electronics

MDPI

Article

A Piecewise Linear Mitchell Algorithm-Based Approximate Multiplier

Hao Liu[1], Mingjiang Wang [1], Longxin Yao [1] and Ming Liu [2,*]

[1] Faculty of Electronics and Information Engineering, Harbin Institute of Technology, Shenzhen 518055, China; gabriel.carson.law@outlook.com (H.L.); mjwang@hit.edu.cn (M.W.); 19B952005@hit.edu.cn (L.Y.)
[2] Sino-German School, Shenzhen Institute of Information Technology, Shenzhen 518055, China
* Correspondence: lm_hit_1986@126.com

Abstract: In the field of integrated circuits, the computational cost has always been a crucial design metric. In recent years, with the continuous development in the field of computing, the requirements for computation have been growing rapidly. Reducing the computational cost and improving computational efficiency have become the key issues in the field. There are many error-tolerant applications in the multimedia field where approximate computing techniques can be applied to improve computational efficiency and reduce computational costs at the cost of acceptable computational errors. This paper proposed a piecewise linear Mitchell algorithm based on Mitchell logarithmic approximation multiplication algorithm. Additionally, the Pwl-Mit multiplier is designed according to the improved algorithm combined with the data truncation technique. The proposed approximate multiplier has better statistical performance compared with the state-of-the-art multipliers. The design is simulated and synthesized at the TSMC 65 nm process, and its reliability is verified using discrete cosine transform (DCT) transform.

Keywords: approximate computing; multiplier; hardware design

Citation: Liu, H.; Wang, M.; Yao, L.; Liu, M. A Piecewise Linear Mitchell Algorithm-Based Approximate Multiplier. *Electronics* **2022**, *11*, 1913. https://doi.org/10.3390/electronics11121913

Academic Editor: Paris Kitsos

Received: 31 May 2022
Accepted: 18 June 2022
Published: 20 June 2022

1. Introduction

In the field of integrated circuit design, the computational cost has been one of the critical design criteria, especially in power-sensitive areas such as embedded systems and sensors. Rising performance demands will soon outpace the growth of resource budgets [1]. Over-allocation of resources alone will not solve the challenges that await the computing industry in the near future. Researchers have made great efforts in algorithm and hardware design to improve computational efficiency and reduce computational costs. Improving computational efficiency while ensuring circuit performance has become a hot research topic in academia and industry.

With the prevalence of mobile computing and embedded systems, many applications that exhibit inherent error tolerance in computing, various studies have shown that some highly computationally intensive applications (e.g., multimedia processing and machine learning) are error-tolerant [2,3]. Error-tolerant applications are approximately classified as follows:

- Due to the limited perceptual ability of humans, there are many error-tolerant scenarios in fields related to human interaction.
- The input data itself is noisy, such as sensor monitoring data, and the environment itself has considerable noise interference, so a completely accurate computation process is unnecessary.
- The problem itself has no precise solution. A typical example is artificial neural networks, where the boundary function itself has no exact solution. Because deep neural networks have many parameters, they have considerable error tolerance and are well suited for optimization using approximate computation techniques.

Approximate computation techniques aim to reduce the gap between computational requirements and available processing resources. The term "approximate" means that the results are not necessarily precise but are within tolerable error and can consume fewer resources than the exact solution.

Approximate computing of circuit design is an emerging circuit design method that results in an approximation with errors, which allows circuit structures to be simplified, thereby improving circuit performance, reducing circuit power consumption, and saving circuit area. An approximate circuit can be used in many error-tolerant computing systems where the computation does not require a completely accurate result. However, as an emerging design paradigm, the approximate circuit is still in the initial stages of its development. One of the many issues that need to be addressed to bring approximate computing to the mainstream is the modeling and analysis of accuracy. This is because over-approximation can produce unacceptable quality loss [4,5].

Approximate circuit design can be performed at multiple levels, including circuit-level, system-level, and software-level. Among circuit-level designs, the design of approximate arithmetic operation units has received a lot of attention due to its importance in many computing applications. Arithmetic units play an important role in most digital systems, often determining the operating frequency of the whole system and greatly influencing the system's power performance. Therefore, one of the keys to the design of approximate computational circuits is approximate arithmetic units. Multiplication is a crucial component, and by reducing the computational cost of multiplication, the efficiency of the operation can be greatly improved. Therefore, many designs of approximate multipliers have been proposed. The main contributions of this paper are as follows.

1. A piecewise linear Mitchell algorithm is proposed to improve the statistical performance of the algorithm at the cost of acceptable computational cost.
2. A high-performance approximate multiplier circuit structure is demonstrated.
3. The performance of the proposed multiplier is evaluated by various metrics and compared with the existing state-of-the-art approximate multipliers for analysis.

The rest of this paper is organized as follows: Section 2 discusses the related work and summarizes the existing approximate multipliers. An improved approximate multiplier algorithm based on Mitchell's logarithmic algorithm is proposed in Section 3. The circuit structure of the approximate multiplier based on the improved algorithm is presented in Section 4. The proposed approximate multiplier circuit's performance evaluation and error analysis will be presented in Section 5. In Section 6, the approximate computational multiplier is tested for application to verify its usability. Finally, conclusions are presented in Section 7.

2. Related Work

In this section, related work in approximate computational multipliers will be briefly introduced.

Ref. [6] proposed a minimization bias approximate multiplier (MBM) based on the Mitchell multiplier, and the design adds a constant compensation term to the basic Mitchell algorithm, which solves the problem that the error bias is always negative in the Mitchell algorithm. In Ref. [7], the researchers proposed three novel approximate 4-2 compressors, where the number of outputs of the approximate 4-2 compressor is reduced to one and utilized in an 8-bit multiplier. Additionally, an error correction module (ECM) is proposed to facilitate the error performance of the approximate multipliers with the proposed 4-2 compressors. Ref. [8] proposed an approximate high radix encoding for generating the partial product in signed multiplication, encoding the most significant bits. The unimportant bits are approximated by rounding the high radix encoding to its nearest power of 2.

