Approximate computing: A new trend in VLSI based multipliers for error resilient DIP applications

Arya N S[1], Rahul M Nair[2]

[1]*M.Tech.Student, ECE Department, NCERC college, Kerala, India*
[2]*Asst Professor, ECE Department, NCERC College, Kerala, India*

Abstract - *In modern era low power consumption and smaller area are some of the most important criteria for the fabrication of DSP systems and high performance systems. Digital signal processing blocks are key components of these portable devices for realizing various multimedia applications. The computational core of these blocks is arithmetic logic unit where multiplications have greatest share among all arithmetic operation performed in these DSP systems. Optimizing the area of the multiplier is a major design issue. In this paper, propose an approximate multiplier that is high speed and energy efficient. The approach is to round the operands to the nearest exponent of two. This way the computational intensive part of the multiplications omitted improving speed and energy consumption. This is applicable to both signed and unsigned numbers. In addition efficiency of the proposed multiplier is studied in two image processing applications that is image sharpening and smoothing.*

Key Words: **DSP, Multiplier, VLSI, Rounding, Approximation**

1. INTRODUCTION

The International Technology Roadmap for Semi-conductors(ITRS) has anticipated imprecise/ approximate designs that became a state of the art demand for the emerging class of killer applications that manifest inherent error resilience such as multimedia, graphics, and wireless communications. In the error resilience systems, adders and multipliers are used as basic building blocks and their approximate designs have attracted significant research interest recently. Most of the approximate multiplier designs reported shorten the carry chains in which error is configurable and the algorithms employed in the designs are for smaller numbers and give large magnitude of error as the bit width of operands increases. 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. 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 . 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 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 .The approximation may be performed using different techniques such as allowing some timing violations (e.g., voltage over scaling or overclocking) and function approximation methods (e.g., modifying the Boolean function of a circuit) or a combination of them . In the category of function approximation methods, a number of approximating arithmetic building blocks, such as adders and multipliers, at different design levels have been suggested .In this paper, we focus on proposing a high-speed low power/energy yet approximate multiplier appropriate for error resilient DSP applications. 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 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 paper can be summarized as follows: presenting a new scheme for multiplication by modifying the conventional multiplication approach; and describing three hardware architectures of the proposed approximate multiplication scheme for sign and unsigned operations. which can be applicable to special image processing applications like in the field of astronomy and medical such as ECG.

2. APPROXIMATE MULTIPLIER

Approximate circuits have been considered for error-tolerant applications that can tolerate some loss of accuracy with improved performance and energy efficiency. Multipliers are key arithmetic circuits in many such applications such as digital signal processing (DSP). In this paper, a novel approximate multiplier with a lower power consumption and a shorter critical path than traditional multipliers is proposed for high performance DSP applications. 2.1. Rounding based approximation 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 (2^n). 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

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

The key observation is that the multiplications of Ar*Br,A*Br and Ar *Br 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 equation 1, helping simplify the multiplication operation. Hence, to perform the multiplication process, the following expression is used:

$$A*B = Ar * B + Br * A - Ar * B \quad (2)$$

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 2^nshould 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 2Pand 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 paper. It originates from the fact that the numbers in the form of 3* 2p−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. zero).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*Br ,A*Br and Ar *Br are calculated. A single 2n-bit Brent-Kung adder is used to calculate the summation ofAr*Br, A*Br.output of this adder and the result of Ar*Br are the inputs of the sub tractor block whose output is the absolute value of the output of the proposed multiplier. Finally, if the sign of the final multiplication result should be negative, the output of the sub tractor will be negated in the sign set block. To negate values, which have the twos complement representation, the corresponding circuit based on 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. The significance of the error decreases as the input widths increases. In this paper, if the negation is performed exactly(approximately), the implementation is called signed MRoBA (SMRoBA) multiplier [approximate SMRoBA (ASMRoBA) 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 MRoBA (UMRoBA) multiplier. shifted to left to generate the final output, an approximate 44 WTM has been proposed that uses an inaccurate 4:2 counter. In addition, an error correction unit for correcting the outputs has been suggested. To construct larger multipliers, this 44 inaccurate Wallace multiplier can be used in an array structure. Most of the previously proposed approximate multipliers are based on either modifying the structure or complexity reduction of a specific accurate multiplier. In this paper, similar to propose performing the approximate multiplication through simplifying the operation. The difference between my work and others is that, although the principles in both works are almost similar for unsigned numbers, the mean error of our proposed approach is smaller. In addition, we suggest some approximation techniques when the multiplication is performed for signed numbers. Components that are used in the MRoba multiplier implementation are Brent-Kung adder and parallel shifter

