INTERNATIONAL JOURNAL OF RECENT TRENDS IN ENGINEERING & RESEARCH
Impact Factor 4.101 ISSN -2455-1457
RECENT ADVANCES IN ELECTRONICS & COMMUNICATION ENGINEERING (RAECE-2K17)

Design an Approximate ROBA Multiplier for Digital Signal Processing Applicatons

KONTU VENUKUMAR1, Y DEVENDAR REDDY2
PG Scholar, Dept of ECE, Nalla Narasimha Reddy Group of Institution, JNTUH, Hyderabad, TS, India,
Associate Professor, Dept of ECE, Nalla Narasimha Reddy Group of Institution, JNTUH, Hyderabad, TS, India,

Abstract- The need to support various digital signal processing (DSP)and classification applications on energy-constrained devices has steadily grown. Such applications often extensively perform matrix multiplications using fixed-point arithmetic while exhibiting tolerance for some computational errors. Hence, improving the energy efficiency of multiplications is critical. In this paper, we offer a similar speed, but with energy efficiency. The method is to collect the armature close to the closest momentum of two. An integral part of the computer, so the multiplication is eliminated, improving the speed and power consumption at a small error value. The proposed approach is to apply both signed and neglected. We offer three hardware implementations of an approximate multiplier that includes not being signed and signed for both operations. The effectiveness of this proposed multiplier is estimated by comparing its effectiveness with certain approximate and real-world by using different design parameters. In addition, the effect of the proposed approximate multipliers is examined in two applications for image processing, namely sharpness of the image.

Keywords- Approximate multiplier, Energy efficiency and Power consumption ,integrated circuits, DSP

I. BACKGROUND

Energy minimization is one of the main design requirements in almost any electronic systems, especially the portable ones such as smart phones, tablets, and different gadgets [1]. It is highly desired to achieve this minimization with minimal performance (speed) penalty [1]. Therefore, improving the speed and power/energy-efficiency characteristics of multipliers plays a key role in improving the efficiency of processors. Many of the DSP cores implement image and video processing algorithms where final outputs are either images or videos prepared for human consumptions. This fact enables us to use approximations for improving the speed/energy efficiency.

This originates from the limited perceptual abilities of human beings in observing an image or a video. In addition to the image and video processing applications, there are other areas where the exactness of the arithmetic operations is not critical to the functionality of the system (see [3], [4]). Being able to use the approximate computing provides the designer with the ability of making tradeoffs between the accuracy and the speed as well as power/energy consumption [2], [5]. Applying the approximation to the arithmetic units can be performed at different design abstraction levels including circuit, logic, and architecture levels, as well as algorithm and software layers [2]. The approximation may be performed using different techniques such as allowing some

timing violations (e.g., voltage over scaling or over clocking) and function approximation methods (e.g., modifying the Boolean function of a circuit) or a combination of them [4], [5]. In the category of function approximation methods, a multiplier of approximating arithmetic building blocks, such as adders and multipliers, at different design levels have been suggested (see [6]–[8]). In this paper, we focus on proposing a high-speed low power/ energy yet approximate multiplier appropriate for error resilient DSP applications.

II. LITERATURE SURVEY

In this section, some of the previous works in the field of approximate multipliers are briefly reviewed. In [3], an approximate multiplier and an approximate adder based on a technique named broken-array multiplier (BAM) were proposed. By applying the BAM approximation method of [3] to the conventional modified Booth multiplier, an approximate signed Booth multiplier was presented in [5]. The approximate multiplier provided power consumption savings from 28% to 58.6% and area reductions from 19.7% to 41.8% for different word lengths in comparison with a regular Booth multiplier.

Kulkarni *et al*. [6] suggested an approximate multiplier consisting of a multiplier of 2 × 2 inaccurate building blocks that saved the power by 31.8%–45.4% over an accurate multiplier. An approximate signed 32-bit multiplier for speculation purposes in pipelined processors was designed in [7]. It was 20% faster than a full-adder-based tree multiplier while having a probability of error of around 14%.

