# Low-error Fixed-Width Booth Multiplier Using Approximation Of Carry Function

Ganjikunta Ganesh Kumar[1*], Sibghatullah Inyatullah Khan[1], G Prasad Acharya[1], and Shravan Kumar S M[2]

[1] *Department of Electronics and Communication Engineering, Sreenidhi Institute of Science and Technology, Hyderabad, India*
[2] *Department of Civil Engineering, Sreenidhi Institute of Science and Technology, Hyderabad, India*
[*] *Corresponding author. E-mail: ganesh.g@sreenidhi.edu.in*

*Received: Jun. 06, 2024; Accepted: Dec. 17, 2024*

This paper introduces an innovative solution for increasing precision of fixed-width radix-4 Booth multipliers through variable error compensation functions that leverage Approximation of Carry Function (ACF). Error compensation mechanisms typically comprise two carries-ideal and base carry functions–strategically chosen to minimize mean error. We present three distinct methods-ACF-1, ACF-2, and ACF-3—each employing fixed base values with varying column information (w) and bit lengths (N). Comparative analyses against recent studies demonstrate that our proposed fixed-width Booth multiplier using ACF-1 stands out in terms of accuracy and efficiency tradeoffs.

**Keywords:** Error-compensation function; Fixed-width multiplier; Ideal and base carry functions


http://dx.doi.org/10.6180/jase.202510_28(10).0008

## 1. Introduction

The fundamental operations often employed in digital signal processing systems are Multiplication and Accumulation (MAC). These systems heavily rely on cascading multiplication operations, which can cause an expansion in the bit count needed to represent the multiplication results. To tackle this challenge, a fixed-width multiplier is necessary, where both input and output bits match. Yet, simply truncating the lower bits (direct-truncation) would result in a significant error.

To reduce the truncation error there is an approach which use all the bits available and roundoff to high accuracy (post-truncation). However, the hardware area cost in this case is very high. Therefore, some compensation techniques are introduced which reduce the truncation error while also taking care of the hardware complexity, i.e. something in-between the directtruncation and post-truncation techniques. To attain this balance in design, numerous error compensation circuits have been proposed by researchers for both Baugh Wooley multipliers [1–4] and Booth multipliers [5, 6]. The focus on Booth multipliers has been more extensive than on Baugh Wooley multipliers because of their tendency to exhibit lower truncation errors [7, 8].

In Jou et al. [7], introduced an economical compensation technique using linear regression and statistical analysis to significantly decrease mean error compared with direct-truncated Booth multiplier systems. But other error metrics often remain high due to limited estimation information from truncated portions. To overcome this limitation, in Cho et al. [8] the authors proposes a partitioning approach dividing each major and minor portion based on their impact on induced truncation error. Subsequently, in Wang et al. [9] employs this partition method to derive a simulation-based error estimation formula; however, developing compensation functions requires extensive simulations that take up a significant portion of calculation time and lead to time consuming processes. To speed up

calculations while maintaining accurate estimation, in Li et al. [10] introduces probability estimation bias (PEB).

In Chen [11], the authors use multilevel conditional probability (MLCP) to improve error performance, offering higher accuracy but taking up more area. He et al. [12] and Kumar and Sahoo [13] present solutions to improve accuracy-area trade-offs by combining conditional-probability estimation with computer simulation featuring variable error compensation; while Zhang and He [14] presents an energy efficient Booth encoded multiplier which takes advantage of conditions and expected probabilities to create energy saving multipliers with variable error compensation capabilities; while in He et al. [15], authors propose an error compensation function using probabilistic predictions obtained through Booth encoding to offer improved error performance overall.

In Aizaz and Khare [16], Truncated and Approximate Carry based Booth Multipliers (TACBM) are introduced by selective modifications on truncation factor w. This paper introduces a versatile error-compensation function utilizing ACF that seeks to minimize the mean error. Key contributions include Marimuthu et al. [17] and their approximate 15-4 compressor, which improved power and area efficiency, and Haider and Ko [18], who introduced energy-efficient multipliers optimized for deep learning systems using Booth encoding. Further, Marimuthu et al. [19] and Marimuthu and Mallick [20] explored high-speed multipliers and efficient signed multipliers for FFT architecture, highlighting innovations in compressor design. Zacharelos et al. [21] focused on approximate recursive multipliers with low-power building blocks, while Marimuthu et al. [22] and Liu et al. [23] contributed to the development of high-performance multicolumn compressors and approximate redundant binary multipliers, respectively.

In this paper, we define the ideal carry function as the theoretically optimal carry values in a multiplier circuit, which would produce the most accurate multiplication outcomes by minimizing truncation and rounding errors. In contrast, a fixed base carry refers to predetermined values used as a reference for carry calculations; these values remain constant and serve as a baseline for comparison with ideal carry values. We utilize these concepts by combining variable ideal carry functions $c_1$ and $c_2$, with fixed base carry values such as $c_{b1}$, $c_{b2}$ and $c_{b3}$. This integration of ideal and fixed base carry functions results in approximate carry functions that effectively compensate for truncation errors. Three methods based on base carry values are presented - ACF-1, ACF-2 and ACF-3 - that provide reduced mean error at higher hardware costs while the later method, ACF-1 offers optimal compromise between performance and accuracy.

Section 2 of this paper explores the fundamentals of the radix-4 Booth multiplier, providing essential mathematical formulae essential to its operation. Section 3 provides an in-depth exploration of the ACF method, with its workings and principles fully explained. Section 4 provides a thorough comparison between ACF methods and various circuits, taking into account key parameters like error metrics, area, delay, power, energy and combined parameters - providing insight into performance tradeoffs of each method. Section 5 provides the conclusion to this study by summarizing its findings and implications.

## 2. Materials and methods

### 2.1. Booth Multiplier

Booth encoding has long been used as a technique to reduce partial products. Where A represents multiplicands and B represents multipliers with length N bits respectively. When combined, their two's complement representation yields P:

$$\begin{aligned} A &= -a_{N-1}2^{N-1} + \sum_{i=0}^{N-2} a_i \cdot 2^i \\ B &= -b_{N-1}2^{N-1} + \sum_{i=0}^{N-2} b_i \cdot 2^i \\ P &= A \times B \end{aligned} \tag{1}$$

**Table 1.** Radix-4 Booth Encoder

| $b_{2i+1}$ | $b_{2i}$ | $b_{2i-1}$ | $PP_i$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | A |
| 0 | 1 | 0 | A |
| 0 | 1 | 1 | 2A |
| 1 | 0 | 0 | -2A |
| 1 | 0 | 1 | -A |
| 1 | 1 | 0 | -A |
| 1 | 1 | 1 | 0 |

A radix-4 Booth encoding involves mapping three concatenated inputs $b_{2i+1}$, $b_{2i}$, $b_{2i-1}$ into $PP_i$ as described in Table 1, where N is even. Following encoding, the partial product array for an 8x8 multiplier comprises N/2 rows as detailed by Table 2a. By combining encoder information from Table 1 with partial product data from Table 2a.