2.2. Brent-Kung adder

Thus the proposed full adder shown in Fig. 3 not only reduces area and number of components but delay as well since propagation delay is reduced due to less number of components. Data aware brent kung adder: The Brent-Kung adder is a parallel prefix adder. Parallel prefix adders are special class of adders that are based on the use of generate and propagate signals. Simpler Brent-Kung adders have been proposed to solve the disadvantages of Kogge-Stone adders. The cost and wiring complexity is greatly reduced. But the logic depth of Brent- Kung adders increases to 2log (2n-1), so the speed is lower . We propose a method to reduce delay and power consumed by the Brent Kung adder by analyzing and dividing the inputted data into blocks that will only be added if it holds any value at all. Hence the inputted values are initially compared before deciding up to how many of the adder blocks should be activated. This approach can easily be integrated to the existing design of the Brent Kung adder, thus making it more efficient. Propagate: P0 and P1 Generate: G0 and P1+G1 The second stage calculates both the internal as well as final carry using the initial generate and propagate signals and using the black dot operation as shown in Fig. 4. The Black dot operation takes in 2 pairs of generate.The complete functioning of KSA can be easily comprehended by analyzing it in terms of three

distinct parts :

1) Pre processing

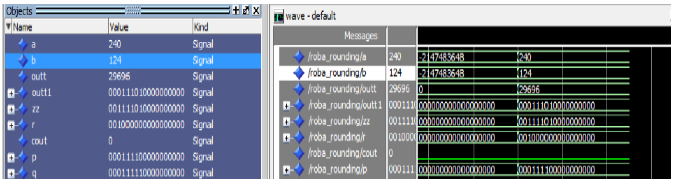
2) Carry look ahead network

3) Post processing

3. RESULTS

To evaluate the efficacy of the proposed multiplier, the three RoBA multiplier implementations were compared with some approximate and one exact multiplier. Baugh Wooley based on Wallace tree architecture (as an exact signed) and Wallace (as an exact unsigned) multipliers were selected as the exact multipliers. The multipliers were implemented using Verilog hardware description language and then synthesized using Modelsim

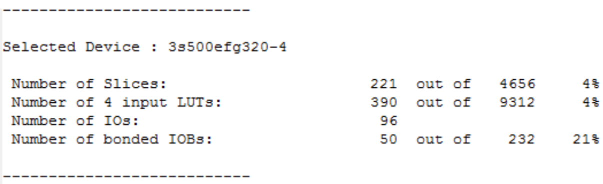
3.1 Existing System



Name	Value	Kind
a	240	Signal
b	124	Signal
outt	29696	Signal
outt1	0001110100000000000	Signal
zz	0011110100000000000	Signal
r	0010000000000000000	Signal
cout	0	Signal
p	0001111000000000000	Signal
q	0001111100000000000	Signal

Messages			
/roba_rounding/a	240	-2147483648	240
/roba_rounding/b	124	-2147483648	124
/roba_rounding/outt	29696	0	29696
/roba_rounding/outt1	000111(	000000000000000000	0001110100000000000
/roba_rounding/zz	001111(	000000000000000000	0011110100000000000
/roba_rounding/r	001000(	000000000000000000	0010000000000000000
/roba_rounding/cout	0		
/roba_rounding/p	000111	000000000000000000	0001111000000000000

Fig -1:Output waveform



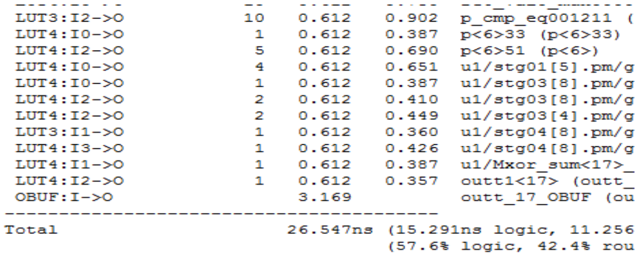
```
---------------------------

Selected Device : 3s500efg320-4

 Number of Slices:                      221  out of   4656     4%
 Number of 4 input LUTs:                390  out of   9312     4%
 Number of IOs:                          96
 Number of bonded IOBs:                  50  out of    232    21%

---------------------------
```