In Ref. [9], the researchers proposed a high-speed and energy-efficient approximate multiplier. The method involves rounding the operands to the nearest power of 2. This eliminates the computationally intensive part of multiplication and improves the speed and energy consumption at the cost of small errors. Ref. [10] proposed an energy-efficient

approximate multiplier based on Mitchell's logarithmic multiplication, optimized for inference on convolutional neural networks (CNNs). The authors truncated the operands in the Mitchell algorithm to propose Mitchell-w multipliers to create customizable logarithmic multipliers that further reduce energy consumption. Ref. [11] proposed a novel approximate multiplier with a dynamic range selection scheme (DRUM). This multiplier combines leading one detection as well as data truncation techniques for approximation with an unbiased error distribution. Ref. [12] proposed a novel leading one bit-based approximate (LoBA) multiplier architecture, which selects k bits from n-bit inputs $(k \leq n/4)$ based on leading one bit (LOB) and then computes the approximate product based on these small inputs. Four 4-2 compressors were proposed in Ref. [13], which can be flexibly switched between exact and approximate modes of operation. In the approximate mode, these dual-quality compressors provide higher speed and lower power consumption at the cost of lower accuracy. Ref. [14] presented a novel design that uses a modification of the previous approximate 4-2 compressor design and adds an error recovery module. Ref. [15] investigated the design of approximate redundant binary (RB) multipliers. Two approximate Booth encoders and two RB 4-2 compressors based on RB (full and half addition) are proposed for RB multipliers. Ref. [16] presented the design of an approximate 15-4 compressor using a 5-3 compressor as the basic module. Four different types of approximate 5-3 compressors were used in the 15-4 compressor to reduce power consumption and improve throughput. Ref. [17] proposed an approximate multiplier that uses a mixed radix-4 and logarithmic encoding of the input operands to generate two partial products. It uses exact radix-4 encoding to generate partial products from the three most significant bits. A tail-trimmed logarithmic approximation is used to generate partial products from the remaining least significant bits. Ref. [18] presented an approximate logarithmic multiplier with two-stage operand pruning that prioritizes area and energy consumption while maintaining acceptable accuracy. This multiplier prunes the least significant part of the input operands in the first stage and prunes the approximation of the obtained operands in the second stage.

The approximate Booth multiplier proposed by Ref. [19] is designed based on the approximate radix-4 modified Booth encoding (MBE) algorithm and the partial product array using approximate Wallace trees. Two approximate Booth encoders are proposed and analyzed for fault-tolerant computations. Ref. [20] proposed a novel approximate multiplier for high-performance DSP applications with low power consumption and a short critical path. The multiplier utilizes a newly designed approximate adder to restrict its carry propagation to the nearest neighbor for fast partial accumulation. Different levels of accuracy can be achieved by using an OR gate or the proposed approximate adder in a configurable error recovery circuit. Ref. [21] proposed an energy-efficient approximate multiplier that combines radix-4 Booth encoding and logarithmic product approximation. In addition, a datapath pruning technique is proposed and investigated to reduce the hardware complexity of the multiplier. The researchers in Ref. [22] described a technique that combines the Mitchell multiplier with a new hardware truncation scheme to form an iterative multiplier with improved accuracy and reduced area. Ref. [23] proposed a scalable approximate multiplier, called truncation and rounding-based scalable approximate multiplier (TOSAM), which reduces the number of partial products by truncating according to the leading bits of each input operand. Ref. [24] proposed an energy-efficient approximate multiplier design obtained by truncating the input operands. In this structure, n-bit multiplication operations are transformed into smaller bit-length multiplication operations and some addition and shift operations. The simple computational core makes this multiplier a scalable, low-power structure. Ref. [25] proposed a novel energy-efficient approximate multiplier design using the significance-driven logic compression (SDLC) method. This approach is based on partial product rows of its asymptotic bit importance and configurable lossy compression. Subsequently, the resulting product terms are permuted and remapped to reduce the number of product rows. Ref. [26] proposed an approximating odd multiple of radix-8 to their nearest power of 2 so that the errors complement each other. In order to

pursue the accuracy-energy balance, two approximate Booth multipliers (HLRBM1 and HLRBM2) based on hybrid low radix (HLR) are designed.

3. Piecewise Linear Mitchell Algorithm

This design improves the Mitchell fast multiplication method and designs an approximate multiplication circuit based on it. In this section, the Mitchell multiplication method and its improved algorithm piecewise linear Mitchell algorithm are introduced.

3.1. Mitchell Algorithm

Mitchell algorithm was proposed by Mitchell in 1962 [27]. It uses logarithmic transformation for the purpose of fast computation of multiplication and division.

For any binary value $N = b_k\ b_{(k-1)}\ b_{(k-2)} \ldots b_0$ can be presented as Equation (1):

$$N = \sum_{i=0}^{i=k} 2^i b_i \tag{1}$$

where b_i takes the values 0, 1. Here we assume that the highest bit b_k's value is 1. The coefficient of the highest bit can be extracted. We can get Equation (2):

$$N = 2^k(1 + \sum_{i=0}^{i=k-1} 2^{i-k} b_i) \tag{2}$$

Let:

$$x = \sum_{i=0}^{i=k-1} 2^{i-k} b_i \tag{3}$$

Then we have:

$$N = 2^k(1 + x) \tag{4}$$

in which $0 \leq x < 1$. Then Mitchell used the approximation as follows:

$$\log_2 N = \log_2 2^k(1 + x) = k + \log_2 (1 + x) \approx k + x \tag{5}$$

If we use $\widehat{\log_2}$ to represent the approximate logarithmic operation, we have

$$\widehat{\log_2} P = \widehat{\log_2}(N_1 \cdot N_2) = k_1 + k_2 + x_1 + x_2 \tag{6}$$

where P is the product of N_1 and N_2. To get P, the approximate inverse transformation will be taken to the Equation (5). Since the limit $0 \leq x < 1$ in Mitchell approximate logarithmic operation, it is necessary to discuss $x_1 + x_2$ when taking the inverse transformation. The result is shown in Equation (7).

$$\begin{cases} \widehat{\log_2} P = k_1 + k_2 + x_1 + x_2 & 0 \leq x_1 + x_2 < 1 \\ \widehat{\log_2} P = 1 + k_1 + k_2 + x_1 + x_2 - 1 & 1 \leq x_1 + x_2 < 2 \end{cases} \tag{7}$$

Combining Equations (5) and (7) we get Equation (8):

$$\begin{cases} P = 2^{k_1+k_2}(x_1 + x_2 + 1) & 0 \leq x_1 + x_2 < 1 \\ P = 2^{k_1+k_2+1}(x_1 + x_2) & 1 \leq x_1 + x_2 < 2 \end{cases} \tag{8}$$

3.2. Piecewise Linear Mitchell Algorithm

In Mitchell's algorithm, the key approximation techniques used are as in Equation (9):

$$\log_2 (1 + x) \approx x \tag{9}$$

This approximation is simple and straightforward. It reduces the computational complexity to a great extent but also loses considerable computational accuracy. If a suitable approximation function can be chosen, better computational performance can be obtained at a small computational cost. According to the above view. We make a piecewise linear approximation to $\log_2(1+x)$.