The use of approximate multipliers in image processing applications, which leads to reductions in power consumption, delay, and transistor count compared with those of an exact multiplier design, has been discussed in the literature. In [10], an accuracy-configurable multiplier architecture (ACMA) was suggested for error-resilient systems. To increase its throughput, the ACMA made use of a technique called carry-in prediction that worked based on a precomputation logic. When compared with the exact one, the proposed approximate multiplication resulted in nearly 50% reduction in the latency by reducing the critical path. Also, Bhardwaj *et al*. [11] presented an approximate Wallace tree multiplier (AWTM). Again, it invoked the carry-in prediction to reduce the critical path.

III. PROPOSED MULTIPLIER

3.1 EXISTING APPROACH

Most of the previously proposed approximate multipliers are based on either modifying the structure or complexity reduction of a specific accurate multiplier. In this project, similar to [12], we propose performing the approximate multiplication through simplifying the operation. The difference between our work and [12] is that, although the principles in both works are almost similar for unsigned multiplier s, the mean error of our proposed approach is smaller. In addition, we suggest some approximation techniques when the multiplication is performed for signed multipliers.

3.2 PROPOSED APPROACH

The proposed approximate multiplier, which is also area efficient, is constructed by modifying the conventional multiplication approach at the algorithm level assuming rounded input values. We call this rounding-based approximate (RoBA) multiplier. The proposed multiplication approach is applicable to both signed and unsigned multiplications for which three optimized architectures are presented. The efficiencies of these structures are assessed by

comparing the delays, power and energy consumptions, energy-delay products (EDPs), and areas with those of some approximate and accurate (exact) multipliers. The contributions of this project can be summarized as follows:

1) Presenting a new scheme for RoBA multiplication by modifying the conventional multiplication approach;

2) Describing three hardware architectures of the proposed approximate multiplication scheme for sign and unsigned operations.

A. Multiplication Algorithm of RoBA Multiplier

The main idea behind the proposed approximate multiplier is to make use of the ease of operation when the numbers are two to the power n (2n). To elaborate on the operation of the approximate multiplier, first, let us denote the rounded numbers of the input of A and B by Ar and Br , respectively. The multiplication of A by B may be rewritten as

$$A \times B = (A_r - A) \times (B_r - B) + A_r \times B + B_r \times A - A_r \times B_r. \quad \text{.....(1)}$$

The key observation is that the multiplications of Ar × Br , Ar×B, and Br×A may be implemented just by the shift operation. The hardware implementation of (Ar − A)×(Br − B), however, is rather complex. The weight of this term in the final result, which depends on differences of the exact numbers from their rounded ones, is typically small. Hence, we propose to omit this part from (1), helping simplify the multiplication operation. Hence, to perform the multiplication process, the following expression is used:

$$A \times B \cong A_r \times B + B_r \times A - A_r \times B_r. \quad \text{.....(2)}$$

Thus, one can perform the multiplication operation using three shift and two addition/subtraction operations. In this approach, the nearest values for A and B in the form of 2n should be determined. When the value of A (or B) is equal to the 3 × 2p−2 (where p is an arbitrary positive integer larger than one), it has two nearest values in the form of 2n with equal absolute differences that are 2p and 2p−1. While both values lead to the same effect on the accuracy of the proposed multiplier, selecting the larger one (except for the case of p = 2) leads to a smaller hardware implementation for determining the nearest rounded value, and hence, it is considered in this project. It originates from the fact that the numbers in the form of 3 × 2*p*−2 are considered as do not care in both rounding up and down simplifying the process, and smaller logic expressions may be achieved if they are used in the rounding up.

The only exception is for three, which in this case, two is considered as its nearest value in the proposed approximate multiplier. It should be noted that contrary to the previous work where the approximate result is smaller than the exact result, the final result calculated by the RoBA multiplier may be either larger or smaller than the exact result depending on the magnitudes of *Ar* and *Br* compared with those of *A* and *B*, respectively.

