Low-Power Approximate Logarithmic Squaring Circuit Design for DSP Applications

Mohammad Saeed Ansari, Bruce F. Cockburn, Jie Han

Abstract—The squaring function is widely used in Digital Signal Processing (DSP). There are many DSP applications with noisy inputs for which simplifying approximations of the squaring function implementation have a minor impact on the output quality, while permitting significant reductions in the hardware cost. This article proposes a Low-Error Squaring Function (LESF) and its low-power hardware implementation. Unlike the existing logarithmic squaring functions, LESF benefits from a double-sided error distribution and, consequently, error cancellation in larger calculations. LESF approximates a base-2 logarithmic function with a linear polynomial, i.e. $\log_2 f(x) \approx ax + b$. Since input b in this sum is a constant, LESF replaces the conventional full-adder with a compact specialized adder for hardware efficiency. Our simulation results show that the 16-bit LESF is 23.23% more accurate (in the mean relative error distance) than the baseline Mitchell approximate logarithmic squaring function while being 1.8$\times$ faster and 39% more energy-efficient. LESF and other logarithmic squaring functions are evaluated for the square-law detector application. LESF is shown to be more than 3$\times$ more accurate in this application (with respect to the Euclidean distance) than the next most accurate design in the literature, which uses an iterative error compensation technique.

Index Terms—squaring function, low-power, approximate arithmetic, logarithmic circuits, AM modulation.

1 Introduction

The performance of a computing system is often determined by its arithmetic modules [1]. A widely-used arithmetic operation in multimedia and Digital Signal Processing (DSP) is the squaring function [2]. DSP applications usually process noisy data from signal acquisition devices and, therefore, faster and more hardware-efficient approximate solutions can often be used at the cost of negligible quality degradation in the final results [3]. In fact, fully accurate results are not required for many applications and in those cases approximation can be beneficial due to the potential to significantly reduce design costs while still producing sufficiently accurate results [4], [5].

Most general-purpose processors can execute DSP algorithms (e.g., using a multiplier with identical inputs to implement a squaring function); however, they might not meet the latency and power consumption constraints for some battery-powered applications, such as mobile phones [6]. The logarithm operation can be used to simplify the computation of arithmetic functions, such as multiplication [7], [8] and the squaring operation [2], [3], [9]. As shown in [3], the baseline Mitchell logarithmic squaring function is 29.79% faster and 2.12$\times$ smaller than the conventional array squaring circuit, while consuming 5.93$\times$ less power.

Computations in the logarithmic domain are performed in three steps [10]: (1) take the base-2 logarithm of the input operand(s), (2) operations in the logarithmic domain, and (3) take the antilogarithm of the results from (2). Logarithmic circuits are inherently approximate designs due to the limited bit precision and the finite accuracy when computing the base-2 logarithm [11], [12]. Using either piecewise linear approximations over a finely subdivided input domain or iterative techniques can compensate for the accuracy loss when computing the base-2 logarithm [8].

Among the existing approaches to the hardware implementation of logarithmic conversion, piecewise polynomial approximation is usually the most efficient solution [10]. An early and influential piecewise polynomial approximation was proposed by Mitchell [7]. Mitchell's approximate logarithm uses a Leading-One Detector (LOD) and always underestimates the actual value. Several Mitchell-based methods have been proposed to improve the accuracy. They typically divide the power-of-two intervals into more than one region and then apply piecewise linear approximation within each region. The designs differ in the number of regions and in the piecewise linear approximation functions used in each region [10], [13]. A Nearest-One Detector (NOD) is proposed in [11] instead of the conventional LOD and, thus, it benefits from a double-sided error distribution and, consequently, the possibility of error cancellation in larger calculations.

A NOD is not used in this article; instead, we exploit the concept of up-rounding in the design of the proposed squaring function. In fact, the base-2 logarithm of the input to the squaring function is calculated by using both up-rounding and the conventional down-rounding methods. These two scenarios are then jointly considered to calculate the output of the squaring function. The proposed Low-Error Squaring Function (LESF) is developed for integers; however, it can be easily extended to operate on floating-point (FP) numbers as well. Moreover, LESF approximates a base-2 logarithm function with a linear polynomial, i.e. $\log_2 f(x) \approx ax + b$. Adding the constant b can be performed by using a simplified adder (as one of the inputs is known and fixed) instead of a general full-adder. The other contribution of this article is to propose an adder that reduces the

The authors are with the Department of Electrical and Computer Engineering, University of Alberta, Edmonton, AB, T6G 1H9, Canada (e-mail: ansari2, cockburn, jhan8@ualberta.ca).

energy consumption of the LESF.