The partial product array can be divided into two distinct segments, the main part (MP) and truncation part (TP), with the latter further subdivided into major term ($TP_{mj}$) and minor term ($TP_{mi}$) parts, such that P can be reformulated accordingly:

$$P = MP + TP \tag{2}$$

**Table 2a.** An eight-bit booth encoder partial product

| $PP_i$ | $pp_{8i}$ | $pp_{7i}$ | $pp_{6i}$ | $pp_{5i}$ | $pp_{4i}$ | $pp_{3i}$ | $pp_{82i}$ | $pp_{81i}$ | $pp_{0i}$ | $n_i$ |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| A | $a_7$ | $a_7$ | $a_6$ | $a_5$ | $a_4$ | $a_3$ | $a_2$ | $a_1$ | $a_0$ | 0 |
| -A | $-a_7$ | $a_7$ | $-a_6$ | $-a_5$ | $-a_4$ | $-a_3$ | $-a_2$ | $-a_1$ | $-a_0$ | 1 |
| 2A | $a_7$ | $a_6$ | $a_5$ | $a_4$ | $a_3$ | $a_2$ | $a_1$ | $a_0$ | 0 | 0 |
| -2A | $-a_7$ | $-a_6$ | $-a_5$ | $-a_4$ | $-a_3$ | $-a_2$ | $-a_1$ | $-a_0$ | 1 | 1 |

**Table 2b.** $K$ metric values for A

| $a_0$ | $a_1$ | $a_2$ | $a_3$ | $a_4$ | $a_5$ | $a_6$ | $a_7$ | Total |
|---|---|---|---|---|---|---|---|---|
| 0.0237 | 0.0141 | 0.0206 | 0.0196 | 0.0291 | 0.0587 | 0.0021 | 0.0015 | 0.1701 |

**Table 2c.** $K$ metric values for A

| $b_0$ | $b_1$ | $b_2$ | $b_3$ | $b_4$ | $b_5$ | $b_6$ | $b_7$ | Total |
|---|---|---|---|---|---|---|---|---|
| 0.1042 | 0.1028 | 0.0163 | 0.0241 | 0.0366 | 0.0644 | 0.1159 | 0.2335 | 0.6978 |

The design of fixed-width Booth multiplier involves eliminating truncation part (TP), in order to limit large truncation errors, while simultaneously including column information from w in order to decrease this error as much as possible. Assuming w=2, for instance, product results are computed with partial products of main part columns $TP_{mj}$ versus those representing the truncated portion $TP_{mi}$ columns as shown in Fig. 1 as an illustration of this concept.

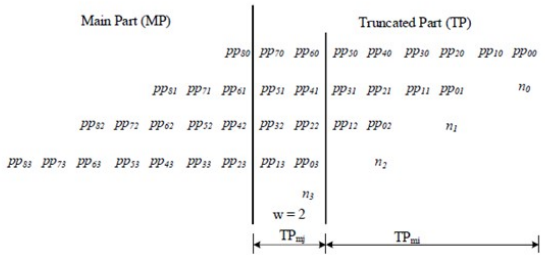


**Fig. 1.** Partial Products based on w=2

### 2.2. Approximation of Carry Function Method

The fixed-width multiplier can be expressed by considering the fixed-width product $P_f$ as:

$$P \approx P_f = MP + \sigma \cdot 2^N \tag{3}$$

where $\sigma$ denotes the compensated bias of the ACF estimator, comprising both $TP_{mj}$ and $TP_{mi}$ parts. The estimation of the $TP_{mi}$ part incorporates a combination of ideal and base carry functions.

Table 2b and Table 2c provide examples of ideal carry functions ($c_1$ and $c_2$) dependent upon bits A and B in an 8-bit fixed width multiplier using $K_u$ as an indicator to quantify each dependency between an ideal carry function ($c_1$) and bits (A and B) respectively. $K_u$ is utilized as a quantitative way to quantify this dependency between ideal carry functions ($c_1$) and bits (A/B).

$$\begin{aligned} K_u = abs\left(P\left(\frac{c_{\text{ideal }1}=1}{u=0}\right) - P\left(\frac{c_{\text{ideal }1}=1}{u=1}\right)\right) \\ + \text{abs}\left(P\left(\frac{c_{\text{ideal }2}=1}{u=0}\right) - \left(\frac{c_{\text{ideal }2}=1}{u=1}\right)\right) \end{aligned} \tag{4}$$

where $u$ denotes the bits of $A$ and $B$.

Eq. (4) is derived from the theoretical framework and methodology, specifically tailored for the approximation of the carry function in fixed-width Booth multipliers. This Eq. (4) characterizes the relationship between ideal carry functions and various bits by measuring how significantly their probabilities change when one bit changes from zero to one; specifically, we will examine an example calculation of $Ku$ values for bits with values such as $u = x_0$ as an illustration of this process.

An 8-bit multiplier offers 65536 input combinations;
when $a_0 = 1$, this gives rise to 20196 possible ones in $c_1$,
when $a_0 = 0$, this gives rise to 20661 possible ones in $c_1$,
when $a_0 = 1$, this gives rise to 28898 possible ones in $c_2$.
when $a_0 = 0$, this gives rise to 28583 possible ones in $c_2$.

To get the probabilities, we divide each by 65536/2 = 32768

$$
\begin{aligned}
P\left(\frac{c_1=1}{a_0=1}\right) &= \frac{20196}{32768} = 0.6163 \\
P\left(\frac{c_1=1}{a_0=0}\right) &= \frac{20661}{32768} = 0.6305 \\
P\left(\frac{c_2=1}{a_0=1}\right) &= \frac{28898}{32768} = 0.8819 \\
P\left(\frac{c_2=1}{a_0=0}\right) &= \frac{28583}{32768} = 0.8723
\end{aligned} \tag{5}
$$

Substituting Eq. (5) into Eq. (4) allows us to obtain the K metric:

$$
\begin{aligned}
K_{b0} &= abs(0.6305 - 0.6163) + abs(0.8723 - 0.8819) \\
&= 0.0238
\end{aligned} \tag{6}
$$

As with the previous $K$ metrics values for different bits of A and B can also be determined for every combination of bits A/B of an N -bit multiplier. When this analysis is carried out for every combination A x B of an N -bit multiplier it becomes evident that its multiplier (B) values exhibit strong relationships to ideal carry functions $c_1$ and $c_2$, as shown herein. Given that twocarry functions c1 and c2 depend on each input value of multiplier (B), we approximated them using this approach:

$$
\begin{aligned}
c_1(B), c_2(B) = \text{argmin} \sum_{A=0}^{A=N-1} \Big[ & absA \times B - MP - TP_{mj} - \\
& \text{round}\left(c_{b3} + 2^{-1}\left(c_{b2} + c_2(B)\right) + 2^{-2}\left(c_{b1} + c_1(B)\right) \cdot 2^{L}\right)\Big]
\end{aligned} \tag{7}
$$

For each combination of $c_1$ and $c_2$ (i.e., 00, 01, 10, and 11) in Eq. (7), sum up all the values of the errors for the fixed multiplier value (B) alongside the fixed base carry value. Then, select the combinations that yield the minimum absolute error. This process is repeated for all values of B to derive $c_1$ and $c_2$ as functions of Y.