Consider the general case, for the function $f(x)$ assuming the existence of segmented linear functions:

$$y = \begin{cases} b_0 x + a_0 & u_0 \le x \le u_1 \\ b_1 x + a_1 & u_1 \le x \le u_2 \\ \cdots & \\ b_n x + a_n & u_n \le x \le u_{n+1} \end{cases} \tag{10}$$

We can construct the cost function:

$$F(a_0, a_1, \cdots, a_n, b_0, b_1, \cdots, b_n, u_0, u_1, \cdots, u_{n+1}) = \sum_{i=0}^{n} \int_{u_i}^{u_{i+1}} (f(x) - a_i - b_i x)^2 dx \tag{11}$$

When the minimum value of F is obtained, the corresponding parameter is the approximation function we need. To simplify the problem. We assume that u is a series of fixed value. Find the partial derivatives for a, b respectively:

$$\frac{\partial F}{\partial a_i} = -2 \int_{u_i}^{u_{i+1}} f(x) dx + 2a_i(u_{i+1} - u_i) + b_i\left(u_{i+1}^2 - u_i^2\right) = 0 \tag{12}$$

$$\frac{\partial F}{\partial b_i} = -2 \int_{u_i}^{u_{i+1}} x f(x) dx + a_i\left(u_{i+1}^2 - u_i^2\right) + \frac{2}{3} b_i\left(u_{i+1}^3 - u_i^3\right) = 0 \tag{13}$$

where $i = 0, 1, 2, 3, \cdots, n$. Then we get the solution:

$$a_i = \frac{3}{(u_{i+1} - u_i)^3} \{\frac{4}{3}(u_{i+1}^2 + u_{i+1}u_i + u_i^2) I(u_{i+1}, u_i) - 2(u_{i+1} + u_i) J(u_{i+1}, u_i)\} \tag{14}$$

$$b_i = \frac{6}{(u_{i+1} - u_i)^3} \{2J(u_{i+1}, u_i) - (u_{i+1} + u_i) I(u_{i+1}, u_i)\} \tag{15}$$

in which:

$$\begin{aligned} I(u_{i+1}, u_i) &= \textstyle\int_{u_i}^{u_{i+1}} f(x) dx \\ J(u_{i+1}, u_i) &= \textstyle\int_{u_i}^{u_{i+1}} x f(x) dx \end{aligned} \tag{16}$$

To simplify the partition condition. We take the values of u as $\{0, 0.5, 1\}$. Combining the above equation, we can solve the approximate function of $f(x) = log_2(1+x)$ as:

$$p(x) = \begin{cases} 0.02 + 1.16x & 0 \le x < 0.5 \\ 0.18 + 0.83x & 0.5 \le x < 1 \end{cases} \tag{17}$$

Similarly for the inverse transformation $g(x) = 2^x - 1$, the corresponding approximate linear function can be solved as follows.

$$q(x) = \begin{cases} -0.01 + 0.83x & 0 \le x < 0.5 \\ -0.19 + 1.17x & 0.5 \le x < 1 \end{cases} \tag{18}$$

Considering the hardware computational cost. We make a second approximation to the equations.

$$p(x) = \begin{cases} 1.25x & 0 \le x < 0.5 \\ 0.25 + 0.75x & 0.5 \le x < 1 \end{cases} \tag{19}$$

$$q(x) = \begin{cases} 0.75x & 0 \leq x < 0.5 \\ -0.25 + 1.25x & 0.5 \leq x < 1 \end{cases} \tag{20}$$

Figure 1 shows a comparison of the proposed approximation function with the Mitchell approximation. The solid line in the figure represents the exact logarithmic and anti-logarithmic transformation function, while the dashed line represents the approximation function in Mitchell's original method (red) and the piecewise linear approximation function proposed above.

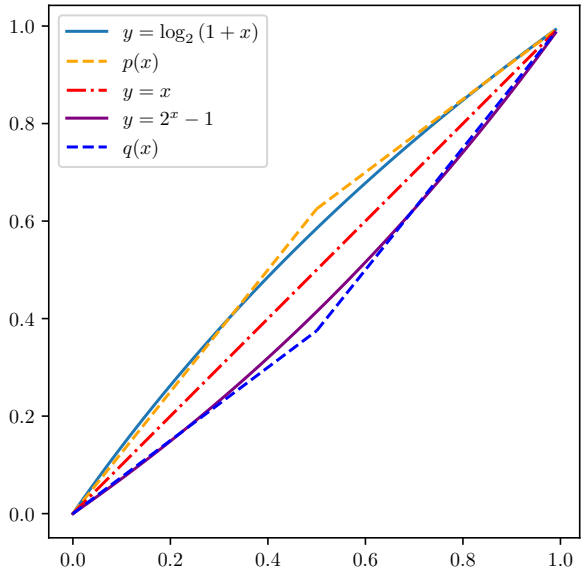
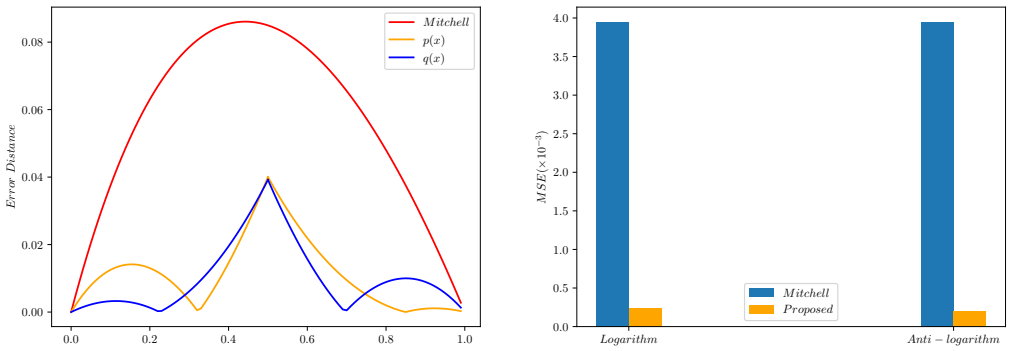


Figure 1. Logarithmic function and its different approximations.

