DESIGN OF AN EFFICIENT LOW ENERGY SIGNAL AWARE APPROXIMATE MULTIPLIER

[#]AMARI BHAVAYA, [$]M PAVITRA
[#]M TECH VLSI STUDENT
[$]ASSOCIATE PROFESSOR, DEPARTMENT OF ECE
PBR VISVODAYA INSTITUTE OF TECHNOLOGY AND SCIENCE (PBRVITS), KAVALI, ANDHRA PRADESH.
bhavya.a410@gmail.com

ABSTRACT

This research paper represents the Error-tolerant applications like multimedia and machine learning frequently use estimated computing, that relaxes full correctness to obtain better performance or decreased power usage. For major of these applications, multiplication is an essential operation. The 4:2 compressor used in this study fairly distributes errors in both positive and negative sign. Two 8 x 8 impartial approximate multipliers (UBAMs), based on the unbiased 4:2 compressor, are created to satisfy different accuracy (or power) needs. According to experimental data, one of the suggested designs offers a 39 percent smaller PDP (power-delay product) and roughly a 46 percent smaller energy-delay product when compared to earlier approximation multipliers (EDP), while the additional offers a decreased NMED (normalized mean error distance). The recommended multipliers outperform prior approximations in image filtering applications because they produce outputs of higher quality while using less energy.

Key Words: Machine Learning, Compressor, Approximate Multipliers.

---

1. INTRODUCTION

Big data processing and artificial intelligence are gaining importance because these applications require vast volumes of data and complex computations. To assist the development of these revolutionary technologies, energy-efficient, high-performance general-purpose compute engines and applications-specific integrated circuits are urgently required. However, it is not always necessary to use accurate or high-precision computing. Instead, a few minor mistakes could cancel each other out or barely affect how the computation turns out. As a result, approximation computing (AC) has become a new strategy for designing energy-efficient systems and boosting computing system performance with a minimal loss in accuracy [1]. A. Inspiration Moore's rule [2] has caused an exponential decline in transistor feature size over the past few decades, which has led to constant advancements in integrated circuit performance and power efficiency. However, the supply voltage cannot be further lowered at the Nano scale , which has significantly increased power density. To completely eliminate the heat problems, a portion of the transistors in an IC must be switched off; these transistors are referred to as "black silicon" [3]. According to a study, the "black silicon" area for an 8-nm technology could be as high as 50% [4]. This shows that employing traditional technologies to increase circuit performance and power efficiency is becoming more and more difficult. In order to overcome this issue, new design approaches like as multi-core architectures, mixed integration, and AC [5].The concept of AC was inspired by the realization that many applications, including as machine learning, multimedia, recognition, and classification, can tolerate small errors. Due to limitations in human perception, some errors in image, audio, and video processing do not result in visibly lower output quality. Additionally, there is already a limit to how precisely or exactly important information may be represented because the external input data to a digital system are normally deafening and quantized. Stochastic computing, a type of probability-based computing, employs straightforward logic gates to perform arithmetic operations on random binary bit streams, with the outcome being unaffected by minor errors [6].

2. APPROXIMATE MULTIPLIERS

The three stages of processing that typically make up a combinational multiplier are PP production, PP accumulation, and a final carry propagate addition, as depicted in Figure 9. Let's say there are two inputs for an unsigned multiplier with n x n.

$$A = \sum_{i=0}^{n-1} A_i 2^i$$

$$B = \sum_{i=0}^{n-1} B_i 2^i$$

Equation (1.1)