Note that the LESF is different from a simplified logarithmic multiplier in which both inputs have identical values. In fact, it uses an approximation that is expressly developed for a squaring function.

The remainder of this article is organized as follows: Section 2 provides the required background information on Logarithmic Number Systems (LNS) and the state-of-the-art squaring functions. Section 3 describes the proposed squaring function and how it is implemented in hardware. It also elaborates on how the proposed method can be applied to FP numbers. Section 4 evaluates and compares the error and hardware performance of the proposed design with other state-of-the-art squaring functions. The square-law detector is considered in Section 5 to evaluate the performance of the LESF in a widely-used application in the field of telecommunications and to compare it quantitatively with the existing designs in the literature. Finally, Section 6 provides concluding remarks.

2 Related Work

Let $Z = z_n z_{n-1} \cdots z_1 z_0$ be the $(n+1)$-bit binary representation of a positive integer N. Without loss of generality, let k, where $k \leq n$, indicate the position of the most significant '1' in Z. Hence, N can be represented as [7]:

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

where $0 \leq x < 1$ is the binary fraction, which can be calculated as: $x = \sum_{i=k-1}^{0} 2^{i-k} z_i = 1$. Consequently, N^2 can be obtained as:

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

and the base-2 logarithm of N can be calculated as:

$$\log_2 N = k + \log_2(1+x). \tag{3}$$

Depending on how $\log_2 N$ is approximated, different approximations for N^2 can be obtained.

The Mitchell algorithm [7], the baseline method for most logarithmic multipliers and squaring functions, approximates $log_2(1+x)$ with x, which suggests the following approximation for N^2:

$$N^2 \approx \begin{cases} 2^{2k}(1+2x), & x < 0.5, \\ 2^{2k+1}(2x), & x \geq 0.5. \end{cases} \tag{4}$$

The exponent of the power of 2 in (4) gives the position of the most significant '1' in the final result, and the fraction part indicates the less significant bits.

The authors in [3] propose an approximate squaring function with error compensation. Their design is composed of a main block that approximates the squaring function with a shift operation and a carry-free subtraction. This block can then be reused for error compensation. In fact, the error term from the first calculation is used as an input to the same block and the results are added to the primary results from the first step. This iterative process continues until an acceptably small error is achieved. This technique is applicable to any logarithmic squaring function, including the LESF. However, this technique significantly increases the hardware costs, as will be discussed in Section 4.2. The authors in [3] rewrite N as $N = 2^k + (N-2^k)$ and, therefore, N^2 can be approximated as:

$$N^2 \approx 2^{2k} + (N-2^k)2^{k+1} = 2^k(2N-2^k) \tag{5}$$

Comparing the approximated value in (5) with the exact result shows that the term $(N-2^k)^2$ gives the approximation error. Similar to Mitchell's method, this approach also always underestimates the actual value of N^2.

A recent logarithmic multiplier with double-sided error distribution is proposed in [8], where the same approximation as used by Mitchell is used and, therefore, (4) would still be valid for this design. However, the hardware efficiency and the accuracy are improved, compared to Mitchell's logarithmic multiplier, by using approximate adders. Three approximate adders are considered in [8] and the Set-to-One-Adder (SOA) that sets a few least significant bits to '1' shows the best accuracy versus hardware cost trade-off. Hence, the only difference between the two designs lies in their adders. We used this design as another reference for comparison purposes.

3 Proposed Squaring Function

Here we propose an approximation for the squaring function which, unlike the existing approaches, has a double-sided error distribution.

