International Journal of Research in Advent Technology, Vol.6, No.6, June 2018
E-ISSN: 2321-9637
Available online at www.ijrat.org

Design and Synthesis Of Signed and Un-Signed Approximate Multiplier Using Rounding Based Approximation

[1]A. Sai Sankalpa, [2]C.Leela Mohan
[1]P.G Scholar, [2] Associate professor
[1] Department of ECE,
[1] Narayana Engineering College, SPSR Nellore,A.P,India

Abstract*:* In this paper, we tend to propose an approximate multiplier that's high speed yet energy efficient. The approach is to round the operands to the closest exponent of 2. This manner the procedure intensive a part of the multiplication is omitted up speed and energy consumption at the worth of little error. The proposed approach is applicable to each signed and unsigned multiplications. we tend to proposed three hardware implementations of the approximate multiplier that features one for the unsigned and 2 for the signed operations. The efficiency of the proposed multiplier is evaluated by examining its performance with those of some approximate and exact multipliers using totally different style parameters. Additionally, the efficiency of the proposed approximate multiplier is studied in two image process applications, i.e., image sharpening and smoothing.

Index Terms - Accuracy, approximate computing, energy efficient, error analysis, high speed, multiplier.

1. INTRODUCTION

Energy minimization is one among the most style necessities in any electronic systems, particularly the moveable ones like mobile phones, tablets, and totally different gadgets [1]. It's extremely desired to realize this minimization with least performance (speed) penalty [1]. Digital signal processing (DSP) blocks are key elements of those moveable devices for realizing varied multimedia system applications. The procedure core of those blocks is that the arithmetic logic unit wherever multiplications have the best share among all arithmetic operations performed in these DSP systems [2]. Therefore, rising the speed and power/energy-efficiency characteristics of multipliers plays a key role in rising the efficiency of processors. Many of the DSP cores implement image and video process algorithms wherever final outputs are either pictures or videos made for human consumptions. This truth permits us to use approximations for raising the speed/energy efficiency. This originates from the restricted sensory activity skills of individuals in perceive a picture or a video. Additionally to the image and video process applications, there are different areas wherever the truth of the arithmetic operations isn't crucial to the performance of the system (see [3], [4]) .The power to use the approximate computing provides the designer with the ability of creating tradeoffs between the accuracy and also the speed additionally as power/energy consumption [2], [5]. Applying the approximation to the arithmetic units are often performed at completely different style abstraction levels together with circuit, logic, and design levels, additionally as algorithmic rule and software system layers [2].

The approximation could also be performed by taking completely different techniques like permitting some temporal order violations (e.g., voltage over scaling or over clocking) and performance approximation strategies (e.g., modifying the Boolean perform of a circuit) or both of them [4], [5]. within the class of perform approximation strategies, variety of approximating arithmetic building blocks, like adders and multipliers, at completely different style levels are instructed (see [6]–[8]). During this paper, we tend to concentrate on proposing a high-speed low power/energy however approximate multiplier applicable for error resilient DSP applications. The proposed approximate multiplier, that is additionally space efficient, is built by modifying the standard multiplication approach at the algorithmic program level assumed as rounded input values. We tend to call as this rounding-based approximate (Roba) multiplier. The proposed multiplication approach is applicable to each signed and unsigned multiplications that 3 optimized architectures are given. The efficiencies of those structures are assessed by examining the delays, power and energy consumptions, energy-delay product (EDPs), and areas with those of some approximate and correct (exact) multipliers. The contributions of this paper will be summarized as follows:

1) Presenting a novel theme for Roba multiplication by modifying the standard multiplication approach;

2) describing three hardware architectures of the proposed approximate multiplication theme for sign and unsigned operations. The remainder of this paper is organized as

follows. Section II discusses the associated works regarding approximate multipliers. The proposed theme of the approximate multiplication, its hardware implementations, and its accuracy results are given in Section III. In Section IV, the characteristics of the proposed approximate multiplier compared with the exact and approximate multipliers, and additionally its effectiveness in image process applications studied. Finally, the conclusion is drawn in section V.