We have compared the error metrics of the Pwl-Mit algorithm with the Mitchell algorithm, and the results are presented in Figure 2. Figure 2a compares the error distance of the two algorithms. Due to the symmetry of the logarithmic and anti-logarithmic transformations about the line $y = x$ in the standard Mitchell algorithm, the two processes have exactly the same error distance distribution, which is indicated by the red line in Figure 2a. The result shows that the Pwl-Mit algorithm has a better error distance distribution. Figure 2b compares the MSE (mean squared error) of the two approximation schemes. Similarly, the standard Mitchell algorithm has exactly the same MSE in both processes, and the comparison results show that the MSE of the proposed algorithm is only 6.09% of the standard Mitchell algorithm in the logarithmic transformation and 5.20% of the standard Mitchell algorithm in the anti-logarithmic transformation.



(**a**) (**b**)

Figure 2. Error metrics comparison of Pwl-Mit and standard Mitchell algorithm. (**a**) Error distance distribution comparison. (**b**) Mean squared error comparison.

Obviously, the new approximation function can fit the logarithmic function better. More accurate computational results are obtained. The piecewise linear Mitchell approximation algorithm is as Equation (21).

$$P = \begin{cases} 2^{k_1+k_2}\{q(p(x_1)+p(x_2))+1\} & 0 \le p(x_1)+p(x_2) < 1 \\ 2^{k_1+k_2+1}\{q(p(x_1)+p(x_2)-1)+1\} & 1 \le p(x_1)+p(x_2) < 2 \end{cases} \tag{21}$$

4. Pwl-Mit Multiplier Structure

This section describes the computational flow and hardware structure of the Pwl-Mit approximate multiplier. The computational flow of the Pwl-Mit multiplier can be divided into the following steps as shown in Figure 3.

1. The leading one bit is detected for the two input data. And the shift operation is performed so that the highest bit is the highest weighted 1 in the input data. Two registers are also used to store the shift count k.
2. To further reduce the computational cost, we use data truncation here to cut the input data, and send the truncated data to the linear transformation module.
3. The value after the shift counter and the result of linear transformation $p(x)$ are spliced to obtain an approximate logarithmic code. Perform the same operation on the second operand to obtain the approximate logarithm code as well.
4. Take out the sum result and linearly transform it by $q(x)$, send the result to the logarithmic decoder, and perform the shift operation to obtain the final result.

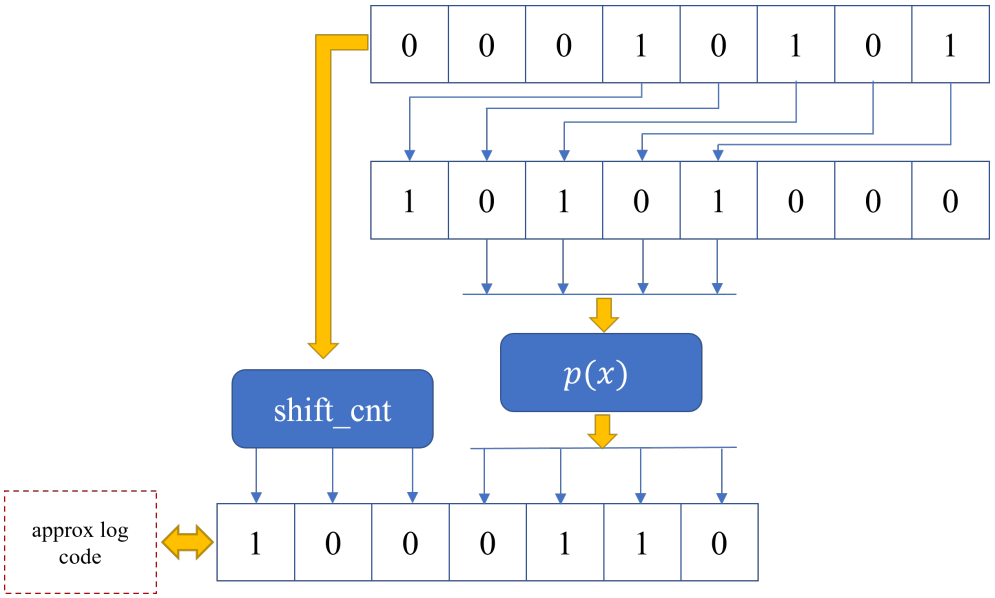
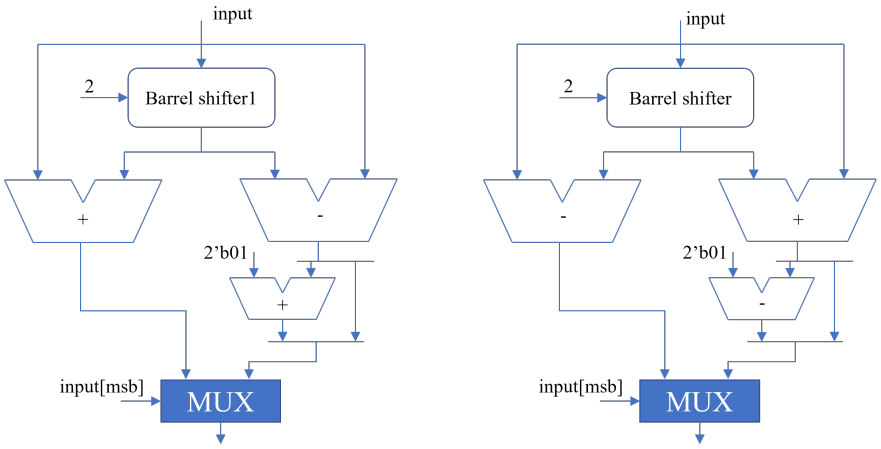


Figure 3. Pwl-Mit algorithm approximate logarithm code generation.

Figure 4a shows the structure of $p(x)$. Firstly, the input data is shifted to obtain 0.25 times of input, and then it is transferred to the adder and subtracter to obtain 1.25 and 0.75 times the value, respectively. Since the highest bit weight of the data is fixed, the summation of the constant 0.25(2'b0.01) in the algorithm can be calculated using only the highest 2 bits of the data. the addition result is obtained by splicing the calculated result with the tail. The final result is determined by whether the input data is greater than 0.5. The structure of $q(x)$ is similar to that of $p(x)$. Only the adder and subtractor need to be swapped.



(**a**) (**b**)

Figure 4. Piecewise linear function modules of Pwl-Mit approximate multiplier. (**a**) Module of linear function $p(x)$. (**b**) Module of linear function $q(x)$.