3.1 Mathematical modeling

Any positive integer N, as expressed in (1), can be also factored as:

$$N = 2^{k+1}(1-t), \tag{6}$$

where $0 < t \leq 1$. Considering (1) and (6), the base-2 logarithm of N can be expressed as:

$$\log_2 N = k + \log_2(1+x) = k + 1 + \log_2(1-t). \tag{7}$$

Hence, $\log_2 N^2 = 2\log_2 N$ can be written as the summation of the middle and right expressions in (7), as given by:

$$\log_2 N^2 = 2k + 1 + \log_2(1 + x - t - tx). \tag{8}$$

The variable t can be obtained as a function of x, i.e. $t = 0.5(1-x)$, by considering the fact that both (1) and (6) represent the same value N. Substituting the expression into (8) results in:

$$\log_2 N^2 = 2k + 1 + \log_2(0.5 + x + 0.5x^2). \tag{9}$$

The least-squares method can be used to linearly approximate $\log_2(0.5 + x + 0.5x^2)$ in (9). This method chooses the coefficients so as to minimize the summed square of residuals. We used the MATLAB *curve fitting* toolbox [14] to this end and the best least squares linear fit over $0 \leq x < 1$ is:

$$\log_2(0.5 + x + 0.5x^2) \approx 1.975x - 0.8732. \tag{10}$$

Hence the base-2 logarithm of N^2 can be approximated by replacing the $\log()$ function in (9) with (10). However, to simplify the hardware implementation, the constant 1.975 is rounded to 2, which is a simple left-shift in hardware.

The approximation in (10) will not remain the best linear fit when the coefficient 1.975 is changed to 2 and, therefore, the other coefficient needs to be adjusted to minimize the approximation error. By trying different values, we found out experimentally that 0.1268 (the constant obtained by replacing (10) in (9)) needs to be changed to $R_c = \frac{5}{128} = 0.039$ to achieve the Lowest Mean Relative Error Distance (LMRED) for the LESF. Fig. 1 shows how the accuracy of the LESF, in terms of the MRED, changes with different constant values. As shown in Fig. 1, reducing this constant from 1 improves the accuracy of the LESF. However, the minimum MRED, i.e. the maximum accuracy, is obtained when the constant in (11) lies within the range [0.035, 0.040], see the inset. Hence, we chose 0.039, which falls into this range and, as mentioned earlier, can be easily implemented in hardware.

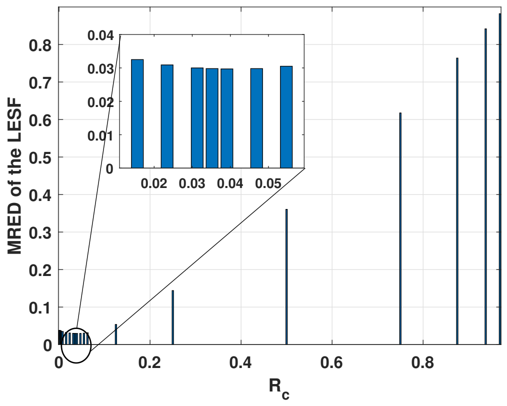


Fig. 1: LESF accuracy w.r.t. the MRED vs. R_c in (11).

The proposed squaring function is therefore given by:

$$N^2 \approx 2^{2k+1.975x+0.1268} \approx 2^{2k+2x+R_c}. \qquad (11)$$

Note that the coefficients that result in the minimum Mean Squared Error (MSE) for the approximation in (10) are 1.975 and -0.8732, according to the MATLAB *curve fitting* toolbox. However, changing the coefficients to what are used in (11) increases the MSE from 0.0026 to 0.0084. Although this increase is notable, the new MSE is still negligible. More importantly, using the modified coefficients significantly simplifies the hardware implementation and still results in a highly-accurate squaring function.

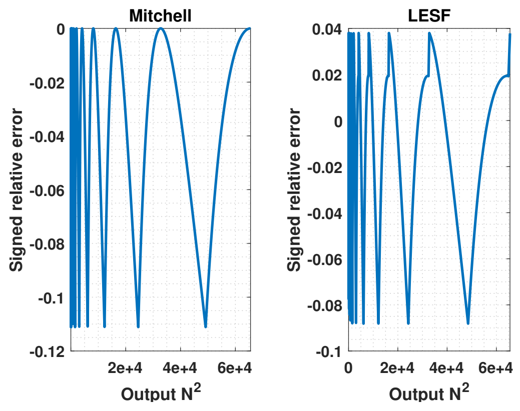


Fig. 2: Signed relative error for the 8-bit Mitchell and the LESF squaring circuits.

The signed relative error for the entire output range is plotted for the 8-bit LESF and the baseline Mitchell squaring circuits in Fig. 2. This figure shows that, unlike Mitchell, LESF has both positive and negative errors.