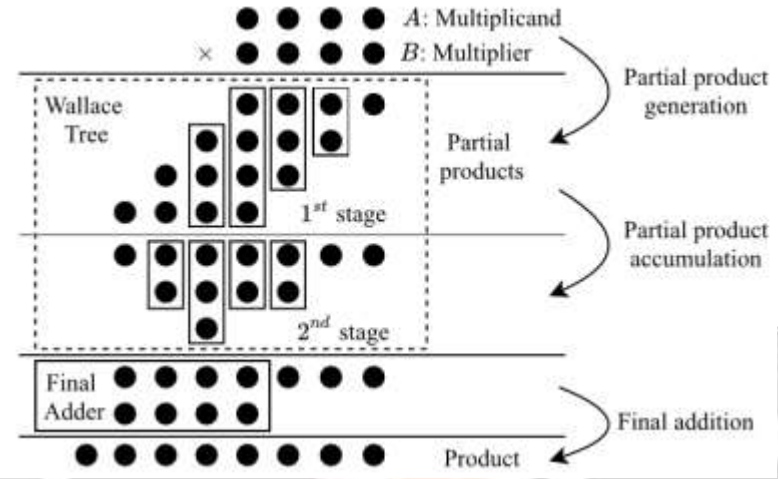

Fig 2.1. Basic arithmetic process of a 4×4 unsigned multiplication. •: an input, a PP, or an output bit;: an HA, an FA, or a 4:2 compressor.

The adders in the following row receive the carry and sum signals produced by the adders in the preceding row in a carry-save adder array, as shown in Fig. 10, for a 4–4 unsigned multiplier. The carry signals are routed diagonally through the adders. As a result, a n n multiplier's critical path is about in O. (n). The array architecture in Fig. 10 can quickly grow to huge arrays while employing mostly short wires due to its regular arrangement. A Wallace tree employs FAs, half adders (HAs), and 4:2 compressors to quickly assemble the PPs for a 44 unsigned multiplier, as shown in the dotted box in Fig. 9. Until there are just two rows of PPs left, the adders in each step operate concurrently without carry propagation. A Wallace tree needs roughly log1.5 (n/2) stages for a n n multiplier [110]. Because it is shorter than that of the array structure, the delay is O (log(n)).

$$A = -A_{n-1}2^{n-1} + \sum_{i=0}^{n-2} A_i 2^i$$

$$B = -B_{n-1}2^{n-1} + \sum_{i=0}^{n-2} B_i 2^i.$$

Equation (1.2)

The multiplier is then recoded using the Booth technique to produce PPs [114]. The number of PPs is cut in half by using the radix-4 version of the Booth algorithm, which was first presented by MacSorley [115]. The foundations of the radix4 scheme are the source of the radix-2r Booth algorithm. Additionally, by merging the complements of the PP rows and preprocessing the constant additions, the Baugh-Wooley algorithm [116] and the modified Baugh-Wooley method [117] can accelerate signed multiplication. The sign extension in Booth multipliers is normally removed using the enhanced Baugh-Wooley method [118]. is quicker than an array structure, but requires more complicated wiring that can expand the circuit area [113]. The two's complement format is used in signed multiplication. the available input operands

Five approaches have been taken into consideration to approximate an unsigned multiplier: 1) approximation in the PP tree [18], [21], [119], [120]; 2) approximation (including truncation) in the PPs [22]; 3) use of approximate adders [121], counters [63], or compressors [79], [80], [122]-[126] in the PP reduction; 4) use of logarithmic approximation [11], [64], [127]-[130]; and 5) use of Approximate Booth multipliers have been created for signed multiplication in order to speed up operation on fewer PPs. As a result, approximation multipliers are divided into signed Booth multipliers and five unsigned categories.

Estimated Unsigned Multipliers, B 1) Approximation in Generating PPs: The UDM uses a roughly 22 multiplier as an early design in order to create higher multipliers [22]. When both inputs are "11," the 22 multiplier approximates the product "1001" with "111," saving one output bit and streamlining the logic circuit. Considering that each input bit has an equal chance of being a "0" or a "1," the ER of this 22 multiplier is 0.54,

or 6.25 percent. 2) In the PP Tree, there is an approximation since some carry-save adders in an array multiplier are left out in both the horizontal and vertical directions [18] for a BAM. Truncating some of the input operands' LSBs makes for a simpler approximation, allowing for the use of a lower multiplier to handle the remaining MSBs. For comparison, this truncated multiplier (TruM) is regarded as a standard design.