Figure 5 shows the hardware structure of the linear Mitchell logarithmic multiplier. The input data op1 and op2 first enter the leading detection circuit and are shifted according to the result. Then the truncated value is fed into the linear transform module. The approximate logarithmic encoding is obtained after splicing with the shifted count. The two encodings are summed and the result is input into the logarithmic linear approximation module. Finally, the decoder deals with the input data to obtain the final calculation result.

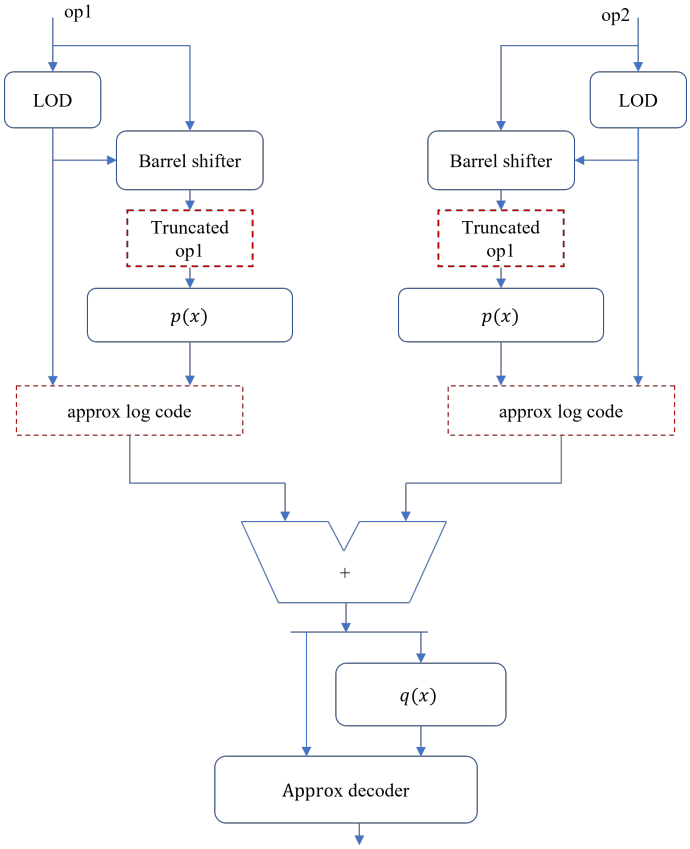


Figure 5. Pwl-Mit algorithm approximate multiplier top level hardware structure.

5. Simulation and Analysis

To verify the performance of the proposed linear approximate Mitchell multiplier. The error characteristics of the proposed linear approximate multiplier are compared with several state-of-the-art approximate multipliers. These multipliers include MBM, Mitch-w, and standard Mitchell multipliers. 8-bit and 16-bit versions of these approximate unsigned multipliers are implemented in Verilog HDL.

Model construction and simulation were developed using python language. For the 8-bit design, full coverage tests were used, and for the 16-bit design, we used 100,000 sets of random data for testing. In this section, the error performance and the synthesis results of the design are shown and analyzed.

The error performance is very critical in the approximation calculation and determines the reliability of the approximate design. The error performance of the proposed design will be illustrated by comparing it with similar methods such as MBM and Mitch-w multiplier. A variety of error metrics will be used in this paper to test and evaluate the error performance of the design. These include acceptable rate (AR), mean relative error distance (MRED), normalized mean error distance (NMED), and peak error (PE).

The error rate (ER) is a very basic metric for approximate adders. In combination with other metrics, the stability of the overall design can be assessed, among which the widely used one is the acceptability rate. AR indicates the tolerance to error, i.e., any result with a relative error distance less than a certain value can be considered as the correct result. Table 1 shows the AR under different thresholds for the 8-bit data set test.

Table 1. Acceptable probability comparison

	≤1%	≤2%	≤5%	≤10%	≤15%	≤20%
Mitchell	22.42	36.09	66.95	97.45	100	100
MBM	20.91	45.76	84.61	100	100	100
Pwl-Mit4	6.57	13.18	37.84	80.85	97.69	100
Pwl-Mit6	43.26	71.83	97.79	100	100	100
Mitch-w4	1.67	2.89	12.00	46.85	84.54	99.06
Mitch-w6	8.73	20.83	56.68	93.10	100	100

In the table, the Pwl-Mit multiplier and Mitch-w multiplier are tested for parameters 4 and 6, respectively. It is clear that the proposed Pwl-Mit multiplier has the best AR performance and the Mitchell multiplier has the second best performance while the Mitch-w multiplier has the worst AP performance.

Figure 6 shows the RED performance for the 8-bit data exhaustive test, where we set the computational parameters of both Pwl-Mit and Mitch-w multipliers to 6. It is obvious to see that Figure 6a–c have similar RED error distributions because Mitchell-w and MBM are both improved multipliers based on the original Mitchell multiplier. MBM adds a bias term to the Mitchell multiplier, so the lower half of the image is inverted. The Mitch-w multiplier takes a truncation operation to the original Mitchell multiplier so error performance is inferior. The comparison result shows that the Pwl-Mit multiplier proposed in the paper has the best RED error distribution.

Figure 7 shows the peak error comparison of several multipliers with the 8-bit data set, where the MBM multiplier has the smallest peak error with a small parameter w. As the parameter w becomes larger, the peak error of the proposed multiplier gradually decreases, and when w = 7, Pwl-Mit has the best peak error performance.

MED stands for mean error distance and is the average of the absolute values of the differences between the approximate and true results. This is a very intuitive and basic metric. However, since its value does not correlate greatly with the width of the input data, there is no way to truly reflect the accuracy performance of the computational performance.

Therefore, in this paper, NMED and MRED are used as the main error metrics. If we use D to denote the maximum error distance (ED), then NMED is calculated as follows.

$$\text{NMED} = \frac{\text{MED}}{D} = \frac{1}{n}\sum_{i=0}^{n}\frac{|\text{ED}_i|}{D} \tag{22}$$

The mean relative error distance (MRED) is the average of the relative errors, which better reflects the confidence level of the measurement. If we use S to represent the standard result, it is calculated as follows.

$$\text{MRED} = \frac{1}{n}\sum_{i=0}^{n}\frac{|\text{ED}_i|}{S_i} \tag{23}$$

To evaluate the actual circuit consumption, we synthesized the design using Synposys' design complier tool at the TSMC 65 nm process. The performance comparison is shown in Table 2.