Note that if one of the operands (say *A*) is smaller than its corresponding rounded value while the other operand (say *B*) is larger than its corresponding rounded value, then the approximate result will be larger than the exact result. This is due to the fact that, in this case, the multiplication result of *(Ar − A)×(Br − B)* will be negative. Since the difference between (1) and (2) is precisely this product, the approximate result becomes larger than the exact one. Similarly, if both *A* and *B* are larger or both are smaller than *Ar* and *Br*, then the approximate result will be smaller than the exact result. Finally, it should be noted the advantage of the proposed RoBA multiplier exists only for positive inputs because in

the two's complement representation, the rounded values of negative inputs are not in the form of 2*n*.

Hence, we suggest that, before the multiplication operation starts, the absolute values of both inputs and the output sign of the multiplication result based on the inputs signs be determined and then the operation be performed for unsigned numbers and, at the last stage, the proper sign be applied to the unsigned result. The hardware implementation of the proposed approximate multiplier is explained next.

B. Hardware Implementation of RoBA Multiplier

Based on (2), we provide the block diagram for the hardware implementation of the proposed multiplier in Fig. 1 where the inputs are represented in two's complement format. First, the signs of the inputs are determined, and for each negative e value, the absolute value is generated. Next, the rounding block extracts the nearest value for each absolute value in the form of 2*n*.

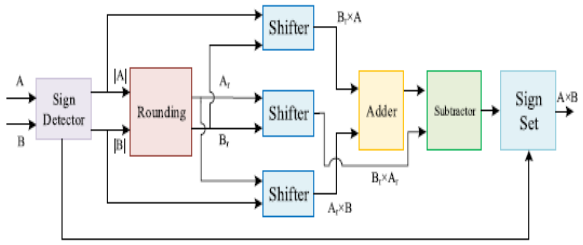

Fig.1. Block diagram for the hardware implementation of the proposed multiplier.

It should be noted that the bit width of the output of this block is *n* (the most significant bit of the absolute value of an *n*-bit number in the two's complement format is zero). To find the nearest value of input *A*, we use the following equation to determine each output bit of the rounding block:

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

In the proposed equation, $Ar\ [i\ ]$ is one in two cases. In the first case, $A[i\ ]$ is one and all the bits on its left side are zero while $A[i-1]$ is zero. In the second case, when $A[i]$ and all its left-side bits are zero, $A[i-1]$ and $A[i-2]$ are both one. Having determined the rounding values, using three barrel shifter blocks, the products $Ar \times Br$, $Ar \times B$, and $Br \times A$ are calculated. Hence, the amount of shifting is determined based on $\log Ar\ 2 - 1$ (or $\log Br\ 2 - 1$) in the case of *A* (or *B*) operand. Here, the input bit width of the shifter blocks is *n*, while their outputs are 2*n*.

A single 2*n*-bit Kogge-Stone adder is used to calculate the summation of $Ar \times B$ and $Br \times A$. The output of this adder and the result of $Ar \times Br$ are the inputs of the *subtractor* block whose output is the absolute value of the output of the proposed multiplier. Because *Ar* and *Br* are in the form of 2*n*, the inputs of the *subtractor* may take one of the three input patterns shown in Table I. The corresponding output patterns are also shown in Table I.

The forms of the inputs and output inspired us to conceive a simple circuit based on the following expression:

$$\text{out} = (P \text{ XOR } Z) \text{ AND } (\{(P \ll 1) \text{ XOR } (P \text{ XOR } Z)\} \text{ or } \{(P \text{ AND } Z) \ll 1\})$$

Table I: All possible cases for Ar × Br AND Ar × B + Br × A Values

Input 1 (A_r×B+B_r×A)	Input 2 (A_r×B_r)	Output
000...11...xxx	000...10...000	000...01...xxx
000...11...xxx	000...01...000	000...10...xxx
000...10...xxx	000...01...000	000...01...xxx

where P is $Ar \times B + Br \times A$ and Z is $Ar \times Br$. The corresponding circuit for implementing this expression is smaller and faster than the conventional subtraction circuit. Finally, if the sign of the final multiplication result should be negative, the output of the subtractor will be negated in the *sign set* block. To negate values, which have the two's complement representation, the corresponding circuit based on $\bar{X}+1$ should be used. To increase the speed of negation operation, one may skip the incrementation process in the negating phase by accepting its associated error. As will be seen later, the significance of the error decreases as the input widths increases. In this project, if the negation is performed exactly (approximately), the implementation is called signed RoBA (S-RoBA) multiplier [approximate S-RoBA (AS-RoBA) multiplier].

