Journal of Advanced Research in Microelectronics and VLSI
Volume 4, Issue 2 - 2021, Pg. No. 1-6
Peer Reviewed Journal

Article

A Novel VLSI Design of Efficient Approximate Booth Multiplier

Murali Dova[1], Anuradha M Sandi[2]

[1]Research Scholar, [2]Professor, Department of Electronics and Communication Engineering, Guru Nanak Dev Engineering College, Bidar, Karnataka, India.

I N F O

Corresponding Author:
Anuradha M Sandi, Department of Electronics and Communication Engineering, Guru Nanak Dev Engineering College, Bidar, Karnataka, India.
E-mail Id:
anu29975@gmail.com

How to cite this article:
Dova M, Sandi AM. A Novel VLSI Design of Efficient Approximate Booth Multiplier. *J Adv Res Microelec VLSI* 2021; 4(2): 1-6.

Date of Submission: 2021-12-03
Date of Acceptance: 2021-12-28

A B S T R A C T

This research proposes an approximation multiplier for error-resistant applications that is fast and energy efficient at the same time. Variable probability components are introduced into the multiplier's partial products in this novel design technique for 16-bit approximation of multiplier. The difficulty of approximation is influenced by the possibility of accumulating changing partial products. XILINX synthesis findings show that the suggested multiplier outperforms an exact multiplier in terms of performance. Existing approximation multipliers don't have as much precision as these.

Keywords: Approximate Computing, Error Analysis, Low Error, Low Power, Multipliers

Introduction

Mobile devices like smartphones, tablets, and other tech gizmos all have one common requirement: to use as little power as possible. Minimizing the performance (speed) penalty while still achieving this minimization is a top priority. These portable devices use Digital Signal Processing (DSP) blocks to implement multimedia applications. The arithmetic logic unit is the heart of these blocks, where multiplications account for the majority of all arithmetic operations in these DSP systems. Improved multiplier performance is critical to boosting CPU efficiencies, so speed and power/ energy efficiency may be improved. Images and movies are prepared for human consumption by several of the DSP cores' image and video processing algorithms. As a result of this fact, we are able to improve speed and energy efficiency by using approximations. This is due to the human brain's limited ability to see details in a picture or video. Some additional applications, such as those involving image and video processing, may not necessitate the use of precise arithmetic operations. The ability to employ approximation computation allows the designer to make compromises between accuracy and speed and power/ energy usage. Circuit, logic and architecture levels, as well as algorithm and software layers, can all be used to approximate the arithmetic units while performing approximation operations. You can use a mix of several strategies, such as permitting some timing violations (for example, voltage over scaling or over clocking) and function approximation methods (for example, altering a circuit's Boolean function), to approximate the data. Approximating arithmetic building pieces like adders and multipliers have been proposed under the field of function approximation approaches.

In fixed-width multiplier designs, truncation is frequently used to decrease the complexity of the circuitry. In order to compensate for the quantization error induced by the reduced portion, a constant or variable correction term is added Accumulation of partial products is a key component of multiplier approximation approaches, which is critical in terms of power usage. If the least significant bits of the inputs can be trimmed, then partial products can be formed in order to decrease hardware complexity. In partial product accumulation, the suggested multiplier in reduces the number of adder circuits required.

Literature Review

Low-Power Digital Signal Processing using Approximate Adders

Devices that use signal processing methods and architectures in portable multimedia devices must consume very little energy. A lot of multimedia apps allow humans to learn something even if the outputs aren't exactly perfect. When it comes to numerical outputs, we don't need to get them absolutely right. Previous research in this area relies mostly on voltage over scaling for error resilience, which is mitigated by computational and architectural strategies. As an alternate way to taking advantage of the relaxation of numerical precision, we suggest in this study a decrease in logic complexity at the transistor level. Imperfect or approximation complete adder cells with decreased transistor complexity are proposed in this paper as examples of how this notion might be implemented in the creation of approximate multi-bit adders. To further enable voltage scaling, our solutions result in shorter critical routes, which reduce switching capacitance. Video and image compression methods are designed and evaluated using the suggested approximation arithmetic units to illustrate the effectiveness of our approach. These approximation adders also have basic mathematical models for inaccuracy and power consumption. As a further example, we illustrate the use of these approximation adders in two digital signal processing systems (the discrete cosine transform and the finite impulse response filter). The suggested approximation adders can save up to 69 percent of the power compared to current implementations employing precise adders, according to simulation data.