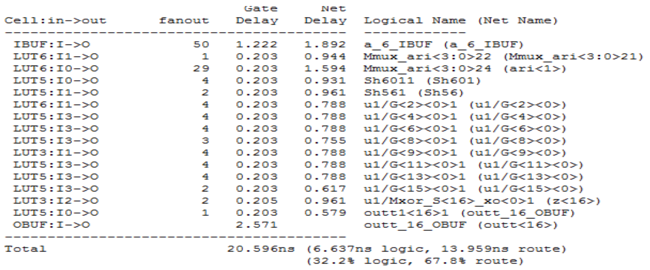
Fig -2: Number of components



```
 LUT3:I2->O           10   0.612   0.902  p_cmp_eq001211 (
 LUT4:I0->O            1   0.612   0.387  p<6>33 (p<6>33)
 LUT4:I2->O            5   0.612   0.690  p<6>51 (p<6>)
 LUT4:I0->O            4   0.612   0.651  u1/stg01[5].pm/g
 LUT4:I0->O            1   0.612   0.387  u1/stg03[8].pm/g
 LUT4:I2->O            2   0.612   0.410  u1/stg03[8].pm/g
 LUT4:I2->O            2   0.612   0.449  u1/stg03[4].pm/g
 LUT3:I1->O            1   0.612   0.360  u1/stg04[8].pm/g
 LUT4:I3->O            1   0.612   0.426  u1/stg04[8].pm/g
 LUT4:I1->O            1   0.612   0.387  u1/Mxor_sum<17>_
 LUT4:I2->O            1   0.612   0.357  outt1<17> (outt_
 OBUF:I->O                 3.169          outt_17_OBUF (ou
----------------------------------------
Total                     26.547ns (15.291ns logic, 11.256
                                   (57.6% logic, 42.4% rou
```

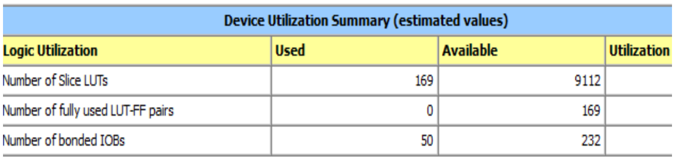
Fig -3:Delay

3.2 Proposed System



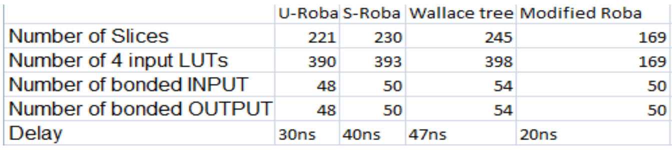
```
                                Gate     Net
Cell:in->out        fanout     Delay    Delay   Logical Name (Net Name)
----------------------------------------   ------------
 IBUF:I->O              50     1.222    1.892   a_6_IBUF (a_6_IBUF)
 LUT6:I1->O              1     0.203    0.944   Mmux_ari<3:0>22 (Mmux_ari<3:0>21)
 LUT6:I0->O             29     0.203    1.594   Mmux_ari<3:0>24 (ari<1>)
 LUT5:I0->O              4     0.203    0.931   Sh6011 (Sh601)
 LUT5:I1->O              2     0.203    0.961   Sh561 (Sh56)
 LUT6:I1->O              4     0.203    0.788   u1/G<2><0>1 (u1/G<2><0>)
 LUT5:I3->O              4     0.203    0.788   u1/G<4><0>1 (u1/G<4><0>)
 LUT5:I3->O              4     0.203    0.788   u1/G<6><0>1 (u1/G<6><0>)
 LUT5:I3->O              3     0.203    0.755   u1/G<8><0>1 (u1/G<8><0>)
 LUT3:I1->O              4     0.203    0.788   u1/G<9><0>1 (u1/G<9><0>)
 LUT5:I3->O              4     0.203    0.788   u1/G<11><0>1 (u1/G<11><0>)
 LUT5:I3->O              4     0.203    0.788   u1/G<13><0>1 (u1/G<13><0>)
 LUT5:I3->O              2     0.203    0.617   u1/G<15><0>1 (u1/G<15><0>)
 LUT3:I2->O              2     0.205    0.961   u1/Mxor_S<16>_xo<0>1 (z<16>)
 LUT5:I0->O              1     0.203    0.579   outt1<16>1 (outt_16_OBUF)
 OBUF:I->O                     2.571            outt_16_OBUF (outt<16>)
----------------------------------------
Total                        20.596ns (6.637ns logic, 13.959ns route)
                                      (32.2% logic, 67.8% route)
```