In the case where the inputs are always positive, to increase the speed and reduce the power consumption, the *sign detector* and *sign set* blocks are omitted from the architecture, providing us with the architecture called unsigned RoBA (U-RoBA) multiplier. In this case, the output width of the *rounding* block is $n+1$ where this bit is determined based on $Ar[n] = A[n-1] \cdot A[n-2]$. This is because in the case of unsigned $11x \ldots x$ (where x denotes do not care) with the bit width of n, its rounding value is $10\ldots0$ with the bit width of $n+1$. Therefore, the input bit width of the shifters is $n+1$. However, because the maximum amount of shifting is $n-1$, $2n$ is considered for the output bit width of the shifters.

IV. RESULTS AND DISCUSSION

To assess the viability of the proposed multiplier, the three RoBA multiplier executions were contrasted and some inexact and correct multipliers. Baugh Wooley in view of Wallace tree design (as a correct marked) and Wallace (as a correct unsigned) multipliers were chosen as the correct multipliers. Additionally, on account of inexact multipliers, DSM8, DRUM6, and HAAM were picked. Since [12] has not given any equipment usage, we prohibited it from this piece of the investigation.

The multipliers were actualized utilizing Verilog equipment portrayal dialect and afterward orchestrated utilizing Synopsys outline compiler with the choice of integrating with the base defer objective under a 45-nm innovation.

A. Existing System Results:

Device Utilization Summary				[-]
Logic Utilization	Used	Available	Utilization	Note(s)
Number of 4 input LUTs	436	4,896	8%	
Number of occupied Slices	225	2,448	9%	
Number of Slices containing only related logic	225	225	100%	
Number of Slices containing unrelated logic	0	225	0%	
Total Number of 4 input LUTs	437	4,896	8%	
Number used as logic	436			
Number used as a route-thru	1			
Number of bonded IOBs	32	158	20%	
Average Fanout of Non-Clock Nets	2.78			

Fig.2. Design summary

```
Timing Summary:
---------------
Speed Grade: -4

   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: 38.263ns

Timing Detail:
--------------
All values displayed in nanoseconds (ns)
```