3.2 Hardware implementation

The form of the proposed squaring function in (11) does not imply any particular hardware implementation. It is useful to express the results in the form of (4). To do so, 2^y, where $y = 2x + R_c$, needs to be approximated. The least-squares method is used again and the best linear fit over $0 \leq y < 1$ according to the MATLAB *curve fitting* toolbox is $2^y \approx 0.9923y+0.9471$. We modified the two coefficients and implemented $2^y \approx y+1$ instead. Although this modification increases the MSE from 0.0007 to 0.003, it is still negligible. More importantly, it has a low-cost hardware implementation and still results in a highly-accurate squaring function. Finally, LESF can be represented by:

$$N^2 \approx \begin{cases} 2^{2k}(y+1), & y < 1, \\ 2^{2k+1}y, & 1 \leq y < 2, \\ 2^{2k+2}(y-1), & 2 \leq y < 3. \end{cases} \qquad (12)$$

For example for $2 \leq y < 3$ in (12), $2^y = 2^{(2+(y-2))}$; let $t = y - 2$ and, consequently, $2^y = 2^2 \times 2^t$. Since $0 \leq t < 1$, 2^y can be approximated as $2^2 \times (1+t) = 2^2 \times (y-1)$.

Fig. 3 shows the block diagram of the n-bit LESF. As shown in Fig. 3, the first step is to find the k and x values from the n-bit input I. We used the conventional Mitchell approach to find these two parameters. The n-bit output of the LOD is used as the input to the priority encoder (PE), which stores the value of k in $\log_2 n$ bits, R_k, which is then used to generate $2k$ in (12) using the Left-Shift unit in Fig. 3. The value of x, on the other hand, is obtained by performing the logical *XOR* between the original n-bit input I and the LOD's output. Since the output of the LOD uses a one-hot representation, performing the *XOR* operation does the subtraction [3]. The result of this subtraction needs to be represented in the $(n-1)$-bit register R_x [7]. If I_i, where $i \in \{0, 1, 2, \cdots, n\}$ is the most significant '1' in I, then $R_x = I_{i-1}I_{i-2}\cdots I_1 I_0$. Hence, zeros should be padded to the least significant bits of R_x for $i < n$, e.g. $R_x = 00100000$ for $I = 00001001$. This is done by using multiplexers ($MuxBank$ in Fig. 3) that use the output of the PE and then append the proper number of zeros accordingly.

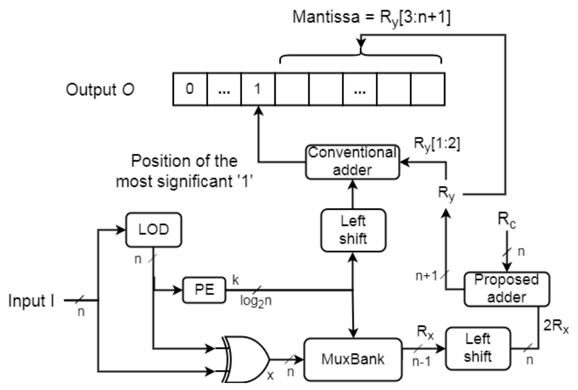


Fig. 3: Architecture of the LESF.

The next step is to calculate y, which can be done by adding the constant R_c to the shifted version of R_x, $2R_x$, which has the value $2x$. Since $2R_x$ is an n-bit number, R_c needs to be represented as an n-bit value. The result of this addition is stored in $(n+1)$-bit register R_y. We implicitly know that R_c contains the fraction part, which lies to the right side of the radix point and, therefore, it can be represented as $R_c = 000010100\cdots00$. Note that the bits following the second '1' in R_c are all zeros.

Algorithm 1 Addition of $2R_x$ and $R_c = 0.039$

```
1: Inputs: $2R_x$ and $R_c$, Output: $R_y$
2: $R_y[9:n+1] = 2R_x[8:n]$
3: $R_y[8] = \neg 2R_x[7]$                          ▷ logical $NOT$
4: $(R_y[7], c_5) = HA(2R_x[6], 2R_x[7])$  ▷ conventional HA
5: $(R_y[6], c_4) = HA(\neg 2R_x[5], c_5)$
6: $(R_y[5], c_3) = HA(2R_x[4], c_4)$
7: $(R_y[4], c_2) = HA(2R_x[3], c_3)$
8: $(R_y[3], c_1) = HA(2R_x[2], c_2)$
9: $(R_y[2], R_y[1]) = HA(2R_x[1], c_1)$
```

