A Leading one bit based Scalable Approximate Multiplier with High-Speed Yet Energy-Efficient Truncation- And Rounding-Based for digital Signal Processing applications

1KOTESWARI GONGATI 2SUDHEER KUMAR POTLURI
1PG Scholar Dept. Of ECE (VLSI SYSTEM DESIGN) A.M REDDY MEMORIAL COLLEGE OF ENGINEERING AND TECHNOLOGY
2Assistant Professor Dept. Of ECE(VLSI SYSTEM DESIGN) A.M REDDY MEMORIAL COLLEGE OF ENGINEERING AND TECHNOLOGY

Abstract

Based on the position of the lead system a TOSAM is installed and the quantity of consumer goods is minimized by cutting any input step. The theoretical Elastic Multiplier (TOSAM) is needed here. That is clear. The proposed model would result in substantial changes in resources and regional employment by moving, removing, and restricting multiplier operations of a given length, as opposed to the efficient multiplier. The input formula for the multiplication equation is rounded up to the closest amount of shortcomings in order to increase overall accuracy. Depending on the input operators' truncated position, because the input operand range is low in precision and the multiplier can be scaled. Further changes will be made where design parameters (e.g. field and power consumption) decline in input-operand length. The planned estimated multiplier’s implementation parameters are compared with the actual multiplier and several other recently suggested expected multipliers for impact evaluation. The results show that the suggested average multiplier with an actual small error of 11% to 0.3% decreases time delays, region and energy consumption respectively to 90% and 98% to 41%.

Index Terms—Accuracy configurable, approximate multiplier, area efficient, low energy, scalable, truncating.

1. INTRODUCTION

Power utilization is one of the major digital network design architectural requirements. Calculation approximation (AR) is one means of increasing and/or increasing the usage of electricity. For error-resilient programs, AC can always be included, since the outcome of the application can not function. Types of these innovations include audio and picture processing[1], machine-learning[2] and data mining[3]. Arithmetic operations in a variety of applications for signal processing are responsible for a significant part of the energy usage (e.g. up to 75% of overall energy consumption of the Fourier rapid transformation system[4]). Using that is often used[5]. This renders approximate multipliers suitable for use in error tolerant signal processing systems. In a multiplication cycle, there are normally three stages. The first step is to produce partial goods dependent on input operations. Just two rows of partial products are deposited in the second floor. A (fast) adder adds the remainder of two sides. Increasing of the steps may be calculated. The approach[1],[6],[7] or an improvement in their generation complexity[8] may be given the first step to the sum of partial items. Added estimate to rising delays or power consumption Decreased multiplication process speeds in the second level.

One of such approaches is rugged compressors[9]–[12]. Throughout the final step of the propagation period the design of the adder has a major effect on latency and energy consumption of the propagation mechanism. An indirect adder may also be used to increase the power consumption[13] of the multiplier in the final step. In this paper, we introduce an method strategy that decreases the amount of missing products. In the proposed
approximate Algorithm the inputs are cut to h and t bits according to their lead bits placed. In addition, by rounding the error resulting from the truncation method the estimated sum of the truncated values is calculated. The accurate and effective multiplier calculation is higher than the arta-of-the-arta-approximate multiplier owing to these simplifications. Multiplication of the specified multiplier measuring centerMultiple truncated number and rounding procedures and activities and the component are moved to the left for the final production. Because the arithmetic operations are performed on truncated quantities, the central calculation of the multiplier suggested is tiny and needs less energy than the same multiplier. The precision of the proposed method is primarily calculated by the parameter value t and h and does not impact the input operand distance greatly. This gives a scalability property for the proposed multiplier. The key conclusions are summarized in this report. 1) Modern approach identifying a leading multiplier position and performing both truncating and rounding operations to improve the multiplication accuracy. 2) Exploring t parameters (truncation) and h (rounding) to establish a balance between exact, delay and energy usage. 3) Presentation of hardware implementation of the estimated functional multiplier truncation and rounding(TOSAM) for activities signed as well as unsigned. 4) Review of the possible multiplier specification criteria for the image processing and grading demands. The remaining documents are grouped accordingly. Section II discusses some of the prior research on estimated multipliers. The architecture of the new approximately multiplier is outlined in Section III, while Sections IV explains its hardware implementation and error analysis.