Fig.3. Synthesis report

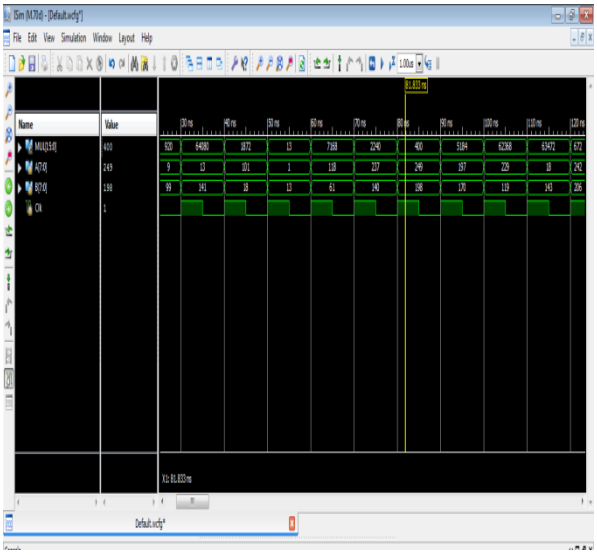

Fig.4. Simulation results

B. Proposed System Results:

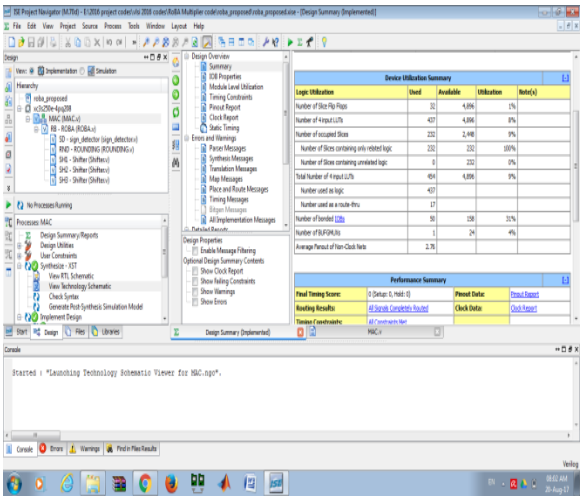

Fig.5. Design summary of proposed system

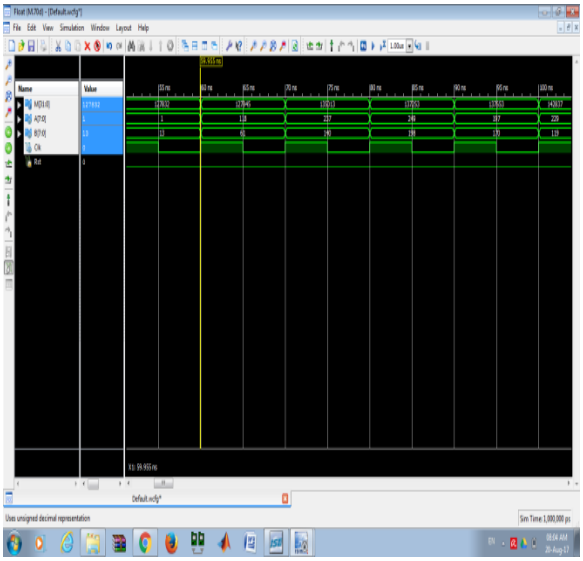

Fig.6. Simulation results

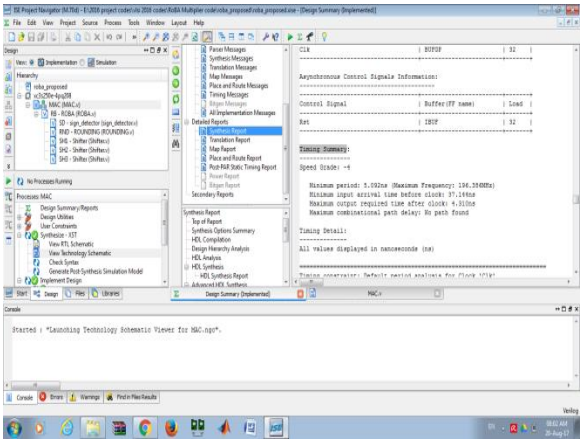

Fig.7. Synthesis report

V. CONCLUSION

In this paper, we proposed a high-speed yet energy efficient approximate multiplier called RoBA multiplier. The proposed multiplier, which had high accuracy, was based on rounding of the inputs in the form of 2*n*. In this way, the computational intensive part of the multiplication was omitted improving speed and energy consumption at the price of a small error. The proposed approach was applicable to both signed and unsigned multiplications. Three hardware implementations of the approximate multiplier including one for the unsigned and two for the signed operations were discussed. The efficiencies of the

proposed multipliers were evaluated by comparing them with those of some accurate and approximate multipliers using different design parameters. The results revealed that, in most (all) cases, the RoBA multiplier architectures outperformed the corresponding approximate (exact) multipliers. Also, the efficacy of the proposed approximate multiplication approach was studied in two image processing applications of sharpening and smoothing. The comparison revealed the same image qualities as those of exact multiplication algorithms.

VI. 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. 3–29, 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. 124–137, Jan. 2013.

[3] 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.

[4] R. Venkatesan, A. Agarwal, K. Roy, and A. Raghunathan, "MACACO: Modeling and analysis of circuits for approximate computing," in *Proc. Int. Conf. Comput.-Aided Design*, Nov. 2011, pp. 667–673.

[5] 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.

[6] P. Kulkarni, P. Gupta, and M. Ercegovac, "Trading accuracy for power with an underdesigned multiplier architecture," in *Proc. 24th Int. Conf. VLSI Design*, Jan. 2011, pp. 346–351.

[7] 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.

[8] 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.

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

[10] 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.