2. LITERATURE SURVEY

In this section, a number of the previous works within the field of approximate multiplier are briefly reviewed. In [3], an approximate multiplier and an approximate adder found a method named broken-array multiplier (BAM) was the multiplier. By applying the BAM approximation methodology of [3] to the standard modified Booth multiplier factor, an approximate signed Booth multiplier was given in [5]. The approximate multiplier provided power consumption savings kind 28th to 58.6% and space reductions from 19.7% to 41.8% for various word lengths as compared with multiplier Booth 31. Kulkarni et al. [6] advised an approximate multiplier consisting of a variety of 2×2 inaccurate building blocks that saved 31.8%–45.4% over an exact multiplier. An approximate signed 32-bit multiplier for speculation functions in pipelined processors was designed in [7]. it had been two hundred times quicker than a full-adder-based tree multiplier whereas having a chance of error of around 14%. In [8], an error-tolerant multiplier, that computed the approximate result by dividing the multiplication into one exact and one approximate part, was introduced, in which the accuracies for various bit widths were reportable. In the case of a 12-bit multiplier, a power saving of over 50% was reported. In [9], two approximate 4:2 compressors for utilizing in a very regular Dadda multiplier were designed and analyzed.

The utilization of approximate multipliers in image process applications that results in reductions in power consumption, delay, and semiconductor unit count compared with those of a particular multiplier style has been mentioned within the literature. In [10], an accuracy-Configurable multiplier design (ACMA) was advised for error-resilient systems to extend its output, the ACMA created use of a method referred to as carry-in prediction that worked supported a precipitation logic. Compared with the precise one, the projected approximate multiplication resulted in nearly 50% reduction within the latency by reducing the important path. Also, Bhardwaj et al. [11] had given an approximate Wallace tree multiplier (AWTM). Again, it invoked the carry-in prediction for the reduction of the vital path. In this work, AWTM was utilized in a time period benchmark image application showing concerning 40% and 30% reductions within the power and space, severally, without any image quality loss compared with the case of using a multiplier Wallace tree multiplier factor (WTM) structure. In [12], approximate unsigned multiplication and division supported an approximate log of the operands are proposed. Within the proposed multiplication, the summation of the approximate logarithms determines the results of the operation. Hence, the multiplication is simplified to some shift and adds operations. In [13], a technique for increasing the accuracy of the multiplication approach of [12] was it had been. it absolutely was it was the decomposition of the input operands. This methodology significantly improved the common error at the worth of increasing the hardware of the approximate multiplier factor by based on twice.

In [16], a dynamic section methodology (DSM) is given, that performs the multiplication operation on an m-bit segment starting from the leading one bit of the input operands. A dynamic vary unbiased multiplier factor (DRUM) multiplier, that selects an m-bit bit of ranging from the leading one little bit of} the input operands and sets the smallest amount significant bit of the truncated values to 1, has been proposed in [17]. In this structure, the truncated values are increased and shifted to left to come up with the final output. In [18], an approximate 4×4 WTM has been projected that uses an inaccurate 4:2 counter. Additionally, miscalculation correction unit for correcting the outputs has been advised. To construct larger multipliers, this 4×4 inaccurate Wallace multiplier may be utilized in an array structure. Most of the previous proposed approximate multipliers are depends on either modifying the structure or complexness reduction of a specific accurate multiplier. In this paper, similar to [12], we have a tendency to propose activity the approximate multiplication through simplifying the operation. The distinction between our work and [12] is that, though the principles in each works are nearly similar for unsigned numbers, the mean error of our proposed approach is smaller. Additionally, we propose some approximation techniques once the multiplication is performed for signed numbers.

3. PROPOSED APPROXIMATE MULTIPLIER

3.1Multiplication rule of Roba multiplier

The main plan behind the proposed approximate multiplier is to create use of the benefit of operation whenever the numbers are 2 to the power n (2n). To elaborate on the operation of the approximate multiplier first, allow us to denote the rounded numbers of the input of A and B by Ar and Br, individually. The multiplication of A by B is also rewritten as

$$A \times B = (Ar - A) \times (Br - B) + Ar \times B + Br \times A - Ar \times Br. \quad (1)$$

The key observation is that the multiplications of Ar ×Br, Ar ×B, and Br ×A could also be enforced simply by the shift operation. The hardware implementation of