Since R_c in the addition $R_y = 2R_x + R_c$ is a constant, a conventional adder can be replaced by a simpler design. The required function is specified below in Algorithm 1. In Algorithm 1, the two signals in pairs $(R_y[j+2], c_j)$, where $j \in \{1, 2, \cdots, 5\}$, denote the sum and $carry_{out}$ signals of a conventional half-adder (HA), respectively. According to Algorithm 1, more savings can be obtained by increasing n. In fact, the second step shows that no calculation is required from the 8^{th} bit down toward the least significant bit, i.e. the $(n+1)^{th}$ bit. Note that since this addition is done for the fraction part of the result, index p has the weight of 2^{-p} and, therefore, indexing starts from 1 (the most significant bit) and goes up to $n+1$.

The extra two bits in R_y compared to R_x are used to handle the three conditions in (12). As shown in (12), y represents the fractional part of the result and, thus, needs to be smaller than 1. Hence, adding two extra bits to the left side of the radix point lets us track the value of y and compare it to the conditions given in (12). Finally, considering that $y < 3$ (as shown in (12)), the first two most significant bits in R_y, i.e. $R_y[1]$ and $R_y[2]$ can be 0, 1, or 2, implements the conditions in (12). Based on the position of the most significant '1', which is determined by the output of the conventional adder (a.k.a. the exponent) in Fig. 3, three cases can occur:

- The exponent is so small that there are not enough bit positions to store the $(n-1)$ bits of R_y. In this case, the less significant bits of R_y are discarded.
- The exponent is such that there are just $(n-1)$ bits left in O. In this case, the $(n-1)$ bits of R_y will exactly fit into the available bits in O, see Fig. 3.
- The exponent is too big to fit into the $(n-1)$ available free bits in O. In this case, after fitting the $(n-1)$ bit of R_y, the other bits are filled with zeros.

For further hardware savings, we used the PE proposed in [15]. This design exploits the fact that the output of the LOD uses a one-hot representation and, therefore, the conventional PE can be simplified. Regarding the LOD, the conventional LOD in [13] is used for all of the designs.

Note that LESF can be used as a more accurate baseline design instead of the Mitchell design. What is more, the existing techniques in the literature for improving the accuracy of the Mitchell design (e.g., the iterative technique in [3]) are also applicable to the proposed design.

3.3 Extension to FP numbers

The LESF can be easily extended to handle FP numbers as well as integers. Fig. 4 shows the architecture of the LESF for 16-bit (a.k.a. half precision) FP numbers. The three main fields for a 16-bit FP number $A = a_{15}a_{14}\cdots a_1 a_0$ in IEEE Standard 754 are: (1) the sign bit ($s = a_{15}$), (2) the biased exponent $e = a_{14}a_{13}\cdots a_{10}$ with the constant bias 15, and (3) the normalized mantissa $m = a_9 a_8 \cdots a_0$, which lies on the right side of the radix point.

Since the position k of the leading-one is given in the exponent field, the LOD and PE in Fig. 3(a) can be removed. On the other hand, the fraction part x is already provided in the mantissa field and, consequently, the XOR operation and the $MuxBank$ would no longer be necessary.

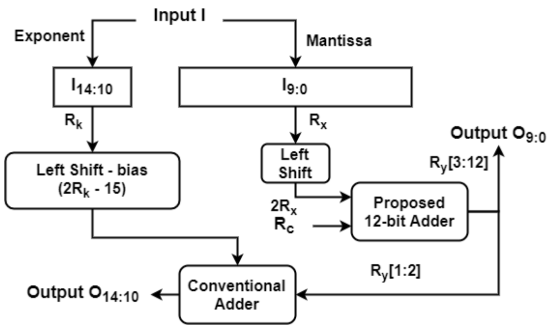


Fig. 4: Architecture of the LESF for FP numbers.

4 Performance Evaluation

The designs in [7] and [8] are logarithmic multipliers, and not squaring circuits. We used the approximation methods in these two references and simplified their hardware implementation (e.g. by removing one of the two LODs in a logarithmic multiplier as there is only one input to a squaring circuit) to create comparable squaring circuits.

Note that there are other types of squaring functions in the literature, such as [2], [16]. Basically, any multiplier design can be simplified and used as a squaring circuit. However, only logarithmic designs are considered in this work. The performance of the squaring functions is evaluated below using both accuracy and hardware metrics.