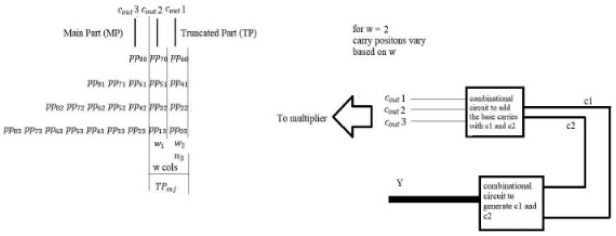


**Fig. 2.** Approximate Carry Function-1 method

To minimize the compensation error, we integrate the ideal carry values with the base carry values. Table 3 presents the base carry values $c_{b1}, c_{b2}$ and $c_{b3}$ offering a global approximation of the $TP_{mi}$ part. As the base carry values are predetermined for each multiplier, separate adders are unnecessary for adding them with $c_1$ and $c_2$.

**Table 3.** Base carry values for ACF methods

| w | Method | $c_{b3}$ | $c_{b2}$ | $c_{b1}$ |
|---|---|---|---|---|
| 1 | ACF-1 | 0 | 0 | 1 |
| | ACF-2 | 0 | 1 | 0 |
| | ACF-3 | 0 | 1 | 1 |
| 2 | ACF-1 | 0 | 1 | 0 |
| | ACF-2 | 0 | 1 | 1 |
| | ACF-3 | 1 | 0 | 0 |
| 3 | ACF-1 | 1 | 0 | 0 |
| | ACF-2 | 1 | 0 | 1 |
| | ACF-3 | 1 | 1 | 0 |

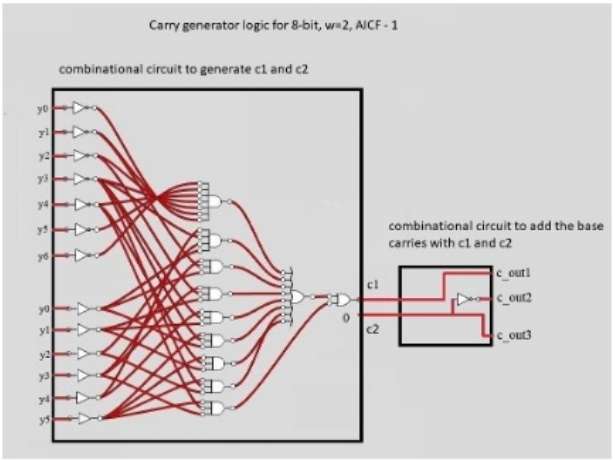


**Fig. 3.** Internal circuit for ideal and base carries

For instance, let's consider the ACF-1 method with $w = 2$ as depicted in Fig. 2. The fixed base carry values are 010, as illustrated in Table 3. By substituting fixed base value with each combination of $c_1$ and $c_2$ based on varying $Y$ in Eq. (7), the function of $c_1$ and $c_2$ is obtained. Here 00 and 01 combinations gives minimum error. Since this combination is contingent upon B, the Boolean logic circuit is derived from the truth table utilizing the QuineMcCluskey (QM) method. In Fig. 3, the first square box shows the generation of $c_1$ and $c_2$ using combinational circuit. The second square box shows the combinational circuit to add the base carries with $c_1$ and $c_2$. The Final carry functions derived as the output from this logic circuit $C_{out\,1}, C_{out2}, C_{out}$ are combined with the MP of the partial product terms in order to reach final output which provides lower error compensation. The base carry values are selected as follows:

For ACF-1 with $w = 1$, the base combination 001, along with $c_1$ and $c_2$, yields the minimum error. For $w = 2$, the combination 010 produces the minimum error, and for $w = 3$, the combination 100 is chosen for ACF-1. For ACF-2 and ACF-3, the aim is to increment the base carry amount by the least count. This is executed to simplify

**Table 4a.** Comparison of $\varepsilon_{\text{mean}}$ values for various methods

| w | Methods N=8 | | N=10 | N=12 | N=14 | N=16 |
|---|---|---|---|---|---|---|
| - | DT | 1.501 | 1.875 | 2.250 | 2.500 | 3.000 |
| - | PT | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| 1 | Chen [11] | 0.3302 | 0.3424 | 0.3567 | 0.3687 | 0.3802 |
| | He et al. [15] | 0.2999 | 0.3197 | 0.3286 | 0.3465 | 0.3594 |
| | Zhang and He [14] | 0.3432 | 0.3976 | 0.4052 | 0.4142 | 0.4221 |
| | Aizaz and Khare [16] | 0.3145 | 0.3269 | 0.3318 | 0.3519 | 0.3605 |
| | He et al. [12] | 0.3254 | 0.3367 | 0.3482 | 0.3604 | 0.3726 |
| | ACF-1 | 0.2989 | 0.3154 | 0.3284 | 0.3418 | 0.3537 |
| | ACF-2 | 0.3144 | 0.3236 | 0.3309 | 0.3426 | 0.3541 |
| | ACF-3 | 0.3397 | 0.3428 | 0.3692 | 0.3922 | 0.4054 |
| 2 | Chen [11] | 0.2621 | 0.2681 | 0.2713 | 0.2753 | 0.2784 |
| | He et al. [15] | 0.2612 | 0.2648 | 0.2682 | 0.2735 | 0.2771 |
| | Zhang and He [14] | 0.2635 | 0.2721 | 0.2882 | 0.2895 | 0.2924 |
| | Aizaz and Khare [16] | 0.2615 | 0.2654 | 0.2699 | 0.2741 | 0.2775 |
| | He et al. [12] | 0.2604 | 0.2643 | 0.2678 | 0.2709 | 0.2742 |
| | ACF-1 | 0.2592 | 0.2640 | 0.2677 | 0.2714 | 0.2740 |
| | ACF-2 | 0.2673 | 0.2684 | 0.2691 | 0.2718 | 0.2743 |
| | ACF-3 | 0.2695 | 0.2705 | 0.2812 | 0.2873 | 0.2902 |
| 3 | Chen [11] | 0.2528 | 0.2549 | 0.2559 | 0.2570 | 0.2584 |
| | He et al. [15] | 0.2524 | 0.2537 | 0.2545 | 0.2562 | 0.2579 |
| | Zhang and He [14] | 0.2563 | 0.2574 | 0.2582 | 0.2596 | 0.2612 |
| | Aizaz and Khare [16] | 0.2529 | 0.2542 | 0.2553 | 0.2568 | 0.2580 |
| | He et al. [12] | 0.2521 | 0.2533 | 0.2541 | 0.2552 | 0.2566 |
| | ACF-1 | 0.2514 | 0.2529 | 0.2541 | 0.2550 | 0.2564 |
| | ACF-2 | 0.2538 | 0.2542 | 0.2545 | 0.2551 | 0.2567 |
| | ACF-3 | 0.2542 | 0.2555 | 0.2583 | 0.2595 | 0.2608 |

the compensation circuits (by adding 1). For instance, for $w = 1$, we add 001 to the combination of ACF-1 to obtain 010, which becomes the combination for ACF-2. Similarly, this process is repeated for all values of w and also for ACF-3. The logic circuit that adds these base values with $c_1$ and $c_2$ to give the final carry values are designed by substituting the above values in the adder circuit.