(Ar −A) 8× (Br −B), however, is very complicated. the weight of this term within the final result, that depends on variations of the precise numbers from their rounded ones, is usually little. Hence, we tend to propose to omit this part from (1), serving to modify the multiplication operation. Hence, to perform the multiplication method, the subsequent expression is used:

$$A \times B \sim = Ar \times B + Br \times A - Ar \times Br. \quad (2)$$

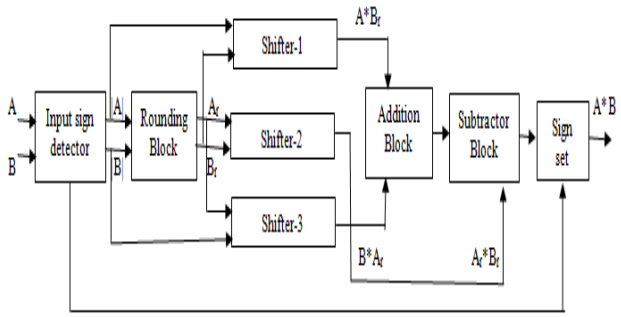

Fig. 1. Block diagram for the hardware implementation of the proposed multiplier

Thus, one will perform the multiplication operation using three shift and two addition/subtraction operations. during this approach, the closest values for A and B within the kind of 2n ought to be determined. When the worth of A (or B) is adequate the 3 × 2p−2 (where p is an absolute positive number larger than one), it's 2 nearest values within the kind of 2n with equal absolute variations that are 2p and 2p−1. Whereas each values result in constant impact on the accuracy of the proposed multiplier, choosing the larger one (except for the case of p = 2) ends up in a smaller hardware implementation for determining the closest rounded worth, and hence, it is considered in this paper.

It originates from the actual fact that the numbers within the sort of 3×2p−2 are thought of as don't care in each rounding error up and down simplifying the method, and smaller logic expressions is also achieved if they're utilized in the rounding error up. The main exception is for 3, that during this case, two is taken into account as its nearest price within the proposed approximate multiplier. It ought to be noted that contrary to the previous work wherever the approximate result's smaller than the precise result, the final result calculated by the Roba multiplier is also either larger or smaller than the precise result counting on the magnitudes of Ar and Br compared with those of A and B, severally. Note that if one amongst the operands (say A) is smaller than its corresponding rounded price whereas the opposite operand(say B) is larger than its corresponding rounded value, then the approximate result are going to be larger than the precise result. This can be due to the actual fact that, in this case, the multiplication result of (Ar −A) × (Br −B) are going to be negative. Since the distinction between (1) and (2) is exactly this product, the approximate result becomes larger than the precise one. Similarly, if each A and B is larger or each is smaller than Ar and Br, then the approximate result are going to be smaller than the precise result. Finally, it ought to be noted the advantage of the proposed Roba multiplier exists just for positive inputs as a result of within the two's complement illustration, the rounded values of negative inputs aren't within the kind of 2n. Hence, we advise that, before the multiplication operation starts, absolutely the values of each inputs and therefore the output sign of the multiplication result depends on the inputs signs be determined and so the operation be performed for unsigned numbers and, at the last stage, the correct sign be applied to the unsigned result. The hardware implementation of the proposed approximate multiplier is explained next.

3.2 Hardware Implementation of Rounding based approximate multiplier

Primarily based on (2), we provide the block diagram for the hardware implementation of the proposed multiplier in Fig. 1 wherever the inputs are depicted in two's complement format. First, the signs of the inputs are determined, and for every negative value, the absolute value is generated. Next, the rounding block extracts the closest price for every exact value within the type of 2n. It ought to be noted that the bit dimension of the output of this block is n (the most significant little bit of a price of an n-bit range within the two's complement format is zero). To find the closest price of input A , we have a tendency to use the subsequent equation to see every output little bit of the rounding error block:

$$\begin{aligned}
A_r[n-1] &= \overline{A[n-1]} \cdot A[n-2] \cdot A[n-3] \\
&\quad + A[n-1] \cdot \overline{A[n-2]} \\
A_r[n-2] &= (\overline{A[n-2]} \cdot A[n-3] \cdot A[n-4] \\
&\quad + A[n-2] \cdot \overline{A[n-3]}) \cdot \overline{A[n-1]} \\
&\vdots \\
A_r[i] &= (\overline{A[i]} \cdot A[i-1] \cdot A[i-2] + A[i] \cdot \overline{A[i-1]}) \cdot \prod_{i=i+1}^{n-1} \overline{A[i]} \\
&\vdots \\
A_r[3] &= (\overline{A[3]} \cdot A[2] \cdot A[1] + A[3] \cdot \overline{A[2]}) \cdot \prod_{i=4}^{n-1} \overline{A[i]} \\
A_r[2] &= A[2] \cdot \overline{A[1]} \cdot \prod_{i=3}^{n-1} \overline{A[i]} \\
A_r[1] &= A[1] \cdot \prod_{i=2}^{n-1} \overline{A[i]} \\
A_r[0] &= A[0] \cdot \prod_{i=1}^{n-1} \overline{A[i]}. \qquad (3)
\end{aligned}$$