II. LITERATURE SURVEY:

We study some of the research performed on the design of estimated multipliers in this segment. The input operands were cut to m bit in the dynamic segment(DSM) method(1) depending on the location of the leading element, while the values were truncated using a fixed width multiplication. The resultant production is often smaller than the actual one by this method of truncation, producing a negative mean relative error. It is unacceptable since the medium loss near to zero would be a larger signal to noise (SNR) ratio when coping with optical signal management with roughly integer units that have the Gaussian error distribution. In the complex DRUM structure [6], the least important bit of the truncated input was set to "1" for the truncation procedure to bring the MRE to zero. In LETAM structure [5], the input operands were truncated and half of partly-produced components were ignored during the multiplication. Delays and electricity usage have also been increased with the removal of the incomplete goods relative to those in the DSM and DRUM systems. The input operand in RoBAmultipliers[7] was rounded to the next power of two in which certain multiplication, addition and subtraction processes provided the production. Throughout this company, In relation to the same multiplier, which led to improved energy and rpm, the amount of elements to incorporate in order to produce the final effect was decreased. In a certain manner, the least important bits of the component items have been eliminated[14] to increase the multiplier’s pace and size. A easy way to construct partial products is to multiply any multiplier portion, which can simply be achieved through logical AND action. Another method is to encrypt and subtract an encoded multiplier by multiplier in higher radices. The encoding of the multiplier becomes more complicated as the radix decreases. And it should be reduced Approximate encoders may be used to produce partial products utilizing this complexity[16]. The partial products were produced and accumulated in approximate radix-4 booth multipliers in [17]. In addition, an estimated radix-9 multiplier was introduced in [18] to generate the smallest significant bits of the triple multiplicand, using estimated additives. In [8], the most relevant bits of the multiplier were coded by identical encoding of radix-4 and the lowest bits were encoded with a higher approx. encoding of the radix that rounded the least meaningful bits to the nearest power of two. In [9] the reduction was suggested and worked with four estimated 4:2 compressors. The multiplier rates. Appropriate 4:2 compressor and an error recovery element have been suggested in [10] to improve the precision of the multiplication. In [11], the largest estimated multiplier arrangement has provided a range of estimated 5:3 compressors in an about 15:4 compressor. It should be stated that the most significant elements of the test were obtained by utilizing precise compressors to improve the accuracy. Many compressors estimate were suggested in [12]. A concept algorithm for effective estimated multipliers made of such compressors has also been suggested. In [19] many estimated additional elements were known as building blocks.
and a design space was investigated to find the right solution for this approximate multiplier. Another solution is to adjust the counting scheme to the logarithmic scheme such that the speed of multiplication is increased rather than compounded. This method produces the logarithm of the input operands, determines the total of them and conducts an antilogarithm procedure on the total to obtain the final result. This method is complex due to the logarithm and antilogarithm steps produced. The accuracy of the multiplier depends on how such measures are correct. Different analyses of logarithm and antilogarithm numbers were conducted [20],[21]. Mitchell [22] suggested a basic indirect procedure for computing and obtaining multiplication results for the logarithm and Antilogarithm of a number (Mitchell). Since then, experiments on the development of the logarithmic multipliers dependent on Mitchell have been carried out [23],[24]. We are proposing in this paper a certain multiplier to search, truncate and round the leading bits of the input operands, and to render certain change, attach and small fixed width multiplication operations to achieve the output.

Fig. 1. Amount of YAPX according to the amount of Y for the case of S = 4.