In contrast to BAM and TruM, the PP reduction in [132] ignores several consecutive rows of PPs that do not necessarily start from the LSB. This gadget is known as a "PP perforation-based multiplier" (PPAM). The ETM is composed of a control block, a non-multiplication section, and a multiplication portion [21]. The following choices are made by a control block based on NOR gates: 1) The multiplication section is activated to multiple the LSBs exactly if all k MSBs in at least one of the two n-bit input operands is zero. If not, the accurate multiplier (AccuM) is used to multiply the MSBs and the non-multiplication section is used to approximately process the LSBs. Because the LSBs are unnecessary, the static segment multiplier (SSM) eliminates the approximation phase and instead uses a similar partitioning [133].
Depending on whether the MSBs of the first input are all zeros, the LSBs of a given input are multiplied by either the MSBs or the LSBs of the other input. Additionally, a precise k k sub multiplier is used to design a n n dynamic range unbiased multiplier (DRUM) [120]. However, the leading or most significant "1s" of the two n-bit input operands are used to dynamically select the k-bit inputs of the reduced width multiplier. If the leading "1" position is higher than k, the superfluous LSBs are condensed, and the LSB of the k selected bits is set to "1". If not, the sub multiplier's inputs are selected from the k LSBs of the input operands, disregarding the leading "1" position. The final output is then created by restoring the computed result using a barrel shifter. Since the input bits are dealt with more effectively in DRUM than in ETM and SSM, it is more accurate. It also produces unbiased errors, which makes it suitable for accumulative processes. However, the dynamic input selection uses a more complicated circuit. A carry in prediction and a bit-width-aware approximate multiplication are used by an approximation Wallace tree multiplier (AWTM) [119]. A n n AWTM is implemented by four n/2n/2 sub multipliers, with the most crucial sub multiplier AH BH further subdivided into four n/4n/4 sub multipliers. By changing the approximate n/4n/4submultipliers in AH BH, the AWTM can be configured in four different modes. The three n/2n/2 submultipliers (AH BL, AL BH, and AL BL) of lower significance are approximations.
1) Using approximation in the PP Reduction by using approximate counters or compressors When all four inputs are "1," a 4:2 counter used in an incorrect 4 by 4 Wallace multiplier approximates the output of the carry and sum, "100," with "10" [63]. The chance of a PP being "1" is 0.25 given the uniform distribution of the inputs, where the likelihood of a bit being "1" is 0.5.

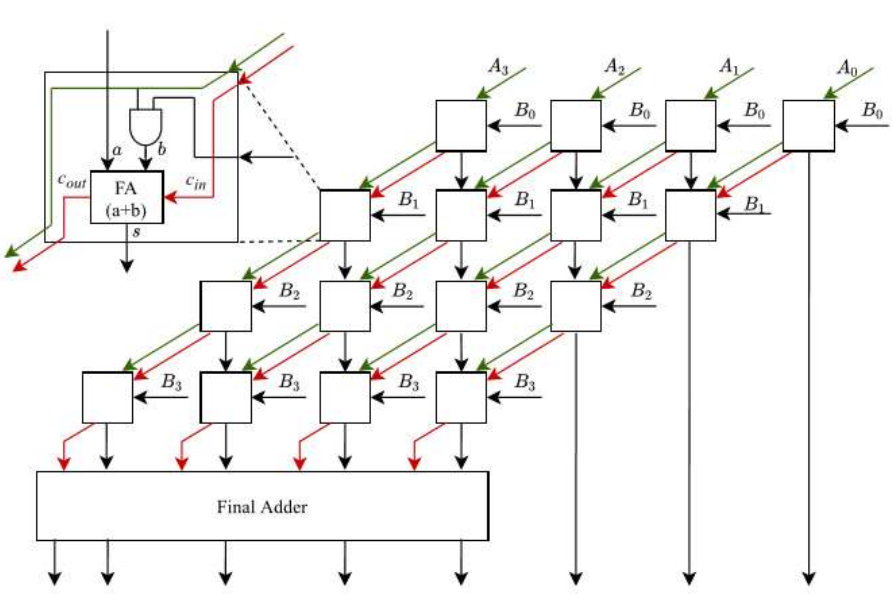

Fig. 1.2 4×4 unsigned multiplier using a carry-save adder array.