The output of this adder and therefore the results of Ar × Br are the inputs of the subtractor block whose output

is that the definite quantity of the output of the proposed multiplier as a result of Ar and Br are within the style of 2n, the inputs of the subtractor might take one amongst the 3 input patterns .The corresponding output patterns are shown .The kinds of the inputs and output influence us to conceive a fundamental circuit supported the subsequent expression:

Out = (P XOR Z) AND ({(P << 1) XOR (P XOR Z)} or {(P AND Z) << 1}) (4)

Where P is Ar × B + Br × A and Z is Ar × Br. The corresponding circuit for implementing this expression is smaller and quicker than the standard subtraction circuit. Finally, if the sign of the final multiplication result ought to be negative, the output of the subtractor is negated within the sign set block. To negate values, that have the two ought to complement illustration, the corresponding circuit supported ~ X + 1 to be used.

To extend the speed of negation operation, one could skip the incrimination method within the negating section by acceptant to its associated error. As are seen later, the significance of the error decreases because the input widths will increase. In this paper, if the negation is performed precisely (approximately), the implementation is named signed Roba (Roba) number [approximate S-Roba (AS-Roba) multiplier]. Within the case wherever the inputs are forever positive, to extend the speed and lower the consumption of power, the sign detector and sign set blocks are omitted from the design, providing us with the design referred to as unsigned ROBA (U-ROBA) number. during this case, the output dimension of the rounding block is n+1 wherever this bit is decided depending on Ar[n]=A[n−1]•A[n−2]. This can be as a result of within the case of unsigned 11x ...x (where x denotes doesn't care) with the bit dimension of n, its rounding value is 10…0 with the bit dimension of n + 1. Therefore, the input bit dimension of the shifters is n + 1. However, as a result of the maximal quantity of shifting is n − 1, 2n is taken into account for the output bit dimension of the shifters.

3.3 Accuracy of ROBA Multiplier in this section

Inaccuracies of the three architectures mentioned are considerable. The inaccuracies of the U-Roba number and S-Roba number, that originate from omitting the term (Ar−A)× (Br−B) from the correct multiplication of A× B, are identical. Hence, the error is

$$\text{error}(A, B) = \frac{(A_r - A)(B_r - B)}{AB}. \quad (5)$$

Assuming Ar and Br are equating to 2n and 2m, identically, the most error happens once A and B are equating to 3×2n and 3×2m, severally. In this case, each Ar and Br has the most arithmetic distinction from their corresponding inputs. Thus

$$\max\{\text{error}(A,B)\} = \frac{(2^n - 3\times 2^{n-2})(2^m - 3\times 2^{m-2})}{(3\times 2^{n-2})\times(3\times 2^{m-2})} = \frac{1}{9}. \quad (6)$$

In the case of the AS-Roba multiplier factor, the error includes a term because of the approximate negation (approximate negation). Therefore, within the worst case (where each inputs square measure negative), one could acquire the most error form.

$$\text{error}(A,B) = \frac{(\bar{A}_r - \bar{A})(\bar{B}_r - \bar{B})}{AB} + \frac{\bar{A} + \bar{B} + 1}{AB}. \quad (7)$$

Compared with (5), the second term comes from the negation approximation obtained from the following relation:

$$A \times B = (\bar{A}+1)(\bar{B}+1) = \bar{A} + \bar{B} + 1 + \bar{A} \times \bar{B} \approx \bar{A} \times \bar{B} \quad (8)$$

Hence, within the case wherever a minimum of one amongst the inputs is negative, the AS-Roba number error is larger than that of the two other Roba multiplier types. Also, once each of the inputs area unit negative, though the final result are going to be positive, one still has to negate the negative inputs. Depending on this formulation, one amongst the inputs is −1, the most error that is 100 %, occurs. To minimize the most error of this case, one could use a detector to detect the case whenever one amongst the inputs is −1, and bypass the multiplication method and generate the output by negating the opposite input. It's clear that this solution has some delay and power consumption overhead.

