3 and Table 4 include the achieved results and prior work for comparison. Several efforts have been attained to improve the accuracy of transformation converters [29-34]. However, certain inaccuracies are occurred due to approximations. Consequently, some error level will produce within implementing the logarithm based square and square root operation. The proposed transformation converters are selected to trade off hardware complexity and accuracy. The proposed piecewise uniform approximation logarithm converter achieves less error as compared to previous work. Also, accuracy is improved by at least 17% for the piecewise uniform antilogarithm converter as compared to prior work by Ellaithy *et al.,* [29]. Using the proposed transformation converters in the implementation of the square and square root operations, significant hardware complexity reduction is accomplished.

Conventionally, the maximum absolute error (MAE), the relative error distance (RED), the mean relative error distance (MRED), and the mean squared error (MSE) are usually utilized to evaluate the accuracy of the investigated designs. The maximum absolute error (MAE) is computed as:

$$\mathrm{MAE} = \max\left[\text{Exact square value} - \text{approximated square value}\right] \tag{5}$$

The relative error distance (RED) is defined as:

$$\mathrm{RED} = \frac{\text{Exact square value} - \text{approximated square value}}{\text{Exact square value}} \tag{6}$$

Moreover, the mean relative error distance (MRED), is denoted as follows:

$$\mathrm{MRED} = \frac{1}{\mathrm{N}}\sum_{\mathrm{i}=1}^{\mathrm{N}}\left[\frac{\text{approximated value} - \text{true value}}{\text{true value}}\right] \tag{7}$$

The mean squared error (MSE) is expressed as [40]:

$$\mathrm{MSE} = \frac{1}{\mathrm{N}}\sum_{\mathrm{i}=1}^{\mathrm{N}}\left[\frac{\text{approximated value} - \text{true value}}{\text{true value}}\right]^2 \tag{8}$$

Where N is the number of all possible input combinations. The average conversion error of the binary number to logarithmic number square/square root operation is 0.04. The error percentage resulting from approximation square/square root is less than this one, at 0.167 by Petra *et al.,* [14]. A certain inaccuracy levels can be accepted for DSP applications.

Table 3
Error comparison between the proposed logarithm converter and the state-of-the-art work using Eq. (3)

Technique	Kuo and Juang [30]	Lyu *et al.,* [33]	Ellaithy *et al.,* [32]	Proposed Work
Segments	8	8	8	8
Max. Positive Error	0.92×10^{-3}	2.501×10^{-3}	--	2.5×10^{-3}
Max. Negative Error	2.0×10^{-3}	-0.0934×10^{-3}	--	0
Total Error Range	2.92×10^{-3}	2.594×10^{-3}	2.47×10^{-3}	2.50×10^{-3}

Table 4
Error comparison between the proposed antilogarithm converter and the state-of-the-art work using Eq. (4)

Technique	Ellaithy *et al.,* [29]	Loukrakpam and Choudhury [31]	Kuo and Juang [34]	Proposed Work

Segments	8	8	11	8
Max. Positive Error (%)	+0.07528	0.0898	1.7327	+0.09747
Max. Negative Error (%)	-0.05138	-0.0244	0.0992	-0.00668
Total Error Range (%)	0.12666	0.0898	1.8319	0.10415

4. VLSI Hardware Implementation and Synthesis Results

In this section, we synthesize the proposed work with the CMOS 90-nm, 1.0 V supply voltage standard cell library using Synopsys Design Compiler and evaluate the area and power overheads. Two types of comparisons are presented as follows. First, the proposed transformation converters are compared with prior logarithm and antilogarithm converters in Table 5 and Table 6. Second, the logarithm based approximation square/square root operations are compared with the exact square radix-4 implementation and prior approximation work in Table 7 [2,10,11,14,16].

Table 5
VLSI hardware implementation and synthesis results comparison between the proposed logarithm converter and the state-of-the-art work

Approach	Kuo and Juang [30]	Lyu *et al.,* [33]	Ellaithy *et al.,* [32]	Proposed Work
Process (nm)	180	90	90	90
Power (μW)	--	864.3	660.49	628.181
Area (μm^2)	27975.02	3611	8600.48	8976
Delay (ns)	3.5	0.95	1.77	1.53

Table 6
VLSI hardware implementation and synthesis results comparison between the proposed antilogarithm converter and the state-of-the-art work

Approach	Ellaithy *et al.,* [29]	Loukrakpam and Choudhury [31]	Kuo and Juang [34]	Proposed Work
Process (nm)	90	65	180	90
Power (μW)	153.343	271	--	170.477
Area (μm^2)	6098.736	701.28	2807	7309
Delay (ns)	1.41	1.34	1.4	1.39

Table 5 compares the proposed uniform eight-segment piecewise logarithm converter hardware performance with the previous piecewise transformation converters [30,32,33]. The proposed logarithm converter achieves less power consumption and high accuracy.