III. PROPOSED APPROXIMATE MULTIPLIER

A. TOSAM Each positive integer number (N) can be represented as

\[ N = \sum_{i=0}^{k} 2^i x_i \]  

(1)

where \( k \) denotes the position of its leading one bit and \( x_i \) is the \( i \)th bit of \( N \). By factoring \( 2k \) from (1), it is rewritten as

\[ N = 2^k \left( \sum_{i=0}^{k} 2^{-k} x_i \right) = 2^k \times X \]  

(2)

where \( X \) is a fractional number between 1.0 and 2.0. Based on (2), the result of multiplying \( A \) by \( B \) may be calculated as

\[ A \times B = 2^{k_A + k_B} \times X_A \times X_B . \]  

(3)

\( X_A \) and \( X_B \) widths are the same as \( A \) and \( B \) for the correct period and power usage amount of \( X \). We propose that the average sum of this word be determined based on \( X_A \) and \( X_B \) fractional sections. They reflect the fraction of \( X \) in the remainder of this article, as \( Y \) is from

\[ Y = X - 1. \]  

(4)

For example, assume that \( X = (1.1101)_2 \). In this case, \( Y = (0.1101)_2 \). To generate the approximate value of \( Y \), we divide this range (0.0–1.0) into \( S \) equal segments where \( S \) is a power of two represented by

\[ S = 2^h \]  

(5)

where \( h \) denotes an arbitrary positive integer which is one of our design parameters. It is obvious that the length of each segment is equal to \( 1/S \). We propose to generate the approximate value of \( Y \) as

\[ Y_{APX} = \frac{2m - 1}{2S} \text{ if } \frac{m - 1}{S} \leq Y \leq \frac{m}{S}, \quad m = 1, 2, \ldots, S . \]  

(6)

The estimated sums of \( Y \) for the situation in which \( S \) equals 4 are seen in Figure for a greater example. In order to locate \( Y_{APX} \), only \( h \) most significant pieces of \( Y \) must be remembered. For eg, if \( S = 4 (h = 2) \) is zero if there are two big bits of \( Y \), it implies 0 is < 1/4. So, as \( Y_{APX} \), we chose \( 1/8 = (0.001)_2 \). If two of the most significant \( Y \) pieces are "10," indicating 2/4 of a new \( Y < 3/4 \), \( Y_{APX} \) is then approximated to 5/8 = (0.101)_2. In other words, \( Y_{APX} \) ‘s meaning is obtained by actually splitting \( Y \) into \( h \) bits and adding a "1" bit on the right side. The range of \( Y_{APX} \) thus corresponds to \( h+1 \) bits. (4), (3) shall be revised as

\[ Mukt Shabd Journal \]

Volume IX, Issue VIII, AUGUST/2020  
ISSN NO : 2347-3150

Page No : 1674
To improve the speed of calculation, we truncate $Y_A$ and $Y_B$ to $t$ bits, where in the rest of this paper, we denote by $(Y_A)_t$ and $(Y_B)_t$. Hence, we modify (8) as

$$A \times B \approx 2^{k+1} \times (1 + Y_A + Y_B + (Y_A)_{APX} \times (Y_B)_{APX}).$$

(9)

where the width of $(Y_A)_{APX}$ ($(Y_B)_{APX}$) is $h+1$ bits. To have a better understanding, the dot diagram of the proposed algorithm for the case where $t = 7$ and $h = 3$ compared to that of an exact 16-bit multiplier is depicted in Fig. 2. The green square shows the “1” bit in the term $1+(Y_A)_h+(Y_B)_h+(Y_A)_{APX} \times (Y_B)_{APX}$. Orange circles denote partial products of $(Y_A)_{APX} \times (Y_B)_{APX}$, whereas purple triangles show the bits of $(Y_A)_h$ and $(Y_B)_h$. Gray circles and triangles are omitted and are not considered in the calculations. As shown in Fig. 2, in the exact 16-bit multiplier, the number of partial products is equal to 256, which must be summed to generate the final result while in the proposed method, only 31 of the partial products are kept (which amounts to $\sim 88\%$ partial products reduction). This reduction rate will rise as the bit length of the multiplier input operands increases. As an example, the steps of multiplying $A$ by $B$ for the case of $t = 7$ and $h = 3$ are depicted in Fig. 3. In the rest of this paper, we denote our proposed structures by TOSAM ($X$, $Y$) where $X$ and $Y$ correspond to $h$ and $t$. The accuracy of the proposed approach depends on the values of the parameters $t$ and $h$. Therefore, in the error analysis section (Section IV), we will find a relationship between $t$ and $h$ parameters to achieve an almost high accuracy while having an acceptable speed and energy consumption. Finally, the proposed multiplication approach is feasible for the case of unsigned operands. To use it for signed multipliers, one may find the absolute value of the input operands, multiply them by the proposed algorithm, and fix the sign of the final result according to the sign of the input operands. Finding the exact absolute value of the input operands may degrade the speed of calculation and, hence, we produce it according to the method presented in [7].