Changes are made to the variable corrections truncated multiplier A truncated multiplier can be minimized using this way. Information from a column adjacent to the shortened LSB helps to lessen the mistake. This reduces the amount of time and effort needed to produce the final product, while minimizing the amount of distortion.

Exact Multiplier

Three steps are involved in multiplication.

This is a Two Step Process: First, create partial products; then fill in the remaining rows with created partial products.

The third step is to combine the remaining two rows together to get the final multiplication results.

In the first stage of the updated Booth algorithm, the number of partial products is reduced by half. The Modified Booth Encoding (MBE) technique introduced in was employed in our analysis. Known as the most effective Booth encoding and decoding method. For example, to multiply two numbers by three using the modified Booth algorithm, the first step is to divide each number by three. For the MBE system, Table 1, presents the principles for generating encoded signals, along with a logic diagram. The Booth decoder uses the encoded signals to create the partial products as shown.

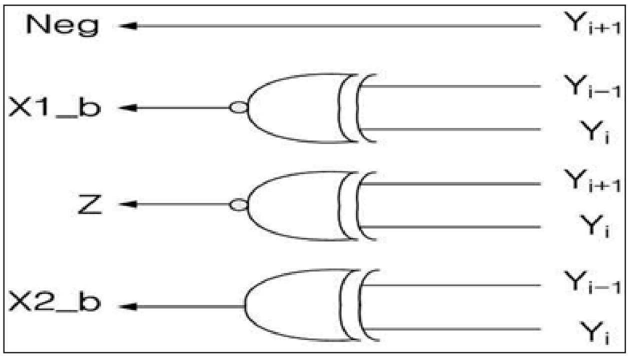

Figure 1.Booth Encoder

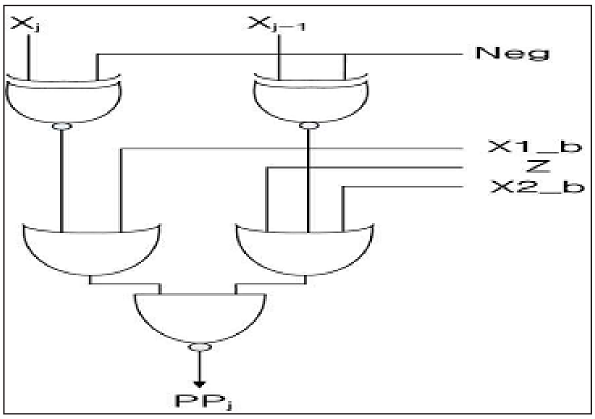

Figure 2.Booth Decoder

Figure 1 and 2, depicts the 8-bit Booth multiplier's produced partial products and sign-extension method. Wallace trees are used to sequentially add partial products from the modified Booth method until the final two rows are left. Last two rows of multiplication are added together to get a final answer. This is when the carry propagation adder comes into play.

Table 1.Truth Table for MBE Scheme

Y_{i+1}	Y_i	Y_{i-1}	Value	X1_b	X2_b	Neg	Z
0	0	0	0	1	0	0	1
0	0	1	1	0	1	0	1
0	1	0	1	0	1	0	0
0	1	1	2	1	0	0	0
1	0	0	-2	1	0	1	0
1	0	1	-1	0	1	1	0
1	1	0	-1	0	1	1	1
1	1	1	0	1	0	1	1

Proposed System

Multiplier implementation has three stages, the formation of partial products, the reduction of partial products, and ultimately, a vector merge addition to produce the final result. More force is needed for the second phase. Approximation is used at the reduction tree step in this short.