Moreover, Table 6 includes the comparison results of the proposed uniform eight-segment antilogarithm converter with prior work [29,31,34]. As compared with prior work, the proposed antilogarithm converter achieves high power saving and lower error level.

The synthesized hardware of 32-bit and 16-bit proposed square/square root scheme is compared with the hardware characteristics of the ASIC-based 32-bit and 16-bit exact squarer scheme and different approximation prior work [2,10,11,14,16,23]. Table 7 demonstrates the synthesized hardware results in terms of area, power, and delay for the exact implementation scheme, five different implementation schemes, and the proposed scheme. Our proposed square scheme incurs approximately 3 ns delay.

Table 7
VLSI hardware implementation and synthesis results comparison between the proposed square/square root process and the state-of-the-art work

Approach		Accurate Square Radix4		Loukrakpam *et al.,* [10]	Reddy *et al.,* [11]	Shao and Li [16]	Ansari *et al.,* [2]	Jiang *et al.,* [23]	Petra *et al.,* [14]	Proposed Work	
Process (nm)		90		65	45	90	28	28	90	90	
Number of Bits		32-Bit	16-Bit	32-Bit	16	16	16	16	16	32-Bit	16-Bit
Power (mW)		1.865	0.89	0.87	0.5206	0.4816	0.0586	0.0626	0.701	0.735	0.367
Area (μm^2)		28492.9	5623	3105.60	1649.4	2090	247.57	182.8	3566	15215	5130
Delay (ns)		8.06	4.15	2.58	1.382	2.45	0.81	4.09	1.11	3.09	1.4
Error Metrics	MAE	0		0.03125	0.0396	--	--	0.24	0.74	0.0195	
	RED	0		3.125	3.96	--	--	--	--	1.95	
	MRED	0		0.42	--	--	0.0297	0.023	--	0.04	
	MSE	0		--	--	0.11	--	--	0.091	0.0016	

The proposed square root operation exhibits a power efficiency 2.5 times higher than the exact one. As demonstrates in Table 7, the proposed architecture occupies an area of 15215 μm^2 and consumes a power of 0.735 mW with average error of 0.04.

Since they targeted different technologies, we scaled the original speed, area, and power performances according to rules provided by Stillmaker and Baas [41]. We have normalized the results for different technology nodes. Scaling down from 90 nm CMOS technology, 1.0 V supply voltage standard cell library to 65 nm CMOS technology, 1.0 V supply voltage standard cell library is attained according to Stillmaker and Baas [41]. The power consumption of the proposed square scheme after scaling down from 90 nm process to 65 nm process becomes (0.647x0.735) 0.4755 mW. Also, the delay in 65 nm process turns into (0.7554x3.09) 2.33 ns. Moreover, the area is reduced to (0.53x15215) 8063.95 μm^2. The hardware implementation results of the proposed scheme achieve saving in power by up to 45% as compared to approximate squaring hardware scheme by Loukrakpam *et al.,* [10]. Also, more than 9.6% saving in delay is attained with higher accuracy.

A straightforward optimization is accomplished to the logarithm and antilogarithm converters to enhance the accuracy level. Consequently, overall improvement is obtained when performing the square/square root operations. Finally, the proposed scheme has a lower power overhead.

In order to examine the efficiency of the proposed techniques in real advanced applications, some works were presented by Bandil and Nagar [22], Jiang *et al.,* [23], Ansari *et al.,* [2], and Avramović *et al.,* [1] introduce an error-resilient application. Some applications in different domains include image processing and computer vision applications specifically edge detection computation by Bandil and Nagar [22] and envelope detection in ultrasound imaging by Jiang *et al.,* [23]. Also, applications in amplitude demodulation to obtain the Euclidian distances between the exact demodulation signal and the approximated technique were examined by Ansari *et al.,* [2], and Avramović *et al.,* [1]. We will include a dedicated application that confirms the efficiency of the proposed designs in an extension work. Also, the extension work will involve designing and implementing the proposed approximate architecture using downsizing of used CMOS technology.

5. Conclusions

A novel uniform eight-region piecewise transformation logarithm and antilogarithm converters are proposed. A shift-and-add algorithm is exploited for hardware implementation of the transformation converters. Up to 27% reduction of power is achieved for the logarithm converter.