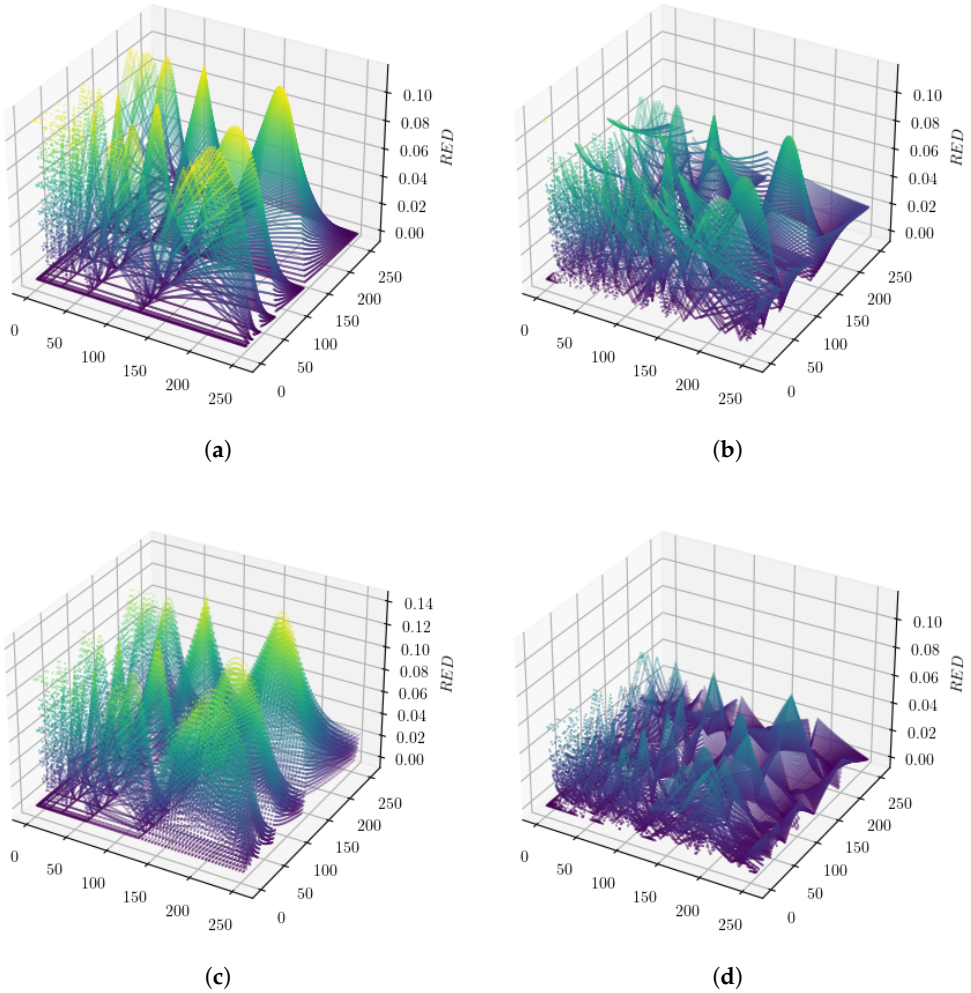


Figure 6. RED distribution comparison. (**a**) Original Mitchell multiplier RED distribution. (**b**) MBM RED distribution. (**c**) Mitchell-w6 multiplier RED distribution. (**d**) Pwl-Mit multiplier RED distribution.

From the table, we can see that our design has the best statistical performance including MRED and NMED. Mitch-w is the simplest and therefore occupies the least circuit area and power consumption. MBM has no adjustable parameters, but its statistical performance is significantly better than that of Mitch-w but weaker than that of the proposed design.

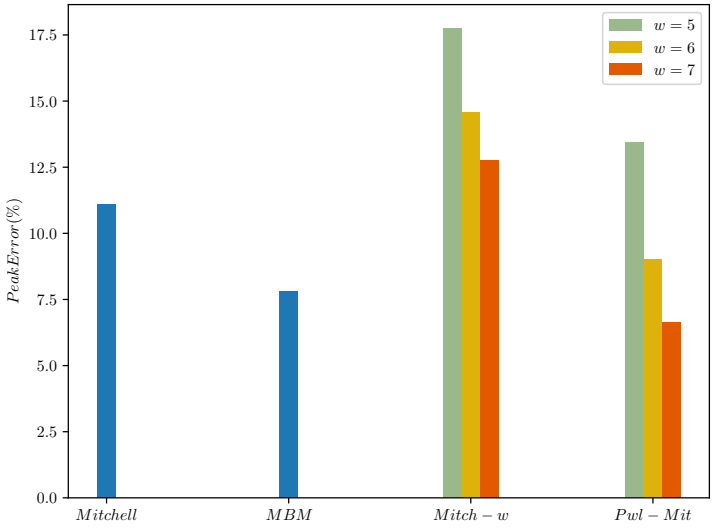


Figure 7. Peak error comparison.

Table 2. Design metrics comparison for proposed Pwl-Mit multiplier

		MRED (%)	NMED (%)	Area (μm^2)	Power (μW)	Delay (ns)
Mitch-w	w = 5	7.90	18.18	976.0	95.32	2.61
	w = 6	5.89	16.83	1018.6	103.66	2.73
	w = 7	4.87	16.46	1043.2	112.52	2.85
Pwl-Mit	w = 5	4.00	15.67	1052.8	109.70	2.84
	w = 6	2.12	14.43	1228.4	156.02	3.06
	w = 7	1.39	11.23	1131.6	190.89	3.20
Mitchell	N/A	3.85	14.75	1463.2	193.35	3.23
MBM	N/A	2.58	13.89	1631.6	208.75	3.35

6. Application Performance Test

To verify the practical working performance of the approximate multiplier proposed in this paper. We choose the discrete cosine transform as the verification. The evaluation metric used is the peak signal-to-noise ratio.

$$\mathrm{PSNR} = 10\log_{10}\frac{Max_I^2}{\mathrm{MSE}} = 20\log_{10} Max_I - 10\log_{10}\mathrm{MSE} \tag{24}$$

In order to verify the usability of the proposed multiplier, classical images from image processing were chosen as the test dataset. To reduce the computational cost and improve the computational efficiency, we divide the input image into 8*8 sub-blocks. We use a DCT transform followed by an iDCT transform to get the target result. The final transformed results are compared with the original images. The results are shown in Table 3.

Figure 8 takes one of the images of a cameraman as an example to show the results of the transformation of several multipliers. Among them, the MBM multiplier has the worst PSNR performance as the image brightness changes significantly, and the Mitch-w multiplier has the most obvious block edge. The Pwl-Mit multiplier proposed in this paper has the best visual effect as well as PSNR.