The ACF-1, ACF-2 and ACF-3 methods differ by using different values for their coefficients $c_{b1}, c_{b2}$ and $c_{b3}$. The output carries are obtained as follows:

$$\{c_{\text{out }4}, c_{\text{out }3}, c_{\text{out }2}, c_{\text{out }1}\} = \{0, c_2, c_1\} + \{c_{\text{b3}}, c_{\text{b }2}, c_{\text{b }1}\} \tag{8}$$

Fig. 4 depicts the mapping of $c_{\text{out }1}, c_{\text{out }2}, c_{\text{out }3}$ and $\text{cout}_4$ to the partial product terms.

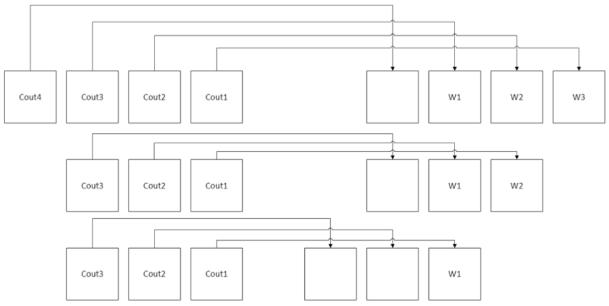


**Fig. 4.** Output carry mapping to partial products

## 3. Results and discussions

This section presents a comparison of the accuracy and circuit performances of the proposed multiplier with various recently proposed fixed-width Booth multipliers [11, 12, 14–16].

### 3.1. Accuracy

To assess the accuracy of various fixed-width Booth multipliers, we use specific error metrics: normalized mean error ($\varepsilon_{\text{mean}}$), maximum absolute error ($|\varepsilon|_{\text{max}}$), and mean absolute error ($|\varepsilon_{\text{mean}}|$). These metrics are defined as follows:

$$\begin{aligned} \varepsilon_{\text{mean}} &= \text{Mean}\{P_A - P_T\}/2^n \\ |\varepsilon|_{\text{max}} &= \max\{|P_A - P_T|\}/2^n \\ |\varepsilon_{\text{mean}}| &= \text{Mean}\{|P_A - P_T|\}/2^n \end{aligned} \tag{9}$$

where Mean {} represents the mean operation.

In order to compare accuracy, we have performed exhaustive simulation on various recent works [11, 12, 14–16] and the proposed fixed-width multipliers for different operand lengths like 8, 10, 12, 14 and 16. Tables 4a to 4c compare the mean error ($\varepsilon_{\text{mean}}$), maximum error ($|\varepsilon|_{\text{max}}$), and average absolute error ($|\varepsilon_{\text{mean}}|$) across various methods

**Table 4b.** Comparison of $|\varepsilon|_{max}$ values for various methods

| w | Methods | N = 8 | N = 10 | N = 12 | N = 14 | N = 16 |
|---|---|---|---|---|---|---|
| x | DT | 4.000 | 5.000 | 6.000 | 7.000 | 8.000 |
| | PT | 0.500 | 0.500 | 0.500 | 0.500 | 0.500 |
| 1 | Chen [11] | 1.500 | 1.500 | 2.000 | 2.000 | 2.500 |
| | He et al. [15] | 1.246 | 1.500 | 1.750 | 2.000 | 2.250 |
| | Zhang and He [14] | 1.600 | 1.800 | 2.100 | 2.300 | 2.600 |
| | Aizaz and Khare [16] | 1.550 | 1.700 | 2.000 | 2.250 | 2.500 |
| | He et al. [12] | 1.500 | 2.000 | 2.000 | 2.000 | 2.167 |
| | ACF-1 | 1.450 | 1.600 | 1.900 | 2.150 | 2.300 |
| | ACF-2 | 1.500 | 1.700 | 2.000 | 2.200 | 2.400 |
| | ACF-3 | 1.550 | 1.750 | 2.050 | 2.300 | 2.500 |
| 2 | Chen [11] | 0.750 | 1.000 | 1.000 | 1.250 | 1.250 |
| | He et al. [15] | 0.720 | 0.950 | 1.000 | 1.150 | 1.200 |
| | Zhang and He [14] | 0.800 | 1.000 | 1.100 | 1.200 | 1.300 |
| | Aizaz and Khare [16] | 0.780 | 0.980 | 1.050 | 1.150 | 1.250 |
| | He et al. [12] | 0.750 | 1.000 | 1.000 | 1.250 | 1.250 |
| | ACF-1 | 0.700 | 0.900 | 0.950 | 1.150 | 1.200 |
| | ACF-2 | 0.720 | 0.950 | 0.970 | 1.180 | 1.230 |
| | ACF-3 | 0.740 | 0.980 | 0.990 | 1.210 | 1.250 |
| 3 | Chen [11] | 0.625 | 0.750 | 0.750 | 0.875 | 0.875 |
| | He et al. [15] | 0.600 | 0.720 | 0.750 | 0.820 | 0.850 |
| | Zhang and He [14] | 0.650 | 0.800 | 0.850 | 0.900 | 0.950 |
| | Aizaz and Khare [16] | 0.630 | 0.770 | 0.800 | 0.870 | 0.900 |
| | He et al. [12] | 0.625 | 0.750 | 0.750 | 0.875 | 0.875 |
| | ACF-1 | 0.580 | 0.700 | 0.720 | 0.840 | 0.860 |
| | ACF-2 | 0.600 | 0.720 | 0.740 | 0.860 | 0.880 |
| | ACF-3 | 0.620 | 0.740 | 0.760 | 0.880 | 0.900 |

for different operand lengths( N = 8 to N = 16). The ACF-1 method demonstrates superior performance in terms of mean error ($\varepsilon_{mean}$ ) and average absolute error ($|\varepsilon_{mean}|$) when compared to other methods. It consistently shows lower and more stable error values across different operand lengths, outperforming most other methods [11, 12, 14–16], including other ACF variations and literature-based approaches. While comparing with maximum absolute error, He et al. [15] is better compared to other methods including proposed design. This is because the He et al. [15] method utilizes additional information from the sign bit of Booth encoding to generate the compensation function, leading to increased hardware costs.

### 3.2. Circuit Performance

In this paper, all fixed-width Booth multipliers [11, 12, 14–16] and the proposed fixed width were implemented using the same Booth encoding method and compression tree structure. Each multiplier was described at the gate level using Verilog HDL, synthesized, and mapped to a UMC 90 nm standard-cell library via the Synopsys Design Compiler (DC) with a specified wire load model. Simulations were conducted at a supply voltage of 1.2 V . Additionally, the average power dissipations were simulated using Synopsys Power Compiler with backannotated switching activity files. These details ensure a fair comparison across different techniques and provide a clear and accurate depiction of the test environment.

Tables 4d and 4e presents synthesized results for the proposed ACF methods and recent works [11, 12, 14–16], focusing on various performance metrics. Table 4d details the delay, and area for different N and w values, while Table 4e specifically lists the power and energy consumption associated with these methods. ACF-1 and ACF-2 methods take larger area and delay is also more compared to other designs [11, 14, 15]. ACF-3 method occupies the least area and the propagation delay is also less compared to [11, 14–16]. Furthermore, in Table 4f shows the comparison of various designs in terms of Area-Delay Product (ADP). From this also, it is evident that ACF-3 method provides superior performance compared to recent works [11, 14–16].