Also, not less than 17% high accuracy is attained for the antilogarithm converter. The proposed converters are used in the hardware implementation of the square/square root functions. The complex square function and square root function hardware implementation are simplified to right and left shifting by adopting the proposed converters. The obtained results show power saving by up to 60%, 45%, when compared with the exact square scheme, and prior approximate work by Loukrakpam *et al.,* [10].

Acknowledgement
This research was funded by Electronics Research Institute, Ministry of Scientific Research, Egypt.

References

[1] Avramović, Aleksej, Zdenka Babić, Dušan Raič, Drago Strle, and Patricio Bulić. "An approximate logarithmic squaring circuit with error compensation for DSP applications." *Microelectronics Journal* 45, no. 3 (2014): 263-271. https://doi.org/10.1016/j.mejo.2014.01.005

[2] Ansari, Mohammad Saeed, Bruce F. Cockburn, and Jie Han. "Low-power approximate logarithmic squaring circuit design for DSP applications." *IEEE Transactions on Emerging Topics in Computing* 10, no. 1 (2020): 500-506. https://doi.org/10.1109/TETC.2020.2989699

[3] Dianov, Anton, and Alecksey Anuchin. "Fast square root calculation for control systems of power electronics." In *2020 23rd International Conference on Electrical Machines and Systems (ICEMS)*, pp. 438-443. IEEE, 2020. https://doi.org/10.23919/ICEMS50442.2020.9290816

[4] Dianov, Anton, and Aleksey Anuchin. "Review of fast square root calculation methods for fixed point microcontroller-based control systems of power electronics." *International Journal of Power Electronics and Drive Systems* 11, no. 3 (2020): 1153. https://doi.org/10.11591/ijpeds.v11.i3.pp1153-1164

[5] Dianov, Anton, Alecksey Anuchin, and Alexey Bodrov. "Fast square root calculation without division for high performance control systems of power electronics." *CES Transactions on Electrical Machines and Systems* 6, no. 2 (2022): 145-152. https://doi.org/10.30941/CESTEMS.2022.00020

[6] Salam, Syed Munimus, and Muhammad Mahbubur Rashid. "Performance Analysis of a Designed Prototype of a Motor Coupled Variable Inertia Flywheel System." *Malaysian Journal on Composites Science and Manufacturing* 12, no. 1 (2023): 62-72. https://doi.org/10.37934/mjcsm.12.1.6272

[7] Tan, Nurfarah Diana Mohd Ridzuan, Fudhail Abdul Munir, Musthafah Mohd Tahir, Herman Saputro, and Masato Mikami. "Preliminary investigation of using DBD plasma for application in micro combustors." *Journal of Advanced Research in Fluid Mechanics and Thermal Sciences* 82, no. 1 (2021): 105-112. https://doi.org/10.37934/arfmts.82.1.105112

[8] Inn, Lai Qit, A. N. Oumer, Azizuddin Abd Aziz, Januar Parlaungan Siregar, and Tezara Cionita. "Numerical Analysis of Battery Thermal Management System of Electric Vehicle." *Journal of Advanced Research in Numerical Heat Transfer* 13, no. 1 (2023): 106-114. https://doi.org/10.37934/arnht.13.1.106114

[9] Noranai, Zamri, Nurul Farahin Mohd Joharudin, Noradila Abdul Latif, Nur Liyana'Amirah Mohd Kamil, and Mohd Azahari Razali. "A Case Study on Potential Saving of Energy Consumption at Hospital Putrajaya." *Journal of Advanced Research in Fluid Mechanics and Thermal Sciences* 100, no. 2 (2022): 15-22. https://doi.org/10.37934/arfmts.100.2.1522

[10] Loukrakpam, Merin, Ch Lison Singh, and Madhuchhanda Choudhury. "Energy-efficient approximate squaring hardware for error-resilient digital systems." In *2018 IEEE Electron Devices Kolkata Conference (EDKCON)*, pp. 202-206. IEEE, 2018. https://doi.org/10.1109/EDKCON.2018.8770453

[11] Reddy, K. Manikantta, Moodabettu Harishchandra Vasantha, YB Nithin Kumar, and Devesh Dwivedi. "Design of approximate booth squarer for error-tolerant computing." *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 28, no. 5 (2020): 1230-1241. https://doi.org/10.1109/TVLSI.2020.2976131