Additionally to the most error, the incidence rate of error condition is obtained because the magnitude relation of the maximum error occurrences to the overall number of outputs. This error rate is another accuracy measure parameter. Here, all the input combinations are assumed to occur. Within the case of n-bit U-ROBA multiplier, there are n − 1 cases for every input wherever the rounded worth has the most distinction to the particular number. The error happens once these numbers are the input operands. This corresponds to (n −1)2 cases. Within the case of S-ROBA multiplier, for every operand, there are 2(n−2) cases wherever the rounded operands has the most error. Hence, the same as the U ROBA multiplier, most the error happens once each of the rounded operands have the error that creates the quantity of maximum error incidence adequate (2(n −2))2. Finally, within the case of the AS-ROBA multiplier, as mentioned before, the most error happens once one amongst the inputs is −1. Hence, the quantity of most error occurrences can be equal to 2×2n−1 −1 (2 n −1).

On the other hand, in the cases of the U-Roba and S-Roba multipliers when the absolute value of the input operand of the multiplier is within the variety of the 2m, the output results of the Roba multiplier is exactly noticed.

Hence, the numbers of correct outputs within the cases of the U-Roba multiplier factor and S-Roba multipliers are 2(n+1)2n−(n+1)2 and n2n+2−4n2, respectively. Within the case of the AS-Roba multiplier factor, when both inputs are positive, the multiplier behaves kind of like the other 2 Roba multiplier factor architectures, and hence, when one of the inputs is within the variety of 2m, the output is an exact one. In addition, there are other combination's that results in the correct output. One example of such cases is (A − AR) (¯B − ¯B R) + A = 1. Analytically finding all the combination's with correct (exact) output is too difficult, and hence for the AS-Roba multiplier we have a tendency to use the boundary of the correct output range that's equal to n2n − n2.

4. RESULTS AND DISCUSSION

4.1 Hardware Implementation

To evaluate the efficiency of the proposed multiplier, the three Roba multiplier implementations were compared with some approximate and actual multipliers. Baugh Woolley supported Wallace tree design and Wallace (as an explicit unsigned) multipliers were chosen as exact multipliers. Also, within the case of approximate multipliers, DSM8 [16], DRUM6 [17], and HAAM [18] were chosen. Since [12] has not provided any hardware implementation, we excluded it from this a part of the study. The multipliers were designed with the help of VHDL then synthesized with the help of Synopsys design compiler with the choice of synthesizing with the minimum delay objective beneath a 45-nm technology [14]. Next, the post layout style parameters of the selected multipliers were extracted by exploiting Cadence system-on-chip encounter. The extracted style parameters of those multipliers are reported .It has to be mentioned that in this paper, the voltage was1.1 V (based on the Nan Gate 45-nm technology [14]), whereas the frequency was chosen by the use of reported delay for every multiplier.

The results reveal that the minimum delay, energy, and ADP belong to the U-Roba whereas DSM8 has the simplest power consumption and DRUM8 has the minimum space and PDA. The delay, energy, and ADP of the U-Roba are having 22% (15%), 5% (13%), and 26% (25%) less than to those of DSM8 (DRUM6). In distinction, the power (area and PDA) of DSM8 (DRUM6) is having 18% (57% and 51%) lower. Also, the negation operation results in larger style parameters for S-Roba and AS-Roba compared with those of U-Roba, DSM8, and DRUM6. Also, HAAM has the worst style parameters because of its array structure.

The results additionally indicate that the precise multipliers have considerably larger style parameters compared with those of the proposed U-Roba and AS-Roba. Within the case of the S-Roba multiplier, the delay is, on average, 3.4% larger than that of the Baugh Woolley multiplier because of the exact negation operation. Apart from the delay parameter, other style parameters of the S-Roba multiplier are much better than those of Bough Woolley multiplier. On the other hand, the power, area, energy, EDP, and organizer of the S-Roba multiplier, are having 47%, 32%, 45%, 43%, and 63%, respectively, less than those of the Woolley multiplier.