Fig. 3. Numeric example of 16-bit TOSAM(3, 7) with $A = 11761$ and $B = 2482$. The approximate result $[(A \times B)_{APX}]$ is equal to 28 901 376 while the exact result $[(A \times B)_{Exact}]$ is equal to 29 190 802. In this case, the absolute error is 289 426 which is about 0.99% of the exact output (the error is less than 1% in this case).

Fig. 4. Block diagram of the proposed approximate signed multiplier.

IV. HARDWARE IMPLEMENTATION

The block diagram of the proposed signed approximate multiplier is depicted in Fig. 4. First, the approximate absolute value of the input operands $[|A|_{app}, |B|_{app}]$ is determined using the Approximate Absolute Unit, similar to the one exploited in [7]. In this unit, the bits of the input are inverted if the input is negative and they are not changed if the input is positive. $|A|_{app}$ and $|B|_{app}$ are injected into the Leading-One Detector Unit [25] and the positions of

$$K[i] = \left\lfloor \frac{n - 2}{\max_{j=0}^{k-1} I[j]} \right\rfloor \wedge I[i] \text{ for } 0 \leq i \leq n-2$$

where $I$ can be either $|A|_{app}$ or $|B|_{app}$. Only one bit of the signal K is “1” revealing the position of the input leading one bit. By using the KA and KB signals in a lookup table, KA and KB signals needed for (7) can be generated. The schematic of the Leading-One Detector Unit for 8-bit input operands is depicted in Fig. 5. For example, assume that $|A|_{app} = (011001)_{2}$, in this case $K_A = (010000)_{2}$ and $K_B = (101010)_{2}$. Signals $|A|_{app}$, $|B|_{app}$, $K_A$, and $K_B$ are then applied to the Truncation Unit [25] to produce $(Y_A)_t$ and $(Y_B)_t$. Assume that the input and output of this unit are I and (Y)_t. In this case, the ith bit
Signals \((Y_A)_h\) and \((Y_B)_h\) are then exerted to the Arithmetic Unit to calculate the term \(1 + (Y_A)_h + (Y_B)_h + (Y_A)_{APX} \times (Y_B)_{APX}\). It should be noted that the h most significant bits of \((Y_A)_{APX}\) and \((Y_B)_{APX}\) are the same as the h most significant bits of \((Y_A)_h\) and \((Y_B)_h\) whose rightmost bits are always “1.” Hence, there is no need to add extra hardware to generate \((Y_A)_{APX}\) and \((Y_B)_{APX}\) signals which are produced by simple wiring. In the Shift Unit, the output of the Arithmetic Unit is shifted to left by \(k_A+k_B\) to produce the term \(2k_A+k_B \times (1+(Y_A)_h + (Y_B)_h + (Y_A)_{APX} \times (Y_B)_{APX})\) [see (9)]. In the Sign and Zero Detector Unit, the sign of the output is set according to the sign of the multiplier input operands and also the output is set to zero if at least one of the inputs is zero. In the case of the unsigned multipliers, the Approximate Absolute Unit should be omitted and the Sign and Zero Detector Unit should be replaced by a Zero Detector Unit. TOSAM can be implemented in an accuracy configurable structure. In order to implement an accuracy configurable structure of TOSAM, all of its units should be designed for the largest desired h and t values such that the design can work in all operation modes. We suggest a configurable TOSAM structure with three different operating modes of T2, T6, and T9 corresponding to TOSAM(0, 2), TOSAM(2, 6), and TOSAM(5, 9), respectively. The Truncation and the Shift Units of the configurable TOSAM should be designed for the largest t and h values \((h = 5\) and \(t = 9\) in this case). In the Arithmetic Unit, some of the adders and logical AND gates should be power gated based on the operating mode to make the design more power efficient. The reduction levels of the partial products based on the operating modes are depicted in Fig. 6. In the last level, a fast 9-bit adder is employed. To decrease its switching activity, some of its inputs are set to “0” with a transmission gate (TG), based on the operating mode. In the T2 mode, only the purple partial products are accumulated, only the purple adders are active (not power gated), and all of the inputs of the 9-bit adder are set to “0.” The 10 least significant bits of the result are set to “0” using the