[12] Datla, Satyendra R., Mitchell A. Thornton, and David W. Matula. "A low power high performance radix-4 approximate squaring circuit." In *2009 20th IEEE International Conference on Application-specific Systems, Architectures and Processors*, pp. 91-97. IEEE, 2009. https://doi.org/10.1109/ASAP.2009.35

[13] Banerjee, Arindam, and Debesh Kumar Das. "A New Squarer design with reduced area and delay." *IET Computers & Digital Techniques* 10, no. 5 (2016): 205-214. https://doi.org/10.1049/iet-cdt.2015.0170

[14] Petra, Nicola, Davide De Caro, Valeria Garofalo, Ettore Napoli, and Antonio G. M. Strollo. "Truncated squarer with minimum mean-square error." *Microelectronics Journal* 45, no. 6 (2014): 799-804. https://doi.org/10.1016/j.mejo.2014.02.018

[15] Reddy, Beechu Naresh Kumar. "Design and implementation of high performance and area efficient square architecture using Vedic Mathematics." *Analog Integrated Circuits and Signal Processing* 102, no. 3 (2020): 501-506. https://doi.org/10.1007/s10470-019-01496-w

[16] Shao, Botang, and Peng Li. "Array-based approximate arithmetic computing: A general model and applications to multiplier and squarer design." *IEEE Transactions on Circuits and Systems I: Regular Papers* 62, no. 4 (2015): 1081-1090. https://doi.org/10.1109/TCSI.2015.2388839

[17] Shao, Botang, and Peng Li. "A model for array-based approximate arithmetic computing with application to multiplier and squarer design." In *Proceedings of the 2014 International Symposium on Low Power Electronics and Design*, pp. 9-14. 2014. https://doi.org/10.1145/2627369.2627617

[18] Haritha, H., and S. R. Ramesh. "Design of an enhanced array based approximate arithmetic computing model for multipliers and squarers." In *2017 14th IEEE India Council International Conference (INDICON)*, pp. 1-5. IEEE, 2017. https://doi.org/10.1109/INDICON.2017.8487762

[19] Gillani, Ghayoor A., Muhammad Abdullah Hanif, M. Krone, Sabih H. Gerez, Muhammad Shafique, and André BJ Kokkeler. "SquASH: Approximate square-accumulate with self-healing." *IEEE Access* 6 (2018): 49112-49128. https://doi.org/10.1109/ACCESS.2018.2868036

[20] Kim, Duhwan, and Sunggu Lee. "Approximate square root circuits with low latency and power dissipation." *Electronics* 11, no. 1 (2021): 46. https://doi.org/10.3390/electronics11010046

[21] Park, In-Cheol, and Tae-Hwan Kim. "Multiplier-less and table-less linear approximation for square-related functions." *IEICE TRANSACTIONS on Information and Systems* 93, no. 11 (2010): 2979-2988. https://doi.org/10.1587/transinf.E93.D.2979

[22] Bandil, Lalit, and Bal Chand Nagar. "Modified restoring array-based power efficient approximate square root circuit and its application." *Integration* 94 (2024): 102106. https://doi.org/10.1016/j.vlsi.2023.102106

[23] Jiang, Honglan, Fabrizio Lombardi, and Jie Han. "Low-power unsigned divider and square root circuit designs using adaptive approximation." *IEEE Transactions on Computers* 68, no. 11 (2019): 1635-1646. https://doi.org/10.1109/TC.2019.2916817

[24] Arya, Neelam, Manisha Pattanaik, and G. K. Sharma. "Energy-efficient logarithmic square rooter for error-resilient applications." *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 29, no. 11 (2021): 1994-1997. https://doi.org/10.1109/TVLSI.2021.3114616

[25] Nasir, Sharifah Noha Zahirah Syed Abdul, Nurul Ain Ab Wahab, and Mohd Agos Salim Nasir. "Graphical User Interface for Solving Non-Linear Equations for Undergraduate Students." *International Journal of Advanced Research in Future Ready Learning and Education* 30, no. 1 (2023): 25-34.

[26] Eshan, Abdul Rauf, and Wah Yen Tey. "Investigation of Shape Parameter for Exponential Weight Function in Moving Least Squares Method." *Progress in Energy and Environment* 2 (2017): 17-24.