Therefore, the ER of the roughly 4:2 counter is only 0.254 = 0.39 percent. The unreliable 4 4 multiplier, which is counter-based, can be used to create larger multipliers (ICM, in general). The approximation counters in [134] do not use the more important output bits in order to implement numerous signed multipliers effectively. For Dadda multipliers, two rough designs that execute the streamlined 4:2 compressor operations are taken into consideration [123]. The PPs are encoded using propagate (i.e., PPi,j +PPj,i) and create (i.e., PPi,j PPj,i) signals

to reduce the error probability, allowing for the design of multiple approximate compressors with a respectably low error probability [124]. Similar to this, a suggested approximate 4:2 compressor for 44 multipliers with encoded inputs is used to build larger multipliers [125]. Using power gating techniques, the dual-quality 4:2 compressors in [79] can change between precise and approximate working modes. The precision of these compressors can be dynamically modified and employed in the PP accumulation of a Dadda multiplier.

A FinFET-based imperfect 4:2 compressor is created using a three-input majority gate for a roughly 88 Dadda multiplier with truncation in the PP array [80]. Each column of PPs in the high-order compressor-based multiplier (HOCM) is accumulated by a single compressor [126]. The usage of accurate and approximate compressors at various phases of the accumulation with the truncation of the lower half PPs is then decided upon using an allocation method. In [121], an unique approximate adder generates a sum from two neighbouring inputs and an error bit for adding up the PPs. Two error recovery approaches that use either OR gates to aggregate the error bits in the so-called approximate multiplier 1 are being studied to reduce the error caused by the approximate adder (AM1) or the approximation adders and OR gates in the approximate multiplier 2 (AM2). The lower half of the PPs in AM1 and AM2 are also truncated to create TAM1 and TAM2, respectively [50], [135].

2) Making Use of Logarithmic Approximation: Mitchell's algorithm makes use of the binary number's logarithmic and anti-logarithmic approximations. The foundation for logarithmic multipliers (LMs) is this [11]. The two unsigned binary input operands A and B of a multiplier are stated as in this procedure.

$$A = 2^{k_1}(1 + x_1)$$
$$B = 2^{k_2}(1 + x_2)$$

Equation (1.3)

where x1 and x2 are the fractional integers that represent the bits to the right of the leading "1"s after being normalised by 2k1 and 2k2, respectively, and k1 and k2 stand for the leading "1" locations of A and B, respectively. A and B's product is then calculated using

$$M = A \times B = 2^{k_1+k_2}(1 + x_1)(1 + x_2).$$

Thus

$$\log_2 M = k_1 + k_2 + \log_2(1 + x_1) + \log_2(1 + x_2).$$

As $0 \le x_1, x_2 < 1$, $\log_2(1 + x_1) \approx x_1$, and $\log_2(1 + x_2)$ x_2. Hence, (8) is approximated by

$$\log_2 M \approx k_1 + k_2 + x_1 + x_2.$$

Equation (1.4)

An antilogarithmic approximation is then used to finish the multiplication. That is

$$M \approx \begin{cases} 2^{k_1+k_2}(x_1 + x_2 + 1), & \text{if } x_1 + x_2 < 1 \\ 2^{k_1+k_2+1}(x_1 + x_2), & \text{if } x_1 + x_2 \ge 1. \end{cases}$$

Equation (1.5)

The ALM-SOA uses a set-one-adder (SOA) for the addition and a truncated binary-logarithm converter to boost the accuracy of an LM [127]. The SOA merely employs an AND gate to produce a carry-in for the MSBs and set the LSBs to a constant value of "1." Furthermore, [64] and [128] both provide an improved method using precise and approximate adders (ILM-EA and ILM-AA). There are various parts to the input operands in [129] between two subsequent powers of two.