All fixed-width multiplier designs that include a compensation function have to compromise between accuracy and hardware costs. A combined parameter, which includes area cost and delay time, is suggested to better understand multipliers. These are critical design metrics for hardware implementation. For error-tolerant applications, it is important to take into account the error metric. Here,

**Table 4c.** Comparison of the Average Absolute Error values for various methods

| $w$ | Methods | N = 8 | N = 10 | N = 12 | N = 14 | N = 16 |
|---|---|---|---|---|---|---|
| 1 | DT | 1.5010 | 1.8752 | 2.2501 | 2.6250 | 3.0000 |
| | Chen [11] | 0.3302 | 0.3424 | 0.3567 | 0.3687 | 0.3809 |
| | He et al. [15] | 0.2999 | 0.3197 | 0.3286 | 0.3465 | 0.3615 |
| | Zhang and He [14] | 0.3432 | 0.3976 | 0.4052 | 0.4142 | 0.4224 |
| | Aizaz and Khare [16] | 0.3145 | 0.3269 | 0.3318 | 0.3519 | 0.3681 |
| | He et al. [12] | 0.3026 | 0.3198 | 0.3320 | 0.3455 | 0.3578 |
| | ACF-1 | 0.2989 | 0.3154 | 0.3284 | 0.3418 | 0.3546 |
| | ACF-2 | 0.3144 | 0.3236 | 0.3309 | 0.3426 | 0.3512 |
| | ACF-3 | 0.3397 | 0.3428 | 0.3692 | 0.3922 | 0.4016 |
| 2 | Chen [11] | 0.2621 | 0.2681 | 0.2713 | 0.2753 | 0.2788 |
| | He et al. [15] | 0.2612 | 0.2648 | 0.2682 | 0.2735 | 0.2782 |
| | Zhang and He [14] | 0.2635 | 0.2721 | 0.2882 | 0.2895 | 0.2965 |
| | Aizaz and Khare [16] | 0.2615 | 0.2654 | 0.2699 | 0.2741 | 0.2781 |
| | He et al. [12] | 0.2621 | 0.2681 | 0.2713 | 0.2753 | 0.2788 |
| | ACF-1 | 0.2592 | 0.2640 | 0.2677 | 0.2714 | 0.2754 |
| | ACF-2 | 0.2673 | 0.2684 | 0.2691 | 0.2718 | 0.2730 |
| | ACF-3 | 0.2695 | 0.2705 | 0.2812 | 0.2873 | 0.2945 |
| 3 | Chen [11] | 0.2528 | 0.2549 | 0.2559 | 0.2570 | 0.2578 |
| | He et al. [15] | 0.2524 | 0.2537 | 0.2545 | 0.2562 | 0.2577 |
| | Zhang and He [14] | 0.2563 | 0.2574 | 0.2582 | 0.2596 | 0.2608 |
| | Aizaz and Khare [16] | 0.2529 | 0.2542 | 0.2553 | 0.2568 | 0.2580 |
| | He et al. [12] | 0.2528 | 0.2550 | 0.2559 | 0.2570 | 0.2578 |
| | ACF-1 | 0.2514 | 0.2529 | 0.2541 | 0.2550 | 0.2557 |
| | ACF-2 | 0.2538 | 0.2542 | 0.2545 | 0.2551 | 0.2556 |
| | ACF-3 | 0.2542 | 0.2555 | 0.2583 | 0.2595 | 0.2601 |
| | PT | 0.2490 | 0.2498 | 0.2499 | 0.2500 | 0.2500 |

**Table 4d.** Comparison of Delay (ns) and Area cost $(\mu m^2)$ for various methods

| w | Methods | Delay N=8 | Delay N=10 | Delay N=12 | Delay N=14 | Area N=8 | Area N=10 | Area N=12 | Area N=14 |
|---|---|---|---|---|---|---|---|---|---|
| 1 | Chen [11] | 0.63 | 0.88 | 1.07 | 1.25 | 107.408 | 133.664 | 168.560 | 225.729 |
| | He et al. [15] | 0.54 | 0.71 | 0.94 | 1.03 | 91.865 | 108.946 | 132.196 | 186.369 |
| | Zhang and He [14] | 0.62 | 0.86 | 1.09 | 1.28 | 106.242 | 136.468 | 172.501 | 231.148 |
| | Aizaz and Khare [16] | 0.52 | 0.68 | 0.91 | 1.01 | 90.316 | 105.648 | 125.428 | 175.927 |
| | ACF-1 | 0.81 | 0.95 | 1.18 | 1.41 | 145.843 | 170.325 | 196.783 | 258.193 |
| | ACF-2 | 0.76 | 0.84 | 1.14 | 1.36 | 121.683 | 149.132 | 183.815 | 232.691 |
| | ACF-3 | 0.48 | 0.62 | 0.83 | 0.96 | 88.452 | 95.328 | 104.681 | 134.214 |
| 2 | Chen [11] | 0.82 | 1.08 | 1.28 | 1.41 | 125.368 | 149.417 | 191.253 | 258.521 |
| | He et al. [15] | 0.67 | 0.83 | 1.01 | 1.15 | 112.012 | 125.271 | 172.928 | 208.793 |
| | Zhang and He [14] | 0.81 | 1.05 | 1.24 | 1.36 | 127.314 | 138.569 | 199.032 | 271.607 |
| | Aizaz and Khare [16] | 0.63 | 0.81 | 0.98 | 1.16 | 110.145 | 120.396 | 164.018 | 205.346 |
| | ACF-1 | 0.95 | 1.11 | 1.29 | 1.43 | 153.657 | 185.672 | 208.634 | 284.956 |
| | ACF-2 | 0.91 | 1.09 | 1.18 | 1.32 | 136.168 | 156.394 | 202.348 | 269.367 |
| | ACF-3 | 0.58 | 0.76 | 0.90 | 1.08 | 109.159 | 128.357 | 135.964 | 168.382 |
| 3 | Chen [11] | 0.84 | 1.14 | 1.32 | 1.62 | 153.561 | 175.946 | 230.928 | 304.793 |
| | He et al. [15] | 0.75 | 0.96 | 1.12 | 1.31 | 137.291 | 164.729 | 201.147 | 285.095 |
| | Zhang and He [14] | 0.84 | 1.11 | 1.29 | 1.56 | 152.142 | 181.193 | 248.324 | 317.289 |
| | Aizaz and Khare [16] | 0.72 | 0.91 | 1.08 | 1.28 | 135.167 | 161.324 | 198.468 | 272.615 |
| | ACF-1 | 0.96 | 1.15 | 1.36 | 1.68 | 168.965 | 201.367 | 285.513 | 357.268 |
| | ACF-2 | 0.92 | 1.13 | 1.32 | 1.64 | 165.637 | 199.634 | 263.496 | 342.627 |
| | ACF-3 | 0.66 | 0.89 | 1.07 | 1.23 | 128.649 | 149.351 | 182.367 | 206.983 |

two parameters are considered as follows:

$$ADM = \text{Area}(A) \times \text{Delay}(D) \times \varepsilon_{\text{mean}}\,(M) \quad (10)$$

$$P_{AEDe} = \text{Area}(A) \times \text{Energy}\,(E) \times \text{Delay}\,(D) \times |\varepsilon_{\text{mean}}| \quad (11)$$