[27] Abdullah, Aslam. "Comparative Study of the Condition for Non-Oscillatory Solution of a Singularly Perturbed Problem on Uniform and Piecewise-Uniform Meshes." *CFD Letters* 12, no. 8 (2020): 108-120. https://doi.org/10.37934/cfdl.12.8.108120

[28] Mitchell, John N. "Computer multiplication and division using binary logarithms." *IRE Transactions on Electronic Computers* 4 (1962): 512-517. https://doi.org/10.1109/TEC.1962.5219391

[29] Ellaithy, Dina M., Magdy A. El-Moursy, Ghada Hamdy, Amal Zaki, and Abdelhalim Zekry. "Efficient piecewise non-uniform approximation logarithmic and antilogarithmic converters." In *2017 International Conference on Advanced Control Circuits Systems (ACCS) Systems & 2017 Intl Conf on New Paradigms in Electronics & Information Technology (PEIT)*, pp. 149-152. IEEE, 2017. https://doi.org/10.1109/ACCS-PEIT.2017.8303034

[30] Kuo, Chao-Tsung, and Tso-Bing Juang. "Design of fast logarithmic converters with high accuracy for digital camera application." *Microsystem Technologies* 24 (2018): 9-17. https://doi.org/10.1007/s00542-016-3105-y

[31] Loukrakpam, Merin, and Madhuchhanda Choudhury. "Error-aware design procedure to implement hardware-efficient antilogarithmic converters." *Circuits, Systems, and Signal Processing* 38, no. 9 (2019): 4266-4279. https://doi.org/10.1007/s00034-019-01062-9

[32] Ellaithy, Dina M., Magdy A. El-Moursy, Ghada Hamdy, Amal Zaki, and Abdelhalim Zekry. "Accurate piecewise uniform approximation logarithmic/antilogarithmic converters for GPU applications." In *2017 29th International Conference on Microelectronics (ICM)*, pp. 1-4. IEEE, 2017. https://doi.org/10.1109/ICM.2017.8268821

[33] Lyu, Fei, Zhelong Mao, Jin Zhang, Yu Wang, and Yuanyong Luo. "PWL-based architecture for the logarithmic computation of floating-point numbers." *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 29, no. 7 (2021): 1470-1474. https://doi.org/10.1109/TVLSI.2021.3081572

[34] Kuo, Chao-Tsung, and Tso-Bing Juang. "Area-efficient and highly accurate antilogarithmic converters with multiple regions of constant compensation schemes." *Microsystem Technologies* 24 (2018): 219-225. https://doi.org/10.1007/s00542-016-3238-z

[35] Loukrakpam, Merin, and Madhuchhanda Choudhury. "Error-aware design procedure to implement hardware-efficient logarithmic circuits." *IEEE Transactions on Circuits and Systems II: Express Briefs* 67, no. 5 (2020): 851-855. https://doi.org/10.1109/TCSII.2020.2979937

[36] Xiong, Botao, Yuanfeng Sui, Zhi Jia, Sicun Li, and Yuchun Chang. "Utilize the shift-and-add architecture to approximate multiple nonlinear functions." *International Journal of Circuit Theory and Applications* 49, no. 7 (2021): 2290-2297. https://doi.org/10.1002/cta.2994

[37] Dong, Hongxi, Manzhen Wang, Yuanyong Luo, Muhan Zheng, Mengyu An, Yajun Ha, and Hongbing Pan. "PLAC: Piecewise linear approximation computation for all nonlinear unary functions." *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 28, no. 9 (2020): 2014-2027. https://doi.org/10.1109/TVLSI.2020.3004602

[38] Ellaithy, Dina M., Magdy A. El-Moursy, Ghada H. Ibrahim, Amal Zaki, and Abdelhalim Zekry. "Double logarithmic arithmetic technique for GPU." In *2017 12th International Conference on Computer Engineering and Systems (ICCES)*, pp. 373-376. IEEE, 2017. https://doi.org/10.1109/ICCES.2017.8275335

[39] Abed, Khalid H., and Raymond E. Siferd. "CMOS VLSI implementation of a low-power logarithmic converter." *IEEE Transactions on Computers* 52, no. 11 (2003): 1421-1433. https://doi.org/10.1109/TC.2003.1244940

[40] Oleolo, Ibrahim, Hayati Abdullah, Maziah Mohamad, Mohammad Nazri Mohd Jaafar, and Akmal Baharain. "Model Selection for the Control of Temperature in a Centralized Air Conditioning System." *Journal of Advanced Research in Applied Mechanics* 74, no. 1 (2020): 10-20. https://doi.org/10.37934/aram.74.1.19

[41] Stillmaker, Aaron, and Bevan Baas. "Scaling equations for the accurate prediction of CMOS device performance from 180 nm to 7 nm." *Integration* 58 (2017): 74-81. https://doi.org/10.1016/j.vlsi.2017.02.002