Image Processing Applications

To evaluate the feasibleness of the proposed multiplier in real applications, we tend to compare the performances of the ROBA number architectures in two image process applications of smoothing and sharpening with those of the corresponding precise ones. For sharpening, two totally different ways were invoked. Within the first one, every element of the sharp image was extracted from [15].

$$Y(i,j) = 2.X(i+m,J+n) - \frac{1}{273}\sum_{m=-2}^{2}\sum_{n=-2}^{2} X(i+m,j+n)\ .\text{MaskSharpening},1(m+3,n+3) \quad (9)$$

Where the X (i, j) [Y (i, j)] indicates the pixel of the i th row and j th column of input (output) image and Masksharpening1is an n × *n* coefficient sharpening matrix given by

$$\text{Mask}_{\text{sharpening},1} = \begin{bmatrix} 1 & 4 & 7 & 4 & 1 \\ 4 & 16 & 26 & 16 & 4 \\ 7 & 26 & 41 & 26 & 7 \\ 4 & 16 & 26 & 16 & 4 \\ 1 & 4 & 7 & 4 & 1 \end{bmatrix}. \quad (10)$$

In the second method, each output pixel is determined from

$$Y(i,j) = \frac{1}{256}\sum_{m=-2}^{2}\sum_{n=-2}^{2} X(i+m,j+n)\ .\ \text{MaskSharpening},2(m+3,n+3) \quad (11)$$

Where the sharpening matrix is [15]

$$\text{Mask}_{\text{sharpening},2} = \begin{bmatrix} -1 & -4 & -6 & -4 & -1 \\ -4 & -16 & -24 & -16 & -4 \\ -6 & -24 & 476 & -24 & -6 \\ -4 & -16 & -24 & -16 & -4 \\ -1 & -4 & -6 & -4 & -1 \end{bmatrix}. \quad (12)$$

In the case of MaskSharpening,1, all the values of the matrix are positive, and hence, all the three Roba multiplier architectures result in identical where as within the case of the Masksharpening,2, each S-Roba and AS-Roba multipliers is also utilized resulting in completely different image qualities. The sharpened pictures for the second approach once the precise multiplier factor, S-ROBA multiplier factor, and AS-ROBA multiplier factor were used .The betterness of the sharpening method might not be simply recognized by human eyes. Next, we tend to report the height signal/noise ratio (PSNR) and mean structural similarity index metric (MSSIM [20]) of the sharpened photos for the two sharpening matrices. It ought to be noted that the PSNRs are determined with the sharpened image obtained with the help of exact multipliers to the sharpened image obtained with the help of the approximate multiplier structures. Also, the MSSIM values nearer to at least indicate higher qualities for the approximate output image.

As the results show, within the case of the positive numbers, the mean average of PSNR (MSSIM) of the proposed multiplier factor is over 43 dB (0.99). Although, within the case of negative numbers, the quality of the pictures is lower, the PSNRs (MSSIM)in all the cases are over 20 dB (0.91), that is suitable in several applications [15]. In all the benchmarks, the DSM8 provides the very best output quality providing an equivalent performance as that of the precise multiplication have the PSNR values of ∞. The Mitchell multiplier factor supports particularly for the unsigned operation, and hence, its results are reportable just for the primary sharpening algorithmic program. The results reveal the bottom quality for this multiplier. Also, our proposed approximate multiplier yields higher (lower) output PSNR values compared with those of the DRUM6 within the case of the primary (second) sharpening algorithmic program. For the second application of the smoothing, we've got utilized the subsequent equation to see the smoothened output image [15]:

$$Y(i,j) = \frac{1}{60}\sum_{m=-2}^{2}\sum_{n=-2}^{2} X(i+m,j+n) \quad . Mask(m+3,n+3) \qquad (13)$$

Here, again X(i, j) [Y (i, j)] is the pixel of the i th row associated j th column of input (output) image and Mask Smoothing is an n × n constant smoothing matrix given by one

$$\text{Mask}_{\text{smoothing}} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 4 & 4 & 4 & 4 \\ 1 & 4 & 12 & 4 & 7 \\ 1 & 4 & 4 & 4 & 4 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}. \qquad (14)$$

Because each constant is positive, all the three Roba multiplier architectures leads to a similar output image quality. Because the results reveal, all the PSNRs (MSSIMs) are on 40 (0.99) demonstrating little errors for the proposed multiplier. The output quality of the Roba altogether the benchmark pictures is best than those of the DRUM6 and Mitchell multipliers. However, like the sharpening application, the DSM8 number provides the best output quality.