Table 4f displays comparison results between various

**Table 4e.** Comparison of power ($\mu$W) and Energy (nJ) for various methods

| w | Methods | N=8 Power ($\mu$W) | N=10 Power ($\mu$W) | N=12 Power ($\mu$W) | N=14 Power ($\mu$W) | N=16 Power ($\mu$W) | N=8 Energy (nJ) | N=10 Energy (nJ) | N=12 Energy (nJ) | N=14 Energy (nJ) | N=16 Energy (nJ) |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Chen [11] | 22.413 | 37.846 | 42.171 | 63.980 | 76.000 | 14.12 | 33.30 | 45.12 | 79.97 | 95.00 |
| | He et al. [15] | 19.320 | 25.680 | 33.860 | 40.510 | 47.260 | 10.43 | 18.22 | 31.82 | 41.73 | 48.68 |
| | Zhang and He [14] | 24.740 | 34.180 | 39.690 | 56.210 | 68.340 | 15.33 | 29.40 | 43.28 | 71.95 | 87.88 |
| | Aizaz and Khare [16] | 20.520 | 28.460 | 36.710 | 48.160 | 57.230 | 10.67 | 19.36 | 33.44 | 48.62 | 57.80 |
| | He et al. [12] | 22.890 | 30.560 | 38.240 | 47.860 | 56.780 | 15.01 | 25.67 | 38.12 | 65.08 | 78.41 |
| | ACF-1 | 27.740 | 37.520 | 49.390 | 62.140 | 74.960 | 22.47 | 35.64 | 58.28 | 87.63 | 105.65 |
| | ACF-2 | 23.180 | 31.250 | 44.280 | 55.670 | 66.470 | 17.61 | 26.25 | 50.48 | 75.71 | 90.41 |
| | ACF-3 | 19.180 | 27.680 | 38.780 | 45.780 | 54.290 | 9.21 | 17.15 | 32.20 | 43.95 | 52.12 |
| 2 | Chen [11] | 23.750 | 34.410 | 46.370 | 56.750 | 65.470 | 19.48 | 37.14 | 59.40 | 80.02 | 91.43 |
| | He et al. [15] | 20.180 | 29.870 | 41.730 | 51.580 | 60.290 | 13.53 | 24.80 | 42.17 | 59.32 | 69.94 |
| | Zhang and He [14] | 27.460 | 38.920 | 52.310 | 64.280 | 74.950 | 22.25 | 40.92 | 64.65 | 87.42 | 101.38 |
| | Aizaz and Khare [16] | 21.950 | 31.480 | 43.620 | 53.960 | 63.340 | 13.82 | 25.49 | 42.75 | 62.60 | 75.00 |
| | He et al. [12] | 23.490 | 33.820 | 46.280 | 57.130 | 66.340 | 18.73 | 34.13 | 54.68 | 75.72 | 90.63 |
| | ACF-1 | 30.560 | 41.720 | 55.630 | 69.430 | 82.480 | 28.12 | 46.27 | 71.84 | 99.48 | 117.00 |
| | ACF-2 | 25.830 | 35.640 | 50.170 | 63.290 | 75.890 | 23.52 | 39.30 | 63.20 | 83.54 | 99.77 |
| | ACF-3 | 20.630 | 29.780 | 41.270 | 49.870 | 59.380 | 11.96 | 22.63 | 37.95 | 53.85 | 68.41 |
| 3 | Chen [11] | 27.670 | 42.380 | 55.460 | 68.390 | 78.360 | 23.24 | 47.11 | 73.24 | 110.43 | 127.59 |
| | He et al. [15] | 23.410 | 34.970 | 49.370 | 61.680 | 72.180 | 17.56 | 33.63 | 55.38 | 80.80 | 94.31 |
| | Zhang and He [14] | 28.960 | 42.710 | 58.360 | 72.280 | 84.630 | 24.33 | 47.39 | 75.40 | 112.76 | 132.35 |
| | Aizaz and Khare [16] | 24.580 | 36.980 | 50.640 | 63.790 | 74.180 | 17.71 | 33.64 | 54.69 | 81.66 | 95.10 |
| | He et al. [12] | 26.710 | 38.420 | 52.890 | 65.580 | 77.890 | 21.64 | 45.23 | 69.95 | 103.32 | 123.62 |
| | ACF-1 | 35.460 | 50.640 | 67.390 | 83.760 | 99.460 | 33.80 | 58.24 | 91.62 | 140.56 | 166.61 |
| | ACF-2 | 30.780 | 44.970 | 60.270 | 76.280 | 90.380 | 29.41 | 50.80 | 79.94 | 124.98 | 148.66 |
| | ACF-3 | 23.580 | 34.290 | 47.560 | 58.480 | 69.780 | 15.55 | 30.52 | 48.78 | 72.93 | 85.84 |

**Table 4f.** Comparison of Area Delay Product ($\mu m^2.ns$) and Area-Delay-Mean (ADM) error product for various methods

| w | Methods | N=8 | ADP N=10 | N=12 | N=14 | N=8 | ADM N=10 | N=12 | N=14 |
|---|---|---|---|---|---|---|---|---|---|
| 1 | Chen [11] | 67.667 | 117.624 | 180.359 | 22.34366 | 40.27457 | 64.33413 | 104.0329 | 282.161 |
| | He et al. [15] | 49.607 | 77.351 | 124.264 | 14.87717 | 24.72933 | 40.83323 | 66.51416 | 191.960 |
| | Zhang and He [14] | 65.870 | 117.362 | 188.026 | 22.6066 | 46.66332 | 76.18817 | 122.5491 | 295.869 |
| | Aizaz and Khare [16] | 46.964 | 71.840 | 114.139 | 14.77012 | 23.48443 | 37.87136 | 63.80701 | 177.686 |
| | ACF-1 | 118.132 | 161.808 | 232.203 | 35.3099 | 51.03448 | 76.25577 | 124.4331 | 364.052 |
| | ACF-2 | 92.479 | 125.270 | 209.549 | 29.07542 | 40.53766 | 69.3398 | 108.4191 | 316.459 |
| | ACF-3 | 42.456 | 59.103 | 80.885 | 14.42263 | 20.26063 | 32.07803 | 50.53318 | 128.845 |
| 2 | Chen [11] | 102.801 | 161.370 | 244.803 | 26.94434 | 43.26339 | 66.41528 | 100.3509 | 364.514 |
| | He et al. [15] | 75.048 | 103.974 | 174.657 | 19.60255 | 27.53256 | 46.84308 | 65.67062 | 240.111 |
| | Zhang and He [14] | 103.124 | 145.497 | 246.799 | 27.17326 | 39.58986 | 71.12767 | 106.9371 | 369.385 |
| | Aizaz and Khare [16] | 69.391 | 97.520 | 160.737 | 18.14571 | 25.88186 | 43.38293 | 64.72812 | 225.880 |
| | ACF-1 | 145.974 | 206.095 | 269.137 | 37.83651 | 54.40932 | 72.04821 | 110.592 | 407.487 |
| | ACF-2 | 123.912 | 170.469 | 238.770 | 33.12191 | 45.75412 | 64.25318 | 96.64241 | 355.564 |
| | ACF-3 | 63.312 | 97.551 | 122.367 | 17.06264 | 26.38763 | 34.40977 | 52.24624 | 181.852 |
| 3 | Chen [11] | 128.991 | 200.578 | 304.824 | 32.60899 | 51.12744 | 78.00471 | 126.8975 | 493.764 |
| | He et al. [15] | 102.968 | 158.139 | 225.284 | 25.98919 | 40.12008 | 57.33494 | 95.68415 | 373.474 |
| | Zhang and He [14] | 127.799 | 201.124 | 320.337 | 32.75496 | 51.76938 | 82.71126 | 128.4944 | 494.970 |
| | Aizaz and Khare [16] | 97.320 | 146.804 | 212.185 | 24.61236 | 37.31752 | 54.17054 | 89.53988 | 348.947 |
| | ACF-1 | 162.206 | 231.572 | 388.297 | 40.77869 | 58.56457 | 98.66644 | 153.0536 | 600.210 |
| | ACF-2 | 152.320 | 225.586 | 347.814 | 38.67558 | 57.34407 | 88.51885 | 143.3428 | 561.908 |
| | ACF-3 | 84.908 | 132.922 | 195.132 | 21.5837 | 33.96167 | 50.40277 | 66.06587 | 254.589 |