Table 3. Peak signal-to-noise ratio comparison result

Figure	Mitchell	MBM	Mitch-w8	Pwl-Mit8
cameraman	29.91	17.70	26.64	30.93
house	29.42	18.33	28.67	33.51
jetplane	26.38	15.62	23.88	28.90
lake	28.36	17.69	25.49	29.97
lena	30.28	17.55	26.04	30.90
livingroom	31.31	17.81	26.44	31.20
mandril	29.26	16.48	24.89	29.70
peppers	29.12	17.06	25.19	29.92
Average	29.26	17.28	25.91	30.63

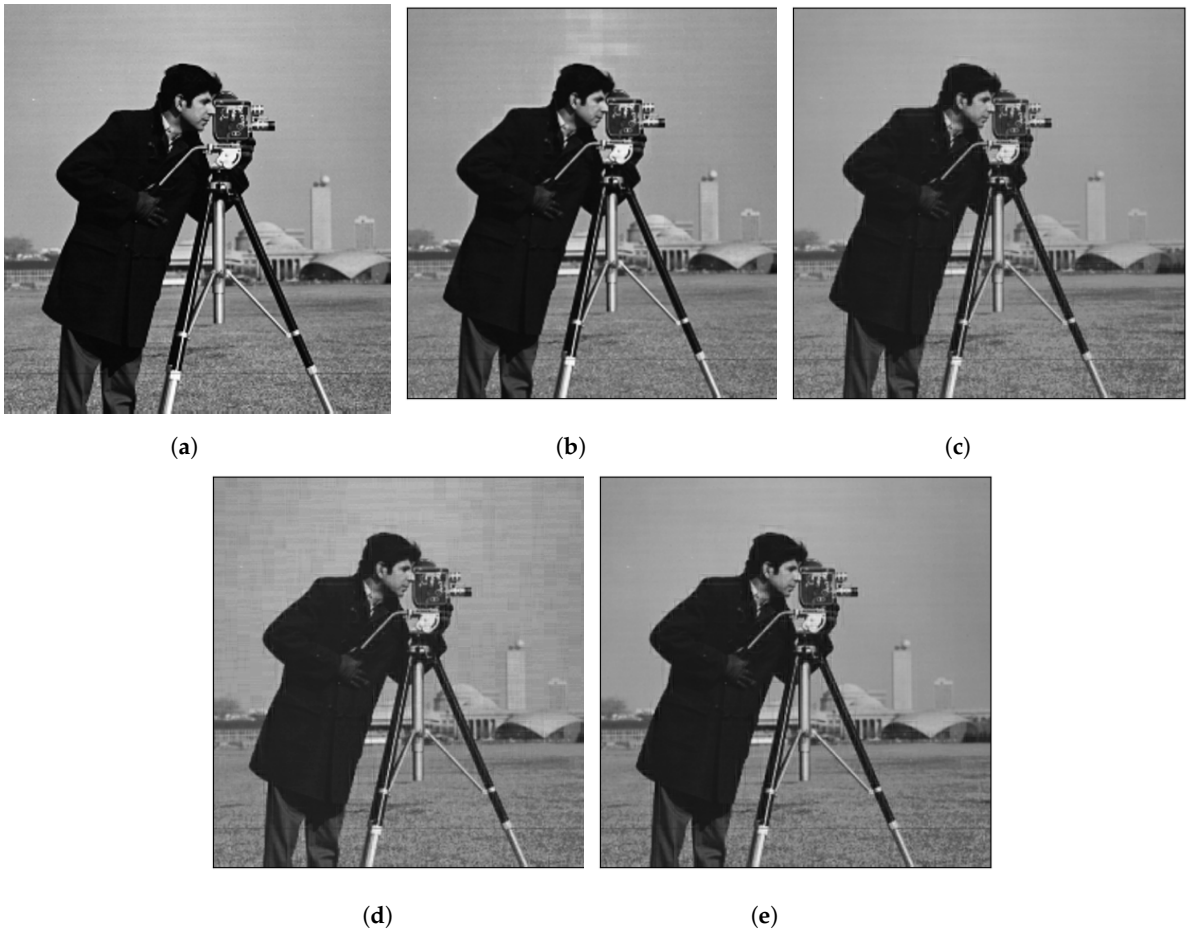


Figure 8. DCT and iDCT PSNR comparison. (**a**) Original. (**b**) Mitchell, PSNR = 29.91. (**c**) MBM, PSNR = 17.70. (**d**) Mitch-w8, PSNR = 26.64. (**e**) Pwl-Mit, PSNR = 30.74.

7. Conclusions

In this paper, we propose the piecewise linear Mitchell algorithm. It is based on the Mitchell approximate multiplication algorithm and improves the computational accuracy of the original algorithm at the cost of a higher computational cost. Based on the improved algorithm, we further propose the Pwl-Mit multiplier structure. The proposed multiplier structure has a tunable parameter w to adjust the balance between area/power and statistical performance. The entire design was simulated and synthesized using the TSMC 65 nm process. With the same parameters, the proposed structure achieves a 70% MRED and 30% NMED performance improvement at an 8% area cost compared to the Mitch-w, and a 65% MRED performance improvement and 25% NMED improvement compared to the standard Mitchell multiplier, while also reducing the circuit area by 23%. In comparison with the

MBM multiplier, the proposed design achieves 47% MRED performance advantage and 20% NMED performance improvement, while occupying only 69% of the area of the MBM. The proposed design also has the best AR performance and the smallest peak error among the four multipliers. Finally, the reliability of the design is verified using the DCT and iDCT transform. Compared with the other three multipliers, the Pwl-Mit transformed image has the highest PSNR and the best visual effect.