3. IMPLEMENTATION FLOW

The suggested architecture is executed using a bottom-up methodology by creating the submodules, which are then connected to form a bigger system. Fig.5.1 depicts the implementation process in flow.

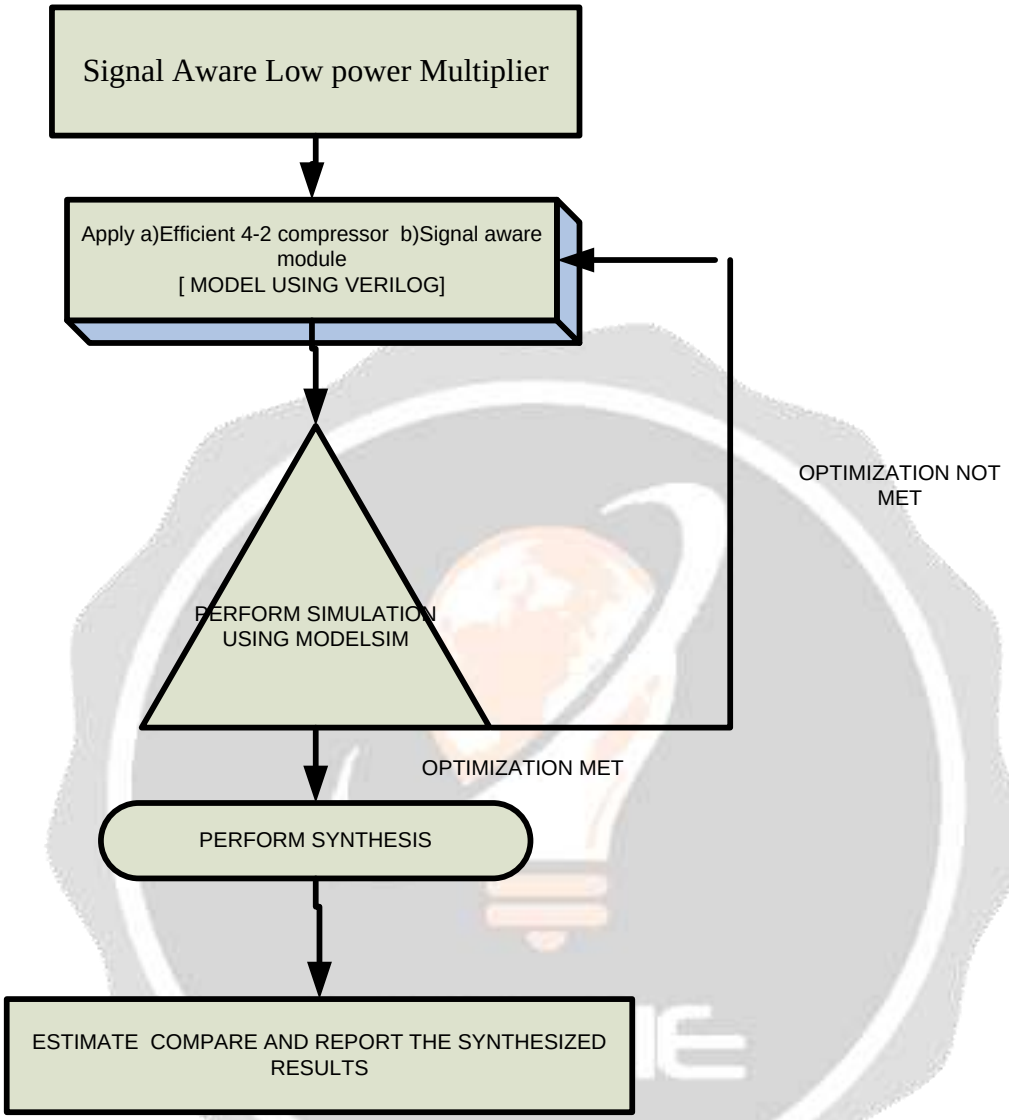

Fig.3.1 Implementation Flow

4. RESULT ANALYSIS

The RTL Verilog code for the hardware implementation was created using the 14.7 update of the Xilinx ISE tool. The various blocks have been synthesised for the Virtex-7 XC7VX980T device, and Table 6.1 displays the implementation results. At the maximum operating frequency fmax, time delay is computed as follows:

We take the framework from [6] as an example, where classifying three different touch input modalities is the objective. The purpose is to evaluate the real-time functionality of the suggested implementations by evaluating the tradeoff between real-time functionality and hardware complexity. Real-time functionality is defined as having a temporal latency of