\[
(Y_A)[i] = \bigvee_{j=0}^{\lceil \frac{n-2}{t} \rceil} (K[j] \land I[j+i+t]) \text{ for } i < t.
\]

Fig. 6. Reduction levels of accuracy configurable TOSAM with three different operating modes.

TGs and purple stars are passed through the TGs to generate four most significant bits of the output. In the T6 mode, only the green and purple partial products are generated and summed to compose the final output and the orange adders are power gated. In addition, the orange inputs of the 9-bit adder are set to “0.” Also, in the eighth column of LEVEL1, there are two orange circles that should be set to “0” by TGs when operating in the T6 mode. In this mode, the six least significant bits of the result are set to “0,” green stars are passed through TGs to generate four intermediate bits of the output, and the four most significant bits of the result are produced by the 9-bit adder. In the T9 mode, all parts are active. The orange stars are passed through the TGs to generate the five least significant bits of the result and the other bits are produced by the 9-bit adder. Note that the least significant bits of \((Y_A)_{APX}\) and \((Y_B)_{APX}\), which depend on the operating mode, should be rounded (set to “1”). It is simply done by performing a logical OR operation on the corresponding bit and the operating mode. For example, when T6 signal is “1,” the logical OR operation sets the corresponding bit of \((Y_A)_{APX}\) and \((Y_B)_{APX}\) to “1.”

EXTENSION:
Here, as an extension, we are going to design a 32-bit TOSAM multiplier with 15, 7. 15-bit truncation and 7-bit rounding. By applying these conditions to basic blocks like an absolute unit and truncation unit, arithmetic unit, and shift unit.

Here, \(a, b\) are operands which are 32-bit wise, then \(n=32\). Approximate unit approximated value according to sign bit wise, then leading one detector unit will detect lead one, then truncation unit truncates value according 15-bit.

IV. SIMULATION RESULTS

PROJECTED SYSTEM:

DESIGN SUMMARY:

TIMING REPORT:

COMPARISION TABLE:

<table>
<thead>
<tr>
<th>S.No</th>
<th>PROJECT</th>
<th>delay</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>TOSAM WITH</td>
<td>22.389ns</td>
</tr>
</tbody>
</table>
V. CONCLUSION

Throughout this paper we proposed a low-energy and area-efficient estimated multiplier, in which the input operations were truncated with t and h of two different lengths and then circled to the closest odd numbers in order to minimize the error induced by the truncation process. In comparison to the same Wallace Multiplier, the proposed 32-bit propagator increased energy efficiency in an average of 95%, and occupied 85% less space. The multiplication delay and power usage were respectively increased by 4% – 41% and 89% – 97%, compared with the same multiplication. The speed, size and energy enhancements of the new multiplier improved as the multiplier diameter expanded in contrast to the same multiplier. This was because the proposed multiplier had a easy and scalable calculation heart. The high accuracy of the suggested multiplier is therefore a strong option for applications like image analysis and classification.

REFERENCES


[14] H. J. Ko and S. F. Hsiao, “Design and application of faithfully rounded and truncated multipliers with combined deletion, reduction,