Fig -4: synthesize report



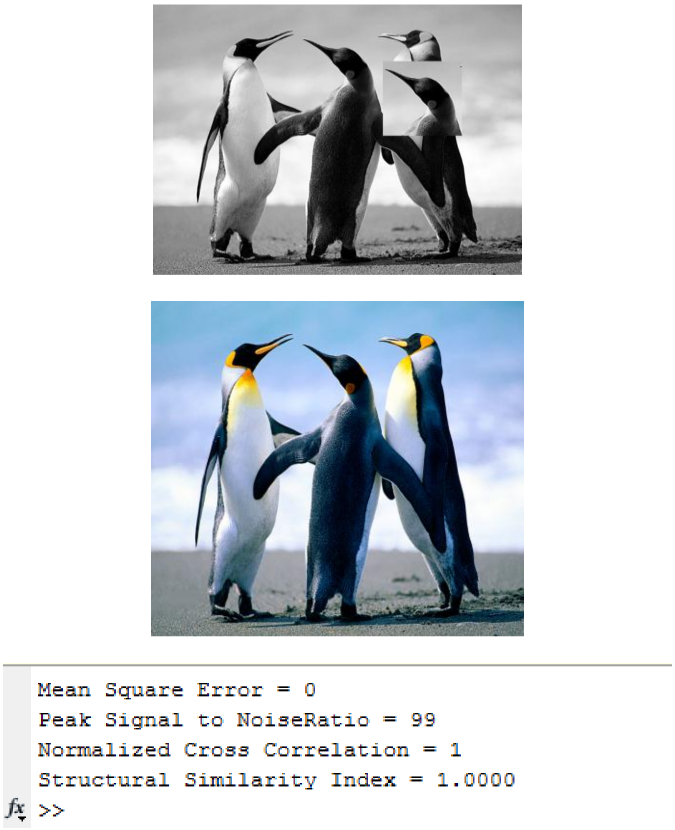
Device Utilization Summary (estimated values)			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice LUTs	169	9112	
Number of fully used LUT-FF pairs	0	169	
Number of bonded IOBs	50	232	

Fig -4: Number of components



	U-Roba	S-Roba	Wallace tree	Modified Roba
Number of Slices	221	230	245	169
Number of 4 input LUTs	390	393	398	169
Number of bonded INPUT	48	50	54	50
Number of bonded OUTPUT	48	50	54	50
Delay	30ns	40ns	47ns	20ns

Table 1. Comparison between existing and proposed multiplier



```
Mean Square Error = 0
Peak Signal to NoiseRatio = 99
Normalized Cross Correlation = 1
Structural Similarity Index = 1.0000
>>
```

Fig -5:Image processing applications

4. CONCLUSION

Multiplication is an important fundamental function in arithmetic logic operation. Computational performance of a DSP system is limited by its performance and since, multiplication dominates the execution time of DSP systems; because of that high speed multiplier is much important. Currently, multiplication time is still the factor for considering DSP performance and determining the instruction cycle time of a DSP chip. Energy minimization is one of the main design requirement in almost any electronic system. In this project propose an approximate multiplier that is high speed yet energy efficient. The approach is to round the operands to the nearest exponent of two. This way the computational intensive part of the multiplication is omitted improving speed and energy consumption at the price of a small error. The proposed approach is applicable to both signed and unsigned multiplications. We propose three hardware implementations of the approximate multiplier that includes one for the unsigned and two for the signed operations. The efficiency of the proposed multiplier is evaluated by comparing its performance with those of some approximate and accurate multipliers using different design parameters. In addition, the efficiacy of the proposed approximate multiplier is studied in two image processing applications, i.e., image sharpening and smoothing.

REFERENCES

[1] Reza Zendegani, Mehdi Kamal, Milad Bahadori, Ali Afzali-Kusha, and Massoud Pedram RoBA Multiplier: A Rounding-Based Approximate Multiplier for High Speed yet Energy Efficient Digital Signal Processing".

[2] J. Liang, J. Han, F. Lombardi, New metrics for the reliability of approximate and probabilistic adders, IEEE Trans. on Computers, vol. 63, no. 9, pp. 1760 - 1771, 2013.

[3] P. Kulkarni, P. Gupta, M. Ercegovac, "Trading accuracy for power with an Underdesigned Multiplier architecture," 24th International Conference on VLSI Design, 2011.

[4] H.R. Mahdiani, A. Ahmadi, S.M. Fakhraie, C. Lucas, "Bio-Inspired imprecise computational blocks for efficient VLSI implementation of soft-computing applications," IEEE Transactions on Circuits and Systems, vol. 57 no. 4, 2010.

[5] K.Y. Kyaw, W.L. Goh, K.S. Yeo, "Low-power high-speed multiplier for error tolerant application," IEEE International Conference of Electron Devices and Solid-State Circuits (EDSSC), 2010.

[6] K. Bhardwaj, P.S. Mane, J. Henkel, "Power- and area-efficient Approximate Wallace Tree Multiplier for error resilient systems," 15th International Symposium on Quality Electronic Design (ISQED), 2014.