less than 400 milliseconds [29], meaning that the system must perform one classification every 400 millisecond. There are around 100 training tensors (Nt) at the lowest and 1000 at the maximum [1]. As a result, there are between 300 and 3000 total training tensors for the three classes that correspond to the three different input touch modalities. If Nt = 100 and the classification scenario uses a cascaded architecture, 300 kernel function computations must be completed in 400 milliseconds in order to achieve real-time functioning.

Table 4.1 Circuit performance of different multipliers.

Multiplier	Delay(ns)	Power(µW)	Area(µm)2
Exact	1.66	91.2	1438
UBAM-M1	1.55	81.6	1320.2
UBAM-M2	1.18	53.7	896.1
AM[21]	1.55	79.5	1282.1
UDM[8]	1.53	81	1071.7
mul8u_DM1[30]	1.77	59.7	800.2
Proposed	1.06	48.1	1112.3

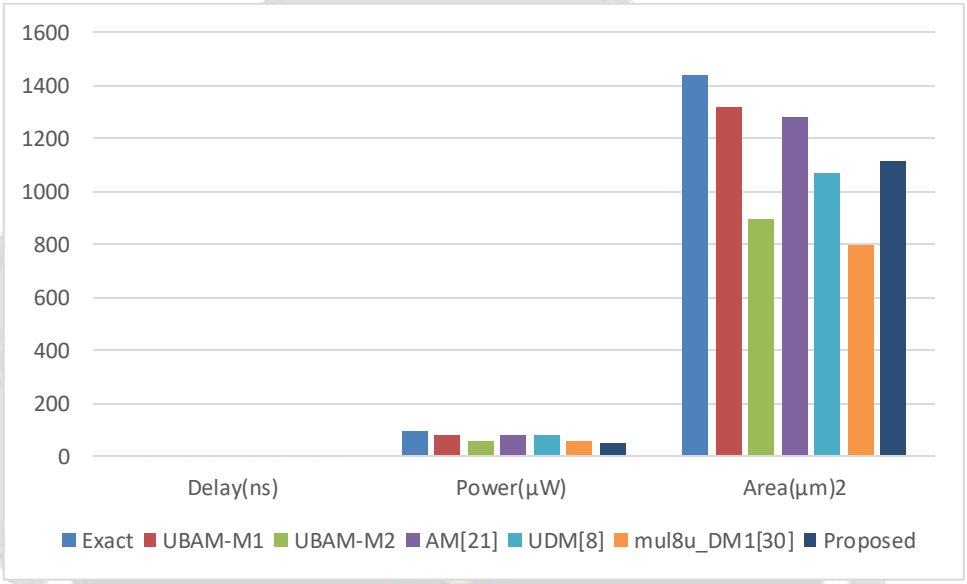

Fig:4.1 Chart of Circuit performance of different multipliers.

Table: 4.2 Comparison of accuracy for various approximation multipliers.