fixed-width Booth multipliers whose operand lengths are 8, 10, 12, 14 and 16 bits. Based on ADM results it is apparent that our proposed ACF-3 fixed-width multiplier outperforms other designs significantly better. The Table 4g shows that ACF-1 consistently exhibits the lowest $P_{AED\varepsilon}$ values across all weight categories (w = 1, 2, 3) and

**Table 4g.** $P_{AED\varepsilon}$ Comparison for Various Methods

| w | Method | N=8 | N=10 | N=12 | N=14 | N=16 |
|---|---|---|---|---|---|---|
| | Chen [11] | 0.187 | 0.188 | 0.250 | 0.250 | 0.313 |
| | He et al. [15] | 0.156 | 0.174 | 0.219 | 0.225 | 0.281 |
| | Zhang and He [14] | 0.171 | 0.199 | 0.203 | 0.207 | 0.210 |
| | Aizaz and Khare [16] | 0.166 | 0.182 | 0.207 | 0.219 | 0.220 |
| | He et al. [12] | 0.187 | 0.250 | 0.250 | 0.250 | 0.270 |
| 1 | ACF-1 | 0.156 | 0.158 | 0.187 | 0.207 | 0.213 |
| | ACF-2 | 0.157 | 0.162 | 0.185 | 0.195 | 0.196 |
| | ACF-3 | 0.170 | 0.171 | 0.185 | 0.196 | 0.200 |
| | Chen [11] | 0.125 | 0.125 | 0.125 | 0.156 | 0.156 |
| | He et al. [12] | 0.125 | 0.125 | 0.125 | 0.156 | 0.156 |
| | Zhang and He [14] | 0.132 | 0.136 | 0.141 | 0.145 | 0.148 |
| 2 | He et al. [15] | 0.131 | 0.134 | 0.135 | 0.137 | 0.140 |
| | Aizaz and Khare [16] | 0.130 | 0.131 | 0.134 | 0.135 | 0.137 |
| | ACF-1 | 0.122 | 0.124 | 0.125 | 0.127 | 0.129 |
| | ACF-2 | 0.127 | 0.128 | 0.129 | 0.130 | 0.132 |
| | ACF-3 | 0.135 | 0.136 | 0.137 | 0.138 | 0.139 |
| | Chen [11] | 0.078 | 0.094 | 0.094 | 0.109 | 0.109 |
| | He et al. [12] | 0.078 | 0.094 | 0.094 | 0.109 | 0.109 |
| | Zhang and He [14] | 0.082 | 0.092 | 0.094 | 0.096 | 0.098 |
| 3 | He et al. [15] | 0.082 | 0.092 | 0.094 | 0.096 | 0.098 |
| | Aizaz and Khare [16] | 0.078 | 0.086 | 0.088 | 0.090 | 0.092 |
| | ACF-1 | 0.078 | 0.082 | 0.083 | 0.085 | 0.087 |
| | ACF-2 | 0.083 | 0.084 | 0.085 | 0.086 | 0.088 |
| | ACF-3 | 0.089 | 0.091 | 0.092 | 0.093 | 0.094 |

**Table 4h.** PSNR Comparison for Various Methods

| w | Method | PSNR (dB) |
|---|---|---|
| 1 | Chen [11] | 29.7 |
| 1 | Zhang and He [14] | 28.9 |
| 1 | He et al. [15] | 29.5 |
| 1 | Aizaz and Khare [16] | 29.3 |
| 1 | He et al. [12] | 30.1 |
| 1 | ACF-1 | 31.2 |
| 1 | ACF-2 | 31.1 |
| 1 | ACF-3 | 30.8 |
| 2 | Chen [11] | 29.1 |
| 2 | Zhang and He [14] | 28.6 |
| 2 | He et al. [15] | 29 |
| 2 | Aizaz and Khare [16] | 28.8 |
| 2 | He et al. [12] | 29.5 |
| 2 | ACF-1 | 30.8 |
| 2 | ACF-2 | 30.5 |
| 2 | ACF-3 | 30.2 |
| 3 | Chen [11] | 28.8 |
| 3 | Zhang and He [14] | 28.2 |
| 3 | He et al. [15] | 28.5 |
| 3 | Aizaz and Khare [16] | 28.4 |
| 3 | He et al. [12] | 29 |
| 3 | ACF-1 | 30.4 |
| 3 | ACF-2 | 30.1 |
| 3 | ACF-3 | 29.8 |

operand lengths (N = 8 to N = 16), making it the most efficient method in balancing power, area, and delay. For w=1, ACF-1 outperforms both other ACF methods and literature-based approaches, particularly at higher N values. This trend continues for w = 2 and w=3, where ACF-1 maintains its superior efficiency compared to alternatives like ACF-2, ACF-3, and methods from Chen [11], He et al. [12], Zhang and He [14], He et al. [15], and Aizaz and Khare [16]. Overall, ACF-1 emerges as the best choice among the methods analyzed in the table.

To demonstrate the effectiveness of the proposed fixed-width Booth multiplier in image processing, we implemented the proposed and also previous multipliers by applying it to edge detection tasks on grayscale images of resolution 256x256 pixels. For this evaluation, we employed a 16-bit operand length (N = 16) with different weight values (w = 1, 2, 3). When designing the multiplier system, these weight values had different impacts. Peak Signal-toNoise Ratio (PSNR) metrics were employed to assess the quality of output images, with higher PSNR values signifying more precise edge detection. As shown in Table 4h, the proposed ACF-1 multiplier consistently achieved higher PSNR values, demonstrating its ability to minimize error and preserve edge detail in edge-detected images. As such, it makes an ideal solution for image processing tasks which require computational efficiency as well as output fidelity.