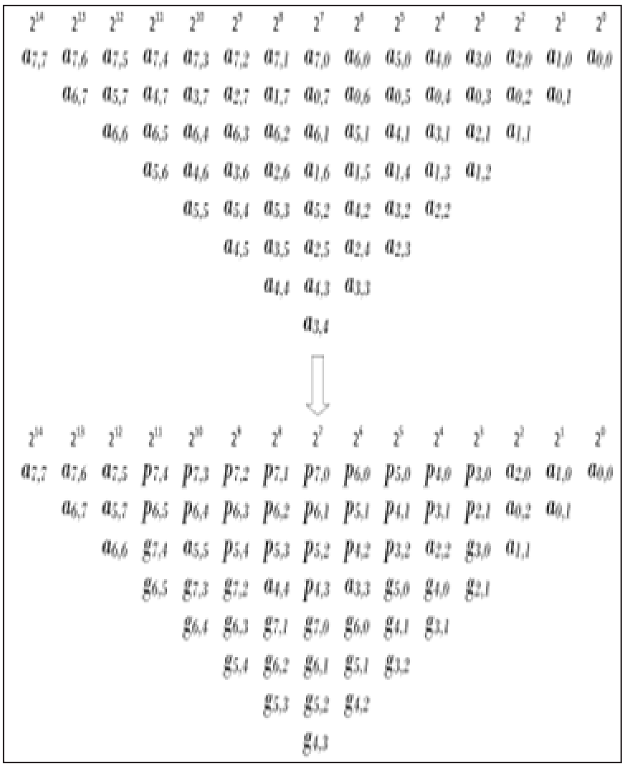

Figure 3.Transformation of Generated Partial Products into Altered Partial Products

Table 2.Probability Statistics of Generate Signals

m	Probability of the *generate* elements being				P_{err}
	all zero	one 1	two 1's	three 1's and more	
2	0.8789	0.1172	0.0039	-	0.00390
3	0.8240	0.1648	0.0110	0.00024	0.01124
4	0.7725	0.2060	0.0206	0.00093	0.02153

A 8-bit unsigned multiplier is used for illustration to describe the proposed method in approximation of multipliers. Consider two 8-bit unsignedinput operands. The partial product a m , n = αm. β n in Figure 3, is the result of AND operation between the bits of α m and βn. From statistical point of view, the partial product a m, n has a probability of 1 / 4 of being 1. In the columns containing more than three partial products, the partial products a m, n and a, n, m are combined to form propagate and generate signals as given in (1). The resulting propagate and generate signals form altered partial products p m, n and g m, n. From column 3 with weight 2^3 to column 11 with weight 2^{11} , the partial products a m, n and a n , m are replaced by altered partial products p m, n and g m , n. The original and transformed partial product matrices are shown in Figure 3.

$$P_{m,n} = a_{m,n} + a_{n,m}$$
$$g_{m,n} = a_{m,n} - a_{n,m}$$

The probability of the altered partial product g m, n being one is 1/ 16, which is significantly lower than 1/ 4 of a m, n. The probability of altered partial product p m, n being one is 1/ 16 + 3/ 16 + 3/ 16 = 7/ 16, which is higher than g m, n. These factors are considered, while applying approximation to the altered partial product matrix.

Approximation of Altered Partial Products g m, n

The accumulation of generate signals is done columnwise. As each element has a probability of 1/ 16 of being one, two elements being 1 in the same column even decreases. For example, in a column with 4 generate signals, probability of all numbers being 0 is $(1 - pr)^4$, only one element being one is $4\,pr\,(1 - pr)^3$, the probability of two elements being one in the column is $6\,pr\,2\,(1 - pr)^2$, three ones is 4 pr 3 (1 - pr) and probability of all elements being 1 is pr 4, where pr is 1/ 16. Table 2, lists the probability statistics for each column's m generated items. When accumulating column-wise produce items in a modified partial product matrix, utilising an OR gate yields precise results in most circumstances, as shown in Table 2.