4.1 Accuracy metrics

All of the considered designs were implemented in MATLAB and their accuracy was evaluated over the entire 16-bit unsigned input domain, i.e. from 0 to 65535. Table 1 reports the accuracy metrics for the proposed LESF and other logarithmic squaring functions in the literature. The MRED and the Average Error (AE, average of the signed error distance) were then calculated.

The results in Table 1 show that the approximate squaring function in [3] with one step of error correction, Sq. Fnc.-1, is the most accurate design, with respect to the MRED.

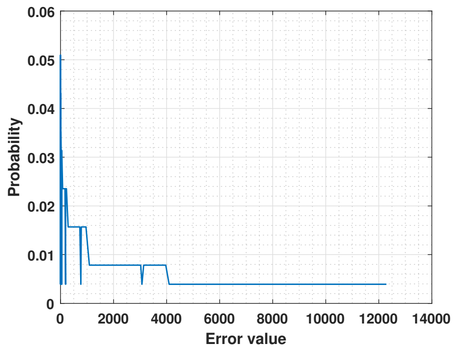
However, using iterative steps for error compensation can be applied to other designs as well at the cost of significant additional hardware, see Section 4.2. Hence, Sq. Fnc.-1 aside, LESF is the most accurate design, being 21.39% more accurate than the next most accurate ALM-SOA-9 squaring function. With respect to the AE, LESF seems to be the best design, due to its double-sided error distribution.

TABLE 1: Error metrics of five different logarithmic squaring functions.

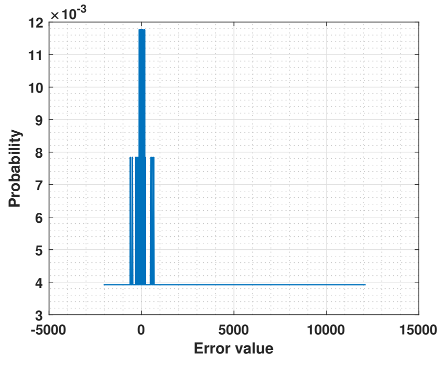
Squaring Function	AE	MRED
Mitchell [7]	5.09e+7	0.0384
ALM-SOA-9 [8]	4.87e+7	0.0374
Sq. Fnc.-0. [3]	2.05e+8	0.1137
Sq. Fnc.-1. [3]	2.91e+7	0.0149
LESF	1.44e+7	0.0297

Regarding the ALM-SOA squaring function, we tried different numbers of approximation bits in the SOA adder and we found that 9 bits produced the lowest MRED.

We also extracted the Probability Density Function (PDF) of the approximation error for Mitchell and the LESF squaring circuits and plotted them in Fig. 5. As shown in Fig. 5(a), the Mitchell squaring circuit has a one-sided error distribution, while the double-sided error distribution of the LESF is clearly shown in Fig. 5(b).



(a) Mitchell [7].



(b) LESF.

Fig. 5: Error PDF of the the 8-bit squaring functions.

4.2 Hardware metrics

All of the designs were implemented in VHDL and then synthesized using the Synopsys Design Compiler for ST Micro's 28-nm CMOS process.

The hardware measurements for three key metrics, area, critical path delay, and power consumption, are given in Table 2. As shown in this table, LESF is the fastest design and the third-best design with respect to the area and power consumption. However, it is the most energy-efficient design, with respect to the PDP and it also achieves the best energy-accuracy trade-off, i.e. the lowest PDP-MRED product. As mentioned earlier, iterative techniques significantly increase the hardware cost and this is well reflected in the results of the Sq. Fnc.-1.

TABLE 2: Hardware metrics of five different logarithmic squaring designs.

Squaring Function	Power (mW)	Delay (nS)	Area (μm^2)	Normalized PDP×MRED
Mitchell [7]	5.24e-2	1.49	184.41	0.28
ALM-SOA-9 [8]	4.33e-2	1.37	144.43	0.22
Sq. Fnc.-0. [3]	8.55e-2	1.07	264.22	1
Sq. Fnc.-1. [3]	1.95e-1	2.86	562.55	0.79
LESF	5.86e-2	0.81	247.57	0.13

5 Application: Square-Law Detector

A radio signal must be modulated to be able to carry information efficiently over a band-limited channel, such as audio information for broadcasting [17]. For example, Amplitude Modulation (AM) is a common modulation technique that is widely used in telecommunications [17].