Author Contributions: Conceptualization, M.W.; Investigation, L.Y.; Methodology, H.L.; Project administration, M.L.; Writing, H.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the scientific research project at school-level (SZIIT2019KJ026) and the Basic Research Discipline Layout Project of Shenzhen under Grant 2020B1515120004, Grant JCYJ20180507182241622, and Grant JCYJ20180503182125190.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Enright Jerger, N.; San Miguel, J. Approximate Computing. *IEEE Micro* **2018**, *38*, 8–10. [CrossRef]
2. Eeckhout, L. Approximate Computing, Intelligent Computing. *IEEE Micro* **2018**, *38*, 6–7. [CrossRef]
3. Rodrigues, G.; Lima Kastensmidt, F.; Bosio, A. Survey on Approximate Computing and Its Intrinsic Fault Tolerance. *Electronics* **2020**, *9*, 557. [CrossRef]
4. Jiang, H.; Liu, C.; Liu, L.; Lombardi, F.; Han, J. A Review, Classification, and Comparative Evaluation of Approximate Arithmetic Circuits. *ACM J. Emerg. Technol. Comput. Syst.* **2017**, *13*, 60, 34. [CrossRef]
5. Mittal, S. A Survey of Techniques for Approximate Computing. *ACM Comput. Surv.* **2016**, *48*, 62, 33. [CrossRef]
6. Saadat, H.; Bokhari, H.; Parameswaran, S. Minimally Biased Multipliers for Approximate Integer and Floating-Point Multiplication. *IEEE Trans.-Comput.-Aided Des. Integr. Circuits Syst.* **2018**, *37*, 2623–2635. [CrossRef]
7. Pei, H.; Yi, X.; Zhou, H.; He, Y. Design of Ultra-Low Power Consumption Approximate 4–2 Compressors Based on the Compensation Characteristic. *IEEE Trans. Circuits Syst. II Express Briefs* **2021**, *68*, 461–465. [CrossRef]
8. Leon, V.; Zervakis, G.; Soudris, D.; Pekmestzi, K. Approximate Hybrid High Radix Encoding for Energy-Efficient Inexact Multipliers. *IEEE Trans. Very Large Scale Integr. Syst.* **2018**, *26*, 421–430. [CrossRef]
9. Zendegani, R.; Kamal, M.; Bahadori, M.; Afzali-Kusha, A.; Pedram, M. RoBA Multiplier: A Rounding-Based Approximate Multiplier for High-Speed yet Energy-Efficient Digital Signal Processing. *IEEE Trans. Very Large Scale Integr. Syst.* **2017**, *25*, 393–401. [CrossRef]
10. Kim, M.S.; Barrio, A.A.D.; Oliveira, L.T.; Hermida, R.; Bagherzadeh, N. Efficient Mitchell's Approximate Log Multipliers for Convolutional Neural Networks. *IEEE Trans. Comput.* **2019**, *68*, 660–675. [CrossRef]
11. Hashemi, S.; Bahar, R.I.; Reda, S. DRUM: A Dynamic Range Unbiased Multiplier for Approximate Applications. In Proceedings of the 2015 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), Austin, TX, USA, 2–6 November 2015; pp. 418–425. [CrossRef]
12. Garg, B.; Patel, S.K.; Dutt, S. LoBA: A Leading One Bit Based Imprecise Multiplier for Efficient Image Processing. *J. Electron. Test.* **2020**, *36*, 429–437. [CrossRef]
13. Akbari, O.; Kamal, M.; Afzali-Kusha, A.; Pedram, M. Dual-Quality 4:2 Compressors for Utilizing in Dynamic Accuracy Configurable Multipliers. *IEEE Trans. Very Large Scale Integr. Syst.* **2017**, *25*, 1352–1361. [CrossRef]
14. Ha, M.; Lee, S. Multipliers With Approximate 4–2 Compressors and Error Recovery Modules. *IEEE Embed. Syst. Lett.* **2018**, *10*, 6–9. [CrossRef]
15. Liu, W.; Cao, T.; Yin, P.; Zhu, Y.; Wang, C.; Swartzlander, E.E.; Lombardi, F. Design and Analysis of Approximate Redundant Binary Multipliers. *IEEE Trans. Comput.* **2019**, *68*, 804–819. [CrossRef]
16. Marimuthu, R.; Rezinold, Y.E.; Mallick, P.S. Design and Analysis of Multiplier Using Approximate 15-4 Compressor. *IEEE Access* **2017**, *5*, 1027–1036. [CrossRef]
17. Lotrič, U.; Pilipović, R.; Bulić, P. A Hybrid Radix-4 and Approximate Logarithmic Multiplier for Energy Efficient Image Processing. *Electronics* **2021**, *10*, 1175. [CrossRef]
18. Pilipović, R.; Bulić, P.; Lotrič, U. A Two-Stage Operand Trimming Approximate Logarithmic Multiplier. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2021**, *68*, 2535–2545. [CrossRef]
19. Liu, W.; Qian, L.; Wang, C.; Jiang, H.; Han, J.; Lombardi, F. Design of Approximate Radix-4 Booth Multipliers for Error-Tolerant Computing. *IEEE Trans. Comput.* **2017**, *66*, 1435–1441. [CrossRef]
20. Jiang, H.; Liu, C.; Lombardi, F.; Han, J. Low-Power Approximate Unsigned Multipliers With Configurable Error Recovery. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2019**, *66*, 189–202. [CrossRef]
21. Pilipović, R.; Bulić, P. On the Design of Logarithmic Multiplier Using Radix-4 Booth Encoding. *IEEE Access* **2020**, *8*, 64578–64590. [CrossRef]

22. Ahmed, S.E.; Srinivas, M.B. An Improved Logarithmic Multiplier for Media Processing. *J. Signal Process. Syst.* **2019**, *91*, 561–574. [CrossRef]
23. Vahdat, S.; Kamal, M.; Afzali-Kusha, A.; Pedram, M. TOSAM: An Energy-Efficient Truncation- and Rounding-Based Scalable Approximate Multiplier. *IEEE Trans. Very Large Scale Integr. Syst.* **2019**, *27*, 1161–1173. [CrossRef]
24. Vahdat, S.; Kamal, M.; Afzali-Kusha, A.; Pedram, M. LETAM: A Low Energy Truncation-Based Approximate Multiplier. *Comput. Electr. Eng.* **2017**, *63*, 1–17. [CrossRef]
25. Qiqieh, I.; Shafik, R.; Tarawneh, G.; Sokolov, D.; Das, S.; Yakovlev, A. Significance-Driven Logic Compression for Energy-Efficient Multiplier Design. *IEEE J. Emerg. Sel. Top. Circuits Syst.* **2018**, *8*, 417–430. [CrossRef]
26. Waris, H.; Wang, C.; Liu, W. Hybrid Low Radix Encoding-Based Approximate Booth Multipliers. *IEEE Trans. Circuits Syst. II Express Briefs* **2020**, *67*, 3367–3371. [CrossRef]
27. Mitchell, J.N. Computer Multiplication and Division Using Binary Logarithms. *IRE Trans. Electron. Comput.* **1962**, *11*, 512–517. [CrossRef]