Table 3.Truth Table of Approximate Half Adder

Inputs		Exact Outputs		Approximate Outputs		Absolute Difference
$x1$	$x2$	*Carry*	*Sum*	*Carry*	*Sum*	
0	0	0	0	0 ✔	0 ✔	0
0	1	0	1	0 ✔	1 ✔	0
1	0	0	1	0 ✔	1 ✔	0
1	1	1	0	1 ✔	1 ✗	1

Table 4.Truth Table of Approximate Full Adder

Inputs			Exact Outputs		Approximate Outputs		Absolute Difference
$x1$	$x2$	$x3$	*Carry*	*Sum*	*Carry*	*Sum*	
0	0	0	0	0	0 ✔	0 ✔	0
0	0	1	0	1	0 ✔	1 ✔	0
0	1	0	0	1	0 ✔	1 ✔	0
0	1	1	1	0	1 ✔	0 ✔	0
1	0	0	0	1	0 ✔	1 ✔	0
1	0	1	1	0	1 ✔	0 ✔	0
1	1	0	1	0	0 ✗	1 ✗	1
1	1	1	1	1	1 ✔	0 ✗	1

Table 2, additionally includes the error probability (P err) for each column when generating signals are reduced using an OR gate. Clearly, there is a very little chance of a forecast going awry. The likelihood of a mistake rising linearly with an increase in the quantity of generated signals. The cost of inaccuracy, on the other hand, grows. To avoid this, the OR gate can only join a maximum of four produce signals together. Using OR gates, you may create a column with m different signals.

The proposed approximate technique can be applied to signed multiplication including Booth multipliers as well, except it is not applied to sign extension bit.

Approximation of Other Partial Products

The accumulation of other partial products with probability 1/ 4 for a m, n and 7/ 16 for p m, n uses approximate circuits. Approximate half-adder, full-adder and 4-2 compressor are proposed for their accumulation. Carry and Sum are two outputs of these approximate circuits. Since Carry has higher weight of binary bit, error in Carry bit will contribute more by producing error difference of two in the output. Approximation is handled in such a way that the absolute difference between actual output and approximate output is always maintained as one. Hence Carry outputs are approximated only for the cases, where Sum is approximated. In adders and compressors, XOR gates tend to contribute to high area and delay.

Table 5.Truth Table of Approximate 4-2 Compressor

Inputs				Approximate outputs		Absolute Difference
$x1$	$x2$	$x3$	$x4$	$Carry$	Sum	
0	0	0	0	0 ✔	0 ✔	0
0	0	0	1	0 ✔	1 ✔	0
0	0	1	0	0 ✔	1 ✔	0
0	0	1	1	1 ✔	0 ✔	0
0	1	0	0	0 ✔	1 ✔	0
0	1	0	1	0 ✗	1 ✗	1
0	1	1	0	0 ✗	1 ✗	1
0	1	1	1	1 ✔	1 ✔	0
1	0	0	0	0 ✔	1 ✔	0
1	0	0	1	0 ✗	1 ✗	1
1	0	1	0	0 ✗	1 ✗	1
1	0	1	1	1 ✔	1 ✔	0
1	1	0	0	1 ✔	0 ✔	0
1	1	0	1	1 ✔	1 ✔	0
1	1	1	0	1 ✔	1 ✔	0
1	1	1	1	1 ✗	1 ✗	1

For approximating half-adder, XOR gate of Sum is replaced with OR gate. This results in one error in the Sum computation as seen in the truth table of approximate half-adder in Table 2. A tick mark denotes that approximate output matches with correct output and cross mark denotes mismatch.

$Sum = x1 + x2$
$Carry = x1 - x2$

In the approximation of full-adder, one of the two XOR gates is replaced with OR gate in Sum calculation. This results in error in last two cases out of eight cases. Carry is modified as in (3) introducing one error. This provides more simplification, while maintaining the difference between original and approximate value as one. The truth Table of approximate full-adder is given in Table 3.