Let $m(t)$ be any arbitrary message signal (we used a square-wave signal at 50 Hz) and let $c(t) = A_c cos(2\pi f_c t)$ be the carrier signal, where A_c and f_c denote the amplitude and frequency of the carrier signal, respectively. The AM modulated signal $s(t)$ can be expressed as:

$$s(t) = \left[1 + k_a m(t)\right] c(t). \tag{13}$$

where the amplitude sensitivity k_a is a constant such that $k_a m(t) \ll 1$ [17]. We will set $k_a m(t) = 0.01$.

MATLAB was used to generate the message signal $m(t)$, the carrier signal $c(t)$ with $f_c = 1\,\text{KHz}$, and the modulated message $s(t)$. Once the modulated signal $s(t)$ is calculated, it is transmitted over an ideal communication channel. At the receiver, the amplitude-modulated signal $s(t)$ can be demodulated using the square-law detector [17]. Signal $s^2(t)$ can be expressed as:

$$s^2(t) = A_c^2 \left[1 + k_a m(t)\right]^2 \left[\frac{1 + cos(4\pi f_c t)}{2}\right]. \tag{14}$$

After dropping the DC terms, $s^2(t)$ is passed through a low-pass filter which removes the high-frequency term $cos(4\pi f_c t)$ and outputs a replica of the message $m(t)$. The MATLAB *lowpass* function is used with the pass-band frequency $f_{pass} = 150\,\text{Hz}$ and a sampling frequency of $f_s = 10\,\text{KHz}$. With these inputs, the *lowpass* function generates a Finite Impulse Response (FIR) filter of order 48. Note that the exact squaring function in (14) was replaced with the LESF and other logarithmic squaring functions.

Fig. 6 shows the demodulated messages $m'(t)$, according to which the LESF produces the closest waveform to the waveform generated by using the exact squaring function.

Sq. Fnc.-1 is the second most accurate design. Finally, the waveforms generated by using the Mitchell and the ALM-SOA-9 squaring functions seem to be worse than the other two designs.

To numerically compare the performance of the squaring functions, the Euclidean distances between the exact demodulated signal and those obtained by using logarithmic squaring functions were calculated and are reported in Table 3. The Euclidean distance $E_{A,B}$ between the two signals A and B measures the straight-line distance between two points in A and B and can be calculated as [18]:

$$E_{A,B} = \sqrt{\sum_{i=1}^{S}(A_i - B_i)^2}, \tag{15}$$

where S is the number of sample points in the two signals and A_i and B_i denote the samples of the two signals A and B, respectively.

According to the results in Table 3, the demodulated signal using the LESF is 67.19% closer to the demodulated signal using an exact squaring function compared to the second best design, Sq. Fnc.-1, which is consistent with the results in Fig. 6. In the figure, the LESF waveform partially overlies the exact waveform.

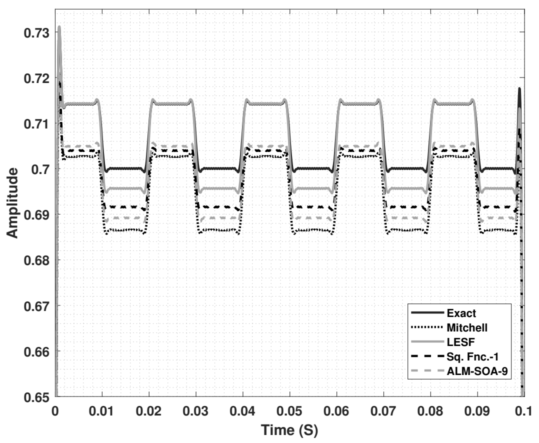


Fig. 6: Comparison of the demodulated signal $m'(t)$ with exact and logarithmic squaring functions.

TABLE 3: Euclidean distance w.r.t. exact squaring demodulation of different logarithmic squaring functions.