A. Proposed System Results:

rr Project Status (05/02/2018 - 11:05:02)			
Project File:	rr.xise	**Parser Errors:**	No Errors
Module Name:	rounding_mul_design	**Implementation State:**	Synthesized
Target Device:	xa6slx4-3csg225	**• Errors:**	No Errors
Product Version:	ISE 14.7	**• Warnings:**	5 Warnings (5 new)
Design Goal:	Balanced	**• Routing Results:**	
Design Strategy:	Xilinx Default (unlocked)	**• Timing Constraints:**	
Environment:	System Settings	**• Final Timing Score:**	

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice LUTs	112	2400	4%
Number of fully used LUT-FF pairs	0	112	0%
Number of bonded IOBs	145	132	109%
Number of DSP48A1s	3	8	37%

Fig2. Design summary of proposed system

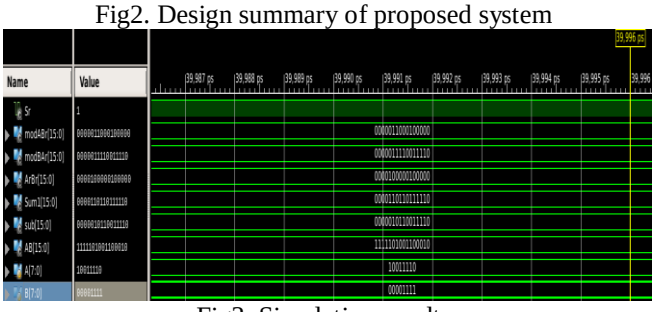

Fig3. Simulation results

```
Timing Summary:
---------------
Speed Grade: -3

   Minimum period: No path found
   Minimum input arrival time before clock: No path found
   Maximum output required time after clock: No path found
   Maximum combinational path delay: 23.949ns

Timing Details:
---------------
All values displayed in nanoseconds (ns)

=========================================================================
Timing constraint: Default path analysis
  Total number of paths / destination ports: 18086562 / 129
-------------------------------------------------------------------------
Delay:               23.949ns (Levels of Logic = 34)
  Source:            B<0> (PAD)
  Destination:       AB<15> (PAD)
```

Fig4. Synthesis report

Table 1.Comparison Table

Parameter	Existing System	Proposed System
Delay(ns)	38.26	23.94
No. of LUTs	437	112
No. of I/o s	163	145

5. CONCLUSION

In this paper, we tend to plan a high speed and energy economical approximate multiplier factor referred to as Roba multiplier factor. The proposed multiplier that had high accuracy was depending on the rounding error of the inputs in 2n. During this approach, the process intensive a

part of the multiplication was omitted rising speed and energy consumption at the worth of a tiny low error. The proposed approach was applicable to each signed and unsigned multiplications. Three hardware implementations of the approximate multiplier including both one for the unsigned and two for the signed operations were mentioned. The efficiencies of the proposed multipliers were evaluated by equivalent with those of some correct and approximate multipliers using totally different style parameters. The results discovered that, in most (all) cases, the Roba multiplier architectures outperformed the corresponding approximate (exact) multipliers. Also, the efficiency of the proposed approximate multiplication approach was studied in two image process applications of sharpening and smoothing. The comparison discovered identical image qualities as those of accurate multiplication algorithms.

REFERENCES

[1] M. Alioto, Ultra-low power VLSI circuit design demystified and explained: A tutorial, IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 59, no. 1, pp. 329, Jan. 2012.

[2] V. Gupta, D. Mohapatra, A. Raghunathan, and K. Roy, Low-power digital signal processing using approximate adders, IEEE Trans. Comput.-Aided design Integr. Circuits Syst., vol. 32, no. 1, pp. 124137, Jan. 2013.

[3] R.Venkatesan, A. Agarwal, K. Roy, and A.Raghunathan, MACACO: M0deling and analysis of circuits for approximate computing, in Proc. Int. Conf. Comput.-Aided style, Nov. 2011, pp. 667673.

[4] F. Farshchi, M. S. Abrishami, and S. M. Fakhraie, New approximate multiplier for low power digital signal processing, in Proc. 17th Int. Symp. Comput. Archit. Digit. Syst. (CADS), oct. 2013, pp. 25-30.

[5] P. Kulkarni, P. Gupta, and M. Ercegovac, Trading accuracy for power with an under designed multiplier architecture, in Proc. 24th Int. Conf. VLSI design, Jan. 2011, pp. 346-351.