Multiplier	ER	ME	NME	MED
Exact	70.2	7.6	1.02E-04	3.03E+02
UBAM-M1	65.2	1.13E+02	1.16E-04	6.46E+01
UBAM-M2	93.99	2.23E+01	1.34E-03	2.89E+02
AM[21]	66.45	8.78E+01	1.38E-02	9.03E+02
UDM[8]	46.72	9.03E+02	3.09E-04	1.35E+03
mul8u_DM1[30]	98.16	2.03E+01	2.07E-02	3.17E+02
Proposed	56.3	1.02E+01	1.01E-05	2.01E+01

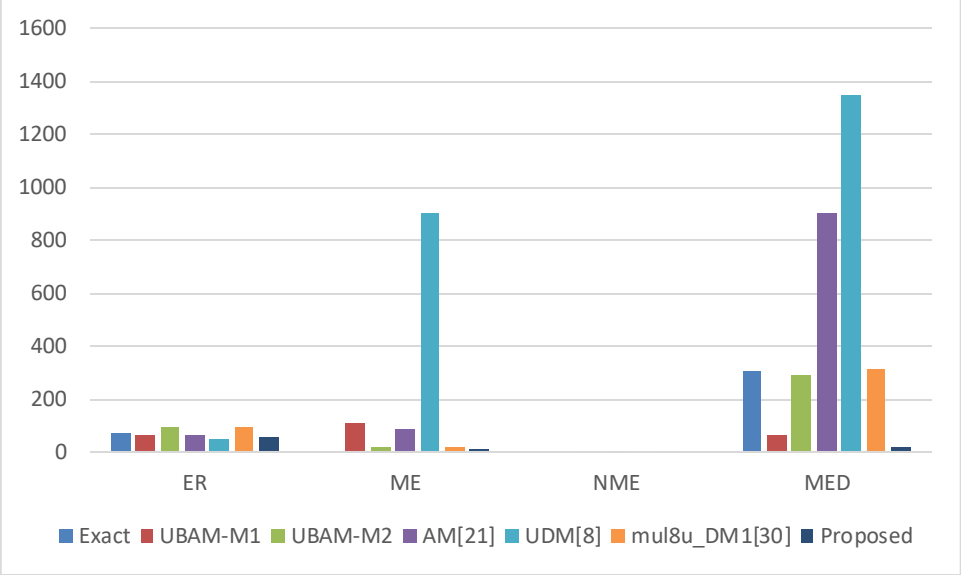

Fig.4.2 Chart comparison for different approximate multipliers.

5. CONCLUSION

This research paper proposed a 4:2 unbiased approximation compressor that yields an equal number of incorrect sign +ve and -ve values. In comparison to the 4:2 compressors in the same column of the partial product compression tree, this can reduce the error of the entire multiplier and increase the likelihood of error compensation. A fresh, 6-inch compressor was recommended. Based on the upcoming compressors, two cutting-edge estimated multipliers, known as UBAM-M1 and UBAM-M2, were developed. The UBAM-M1 can be used in applications that demand high precision. The shortest NMEDs of the approximation designs are all for UBAM-M1 (Table 8), and its MAE is only around 1/18 of UDM's [8]. The UBAM-M2 is advised for better performance and power efficiency because it saves 7 percent on power and 36 percent on delay when compared to the mul8u DM1 [30]. Because of its proposed multiplier architecture, which includes MSB trimming and LSB truncation, UBAM-M2 performs better than AM in terms of latency, power consumption, chip size, roughly 39% PDP, and over 46% EDP. It also outperforms AM by over 22%. The suggested multipliers were evaluated by image filtering applications. It was shown that the UBAM-M1 produces output that is more accurate than that of other approximation multipliers by attaining higher quality (in terms of SSIM or PSNR). UBAM-M2 results in a 58.12 percent PDP savings at a cost of 1.24 percent SSIM deterioration.