Squaring Function	Euclidean distance
Mitchell [7]	0.3961
ALM-SOA-9 [8]	0.3185
Sq. Fnc.-0. [3]	2.0441
Sq. Fnc.-1. [3]	0.2960
LESF	0.0971

6 Conclusion

Logarithmic squaring functions convert squaring into only shift and addition operations, thus significantly reducing the hardware implementation cost. This work proposes a low-error squaring function, LESF, that outperforms the state-of-the-art designs in the literature. LESF is the most hardware efficient design, consuming the least amount of power and while being the fastest design. LESF is also more accurate than the existing designs, in terms of the MRED, except for Sq. Fnc.-1, which uses an iterative error compensation technique. However, this technique can be used to increase the accuracy of any logarithmic squaring function with extra hardware cost. Finally, the LESF and other logarithmic squaring functions were evaluated in a real-world application, the square-law detector. LESF was shown to be almost 3× more accurate than the second most accurate design, Sq. Fnc.-1, with respect to the Euclidean distance metric.

References

[1] M. S. Ansari, H. Jiang, B. F. Cockburn, and J. Han, "Low-power approximate multipliers using encoded partial products and approximate compressors," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 8, no. 3, pp. 404–416, 2018.

[2] M. H. Sheu and S. H. Lin, "Fast compensative design approach for the approximate squaring function," *IEEE Journal of Solid-state Circuits*, vol. 37, no. 1, pp. 95–97, 2002.

[3] A. Avramović, Z. Babić, D. Raič, D. Strle, and P. Bulić, "An approximate logarithmic squaring circuit with error compensation for DSP applications," *Microelectronics Journal*, vol. 45, no. 3, pp. 263–271, 2014.

[4] H. Jiang, F. J. Santiago, M. S. Ansari, L. Liu, B. F. Cockburn, F. Lombardi, and J. Han, "Characterizing approximate adders and multipliers optimized under different design constraints," *Proceedings of the Great Lakes Symposium on VLSI*, pp. 393–398, 2019.

[5] M. S. Ansari, V. Mrazek, B. F. Cockburn, L. Sekanina, Z. Vasicek, and J. Han, "Improving the accuracy and hardware efficiency of neural networks using approximate multipliers," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 28, no. 2, pp. 317–328, 2020.

[6] I. Verbauwhede, P. Schaumont, C. Piguet, and B. Kienhuis, "Architectures and design techniques for energy efficient embedded DSP and multimedia processing," *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 20 988–20 993, 2014.

[7] J. N. Mitchell, "Computer multiplication and division using binary logarithms," *IRE Transactions on Electronic Computers*, no. 4, pp. 512–517, 1962.

[8] W. Liu, J. Xu, D. Wang, C. Wang, P. Montuschi, and F. Lombardi, "Design and evaluation of approximate logarithmic multipliers for low power error-tolerant applications," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 65, no. 9, pp. 2856–2868, 2018.

[9] M. Bilal, S. Masud, and S. Athar, "FPGA design for statistics-inspired approximate sum-of-squared-error computation in multimedia applications," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 59, no. 8, pp. 506–510, 2012.

[10] J. Y. L. Low and C. C. Jong, "Unified Mitchell-based approximation for efficient logarithmic conversion circuit," *IEEE Transactions on Computers*, vol. 64, no. 6, pp. 1783–1797, 2015.

[11] M. S. Ansari, B. F. Cockburn, and J. Han, "A hardware-efficient logarithmic multiplier with improved accuracy," *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 922–925, 2019.

[12] S. Gandhi, M. S. Ansari, B. F. Cockburn, and J. Han, "Approximate leading one detector design for a hardware-efficient Mitchell multiplier," *IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)*, pp. 205–208, 2019.

[13] K. H. Abed and R. E. Siferd, "CMOS VLSI implementation of a low-power logarithmic converter," *IEEE Transactions on Computers*, vol. 52, no. 11, pp. 1421–1433, 2003.

[14] MathWorks, "MATLAB curve fitting toolbox," *Available Online: https://www.mathworks.com/help/curvefit/curve-fitting.html*, 2019.

[15] M. S. Kim, A. A. Del Barrio, R. Hermida, and N. Bagherzadeh, "Low-power implementation of Mitchell's approximate logarithmic multiplication for convolutional neural networks," *23rd Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 617–622, 2018.

[16] J. P. Langlois and D. Al-Khalili, "Carry-free approximate squaring functions with $\mathcal{O}(n)$ complexity and $\mathcal{O}(1)$ delay," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 53, no. 5, pp. 374–378, 2006.

[17] H. Taub and D. L. Schilling, *Principles of Communication Systems.* McGraw-Hill Higher Education, New York, NY, 1986.

[18] H. Anton, *Elementary Linear Algebra*. John Wiley & Sons, Ltd, Hoboken, NJ, 2019.