## 4. Conclusion

This paper introduces an ACF approach by employing both ideal and base carry functions, an adaptive ACF approach has been devised. The proposed ACF method demonstrates superior performance across various key metrics, including mean error ($\varepsilon_{\text{mean}}$) and average absolute error ($|\varepsilon_{\text{mean}}|$), and $P_{AED\varepsilon}$. The ACF-1 variant consistently outperforms other recent methods [11, 12, 14–16] in terms of accuracy and efficiency, particularly in balancing power, area, and delay, making it an optimal choice for error-tolerant applications. Notably, in edge detection applications, the ACF-1 method provides higher PSNR, ensuring better image clarity and quality retention. Additionally, the ACF-3 method shows significant advantages in minimizing area and delay, further highlighting the versatility of the proposed designs.

## References


[1] C. R. Baugh and B. A. Wooley, (1973) *"A two's complement parallel array multiplication algorithm"* **IEEE Transactions on computers *100***(12): 1045–1047. DOI: 10.1109/T-C.1973.223648.

[2] J.-H. Tu and L.-D. Van, (2009) *"Power-efficient pipelined reconfigurable fixed-width Baugh-Wooley multipliers"* **IEEE transactions on computers *58***(10): 1346–1355. DOI: 10.1109/TC.2009.89.

[3] A. Badawi, A. Alqarni, A. Aljuffri, M. S. BenSaleh, A. M. Obeid, and S. M. Qasim. "FPGA realization and performance evaluation of fixed-width modified Baugh-Wooley multiplier". In: *2015 Third International Conference on Technological Advances in Electrical, Electronics and Computer Engineering (TAEECE)*. IEEE. 2015, 155–158. DOI: 10.1109/TAEECE.2015.7113618.

[4] R. Keote and P. Karule, (2018) *"Performance Analysis of Fixed Width Multiplier using Baugh Wooley Algorithm"* **International Organization of Scientific Research Journal of Very Large-Scale Integration and Signal Processing *8***(3): 31–38. DOI: 10.9790/4200-0803013138.

[5] A. D. Booth, (1951) *"A signed binary multiplication technique"* **The Quarterly Journal of Mechanics and Applied Mathematics *4***(2): 236–240. DOI: 10.1093/qjmam/4.2.236.

[6] L.-D. Van, S.-S. Wang, and W.-S. Feng, (2000) *"Design of the lower error fixed-width multiplier and its application"* **IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing *47***(10): 1112–1118. DOI: 10.1109/82.877155.

[7] S.-J. Jou, M.-H. Tsai, and Y.-L. Tsao, (2003) *"Low-error reduced-width Booth multipliers for DSP applications"* **IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications *50***(11): 1470–1474. DOI: 10.1109/TCSI.2003.817779.

[8] K.-J. Cho, K.-C. Lee, J.-G. Chung, and K. K. Parhi, (2004) *"Design of low-error fixed-width modified booth multiplier"* **IEEE Transactions on Very Large Scale Integration (VLSI) Systems *12***(5): 522–531. DOI: 10.1109/TVLSI.2004.825853.

[9] J.-P. Wang, S.-R. Kuang, and S.-C. Liang, (2009) *"High-accuracy fixed-width modified booth multipliers for lossy applications"* **IEEE transactions on very large scale integration (VLSI) systems *19***(1): 52–60. DOI: 10.1109/TVLSI.2009.2032289.

[10] C.-Y. Li, Y.-H. Chen, T.-Y. Chang, and J.-N. Chen, (2011) *"A probabilistic estimation bias circuit for fixed-width Booth multiplier and its DCT applications"* **IEEE Transactions on Circuits and Systems II: Express Briefs *58***(4): 215–219. DOI: 10.1109/TCSII.2011.2111610.

[11] Y.-H. Chen, (2014) *"An accuracy-adjustment fixed-width booth multiplier based on multilevel conditional probability"* **IEEE transactions on very large scale integration (VLSI) systems *23***(1): 203–207. DOI: 10.1109/TVLSI.2014.2302447.

[12] W.-Q. He, Y.-H. Chen, and S.-J. Jou, (2015) *"High-accuracy fixed-width booth multipliers based on probability and simulation"* **IEEE Transactions on Circuits and Systems I: Regular Papers *62***(8): 2052–2061. DOI: 10.1109/TCSI.2015.2440731.

[13] G. G. Kumar and S. K. Sahoo. "Power-delay product minimization in high-performance fixed-width multiplier". In: *TENCON 2015-2015 IEEE Region 10 Conference*. IEEE. 2015, 1–4. DOI: 10.1109/TENCON.2015.7372864.

[14] Z. Zhang and Y. He, (2017) *"A low-error energy-efficient fixed-width booth multiplier with sign-digit-based conditional probability estimation"* **IEEE Transactions on Circuits and Systems II: Express Briefs *65***(2): 236–240. DOI: 10.1109/TCSII.2017.2709801.

[15] Y. He, X. Yi, Z. Zhang, B. Ma, and Q. Li, (2020) *"A probabilistic prediction-based fixed-width booth multiplier for approximate computing"* **IEEE Transactions on Circuits and Systems I: Regular Papers *67***(12): 4794–4803. DOI: 10.1109/TCSI.2020.3001654.

**[16]** Z. Aizaz and K. Khare, (2021) *"Area and power efficient truncated booth multipliers using approximate carry-based error compensation"* **IEEE Transactions on Circuits and Systems II: Express Briefs *69***(2): 579–583. DOI: 10.1109/TCSII.2021.3094910.

**[17]** R. Marimuthu, Y. E. Rezinold, and P. S. Mallick, (2016) *"Design and analysis of multiplier using approximate 15-4 compressor"* **IEEE Access *5***: 1027–1036. DOI: 10.1109/ACCESS.2016.2636128.

**[18]** M. H. Haider and S.-B. Ko, (2023) *"Booth encoding-based energy efficient multipliers for deep learning systems"* **IEEE Transactions on Circuits and Systems II: Express Briefs *70***(6): 2241–2245. DOI: 10.1109/TCSII.2022.3233923.

**[19]** R. Marimuthu, D. Bansal, S. Balamurugan, and P. Mallick, (2013) *"Design of 8-4 and 9-4 compressors forhigh speed multiplication"* **American Journal of Applied Sciences *10***(8): 893. DOI: 10.3844/ajassp.2013.893.900.

**[20]** R. Marimuthu and P. Mallick, (2017) *"Design of Efficient Signed Multiplier Using Compressors for FFT Architecture."* **Journal of Engineering Science & Technology Review *10***(2): DOI: 10.25103/jestr.102.13.

**[21]** E. Zacharelos, I. Nunziata, G. Saggese, A. G. Strollo, and E. Napoli, (2022) *"Approximate recursive multipliers using low power building blocks"* **IEEE Transactions on Emerging Topics in Computing *10***(3): 1315–1330. DOI: 10.1109/TETC.2022.3186240.

**[22]** R. Marimuthu, S. Balamurugan, and P. S. Mallick, (2018) *"Design of 5-3 multicolumn compressor for high performance multiplier"* **International Journal of Computer Aided Engineering and Technology *10***(5): 568–575. DOI: 10.1504/IJCAET.2018.094334.

**[23]** W. Liu, T. Cao, P. Yin, Y. Zhu, C. Wang, E. E. Swartzlander, and F. Lombardi, (2018) *"Design and analysis of approximate redundant binary multipliers"* **IEEE Transactions on computers *68***(6): 804–819. DOI: 10.1109/TC.2018.2890222.