[6] A. Momeni, J. Han, P. Montuschi, and F. Lombardi, "Design and analysis of approximate compressors for multiplication," *IEEE Trans. Comput.*, vol. 64, no. 4, pp. 984–994, Apr. 2015.

[7] A. B. Kahng and S. Kang, "Accuracy-configurable adder for approximate arithmetic designs," in *Proc. 49th Design Autom. Conf. (DAC)*, Jun. 2012, pp. 820–825.

[8] K. Bhardwaj, P. S. Mane, and J. Henkel, "Power- and area-efficient approximate wallace tree multiplier for error-resilient systems," in *Proc. 15th Int. Symp. Quality Electron. Design (ISQED)*, 2014, pp. 263–269.

[9] V. Mahalingam and N. Ranganathan, "Improving accuracy in Mitchell's logarithmic multiplication using operand decomposition," *IEEE Trans. Comput.*, vol. 55, no. 12, pp. 1523–1535, Dec. 2006.

[10] S. Hashemi, R. I. Bahar, and S. Reda, "DRUM: A dynamic range unbiased multiplier for approximate applications," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Austin, TX, USA, 2015,pp. 418–425

[11]H. R. Mahdiani, A. Ahmadi, S. M. Fakhraie, and C. Lucas, "Bio-inspired imprecise computational blocks for efficient VLSI implementation of soft-computing applications," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 57, no. 4, pp. 850–862, Apr. 2010.

[12] D. R. Kelly, B. J. Phillips, and S. Al-Sarawi, "Approximate signed binary integer multipliers for arithmetic data value speculation," in *Proc. Conf. Design Archit. Signal Image Process.*, 2009, pp. 97–104.

[13] K. Y. Kyaw, W. L. Goh, and K. S. Yeo, "Low-power high-speed multiplier for error-tolerant application," in *Proc. IEEE Int. Conf. Electron Devices Solid-State Circuits (EDSSC)*, Dec. 2010, pp. 1–4.

[14] K. Bhardwaj and P. S. Mane, "ACMA: Accuracy-configurable multiplier architecture for error-resilient system-on-chip," in *Proc. 8th Int. Workshop Reconfigurable Commun.-Centric Syst.-Chip*, 2013, pp. 1–6.

[15] J. N. Mitchell, "Computer multiplication and division using binary logarithms," *IRE Trans. Electron. Comput.*, vol. EC-11, no. 4, pp. 512–517, Aug. 1962

[16] *Nangate 45nm Open Cell Library*, accessed on 2010. [Online]. Available: http://www.nangate.com/

[17] H. R. Myler and A. R. Weeks, *The Pocket Handbook of Image Processing Algorithms in C*. Englewood Cliffs, NJ, USA: Prentice-Hall, 2009.

[18] S. Narayanamoorthy, H. A. Moghaddam, Z. Liu, T. Park, and N. S. Kim, "Energy-efficient approximate multiplication for digital signal processing and classification applications," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 23, no. 6, pp. 1180–1184, Jun. 2015.

[19] C.-H. Lin and I.-C. Lin, "High accuracy approximate multiplier with error correction," in *Proc. 31st Int. Conf. Comput. Design (ICCD)*, 2013,
pp. 33–38.

[20] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, Apr. 2004.

[21] J. Liang, J. Han, and F. Lombardi, "New metrics for the reliability of approximate and probabilistic adders," *IEEE Trans. Compute.*, vol. 62, no. 9, pp. 1760–1771, Sep. 2013.

Authors:

This is A. Sai Sankalpa pursuing Master of technology in Very Large Scale Integration (VLSI) in Narayana Engineering College, Nellore and I had completed bachelor's degree in the stream of Electronics and Communication Engineering in Andhra Engineering College, Atmakur. I have Certification of Appreciation on" Signal Processing and its Applications using MATLAB" Organized by Andhra Engineering College, Atmakur

C.Leela Mohan working as Associate professor in the Dept of ECE, since four year six Months and also worked as in charge HOD, Library In-Charge Officer (LIO) and Academic in Charge for Narayana Engineering College, Nellore. I have published around 15 papers in various International journals and also attended 3 International & 2 national conferences as well as many workshops'. Currently pursuing Ph.D from KL Deemed to be University. So for guided 20 UG Students and 10 PG Students.