W = $(x1 - x2)$
$Sum = W \oplus x3$
$Carry = W. x3.$

Two approximate 4-2 compressors produce nonzero output even for the cases where all inputs are zero. This results in high ED and high degree of precision loss especially in cases of zeros in all bits or in most significant parts of the reduction tree. The proposed 4-2 compressor overcomes this drawback. In 4-2 compressor, three bits are required for the output only when all the four inputs are 1, which happens only once out of 16 cases.

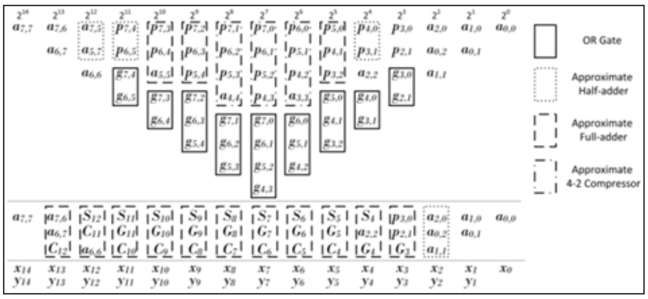

Figure 4.Reduction of Altered Partial Products

This property is taken to eliminate one of the three output bits in 4-2 compressor. To maintain minimal error difference as one, the output "100" (the value of 4) for four inputs being one has to be replaced with outputs "11" (the value of 3). For Sum computation, one out of three XOR gates is replaced with OR gate. Also, to make the Sum corresponding to the case where all inputs are ones as one, an additional circuit x 1 · x 2 · x 3 · x 4 is added to the Sum expression. This results in error in five out of 16 cases. Carry is simplified as in (4). The corresponding truth table is given in Table 5.

$W1 = x1.x2$
$W2 = X3.x4$
$Sum = (x1 \oplus x2) + (x3 \oplus x4) + W1.W2$
$Carry = W1 + W2.$

Figure 4, depicts the reduction of the 8 8 approximation multiplier's revised partial product matrix. The sum and carry outputs for the vector merge addition phase need two steps. There are four 2-input OR gates, four 3-input OR gates and a 4-input OR gate needed to reduce the generated signals from columns 3-11. A signal generated by an OR gate is designated as G I with the weight 2i being assigned to the column I that corresponds to that signal. In order to create the Sum and Carry signals S I and C I that correspond to column I three approximate half-adders, three approximate full-adders and three approximate compressors are needed in the first stage for decreasing other partial products. Two operands x I and y I are produced by reducing the elements in the second stage using one approximate half-adder and eleven approximate full-adders and they are passed to a ripple carry adder for the final result computation.

Two Variants of Multipliers

Two variants of multipliers are proposed. In the first case (Multi-plier 1), approximation is applied in all columns of partial products of n -bit multiplier, whereas in Multiplier 2, approximate circuits are used in n - 1 least significant columns.

Result

The Verilog HDL Modules have successfully simulate, verified and synthesized using Xilinxise 13.2.

Proposed Result

Simulation

Figure 5

Synthesis Results

RTL Schematic

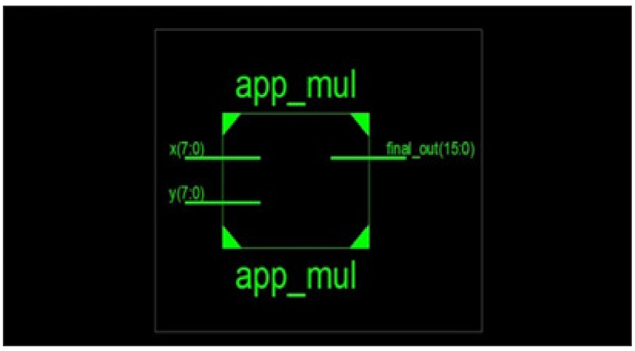

Figure 6

Technology Schematic

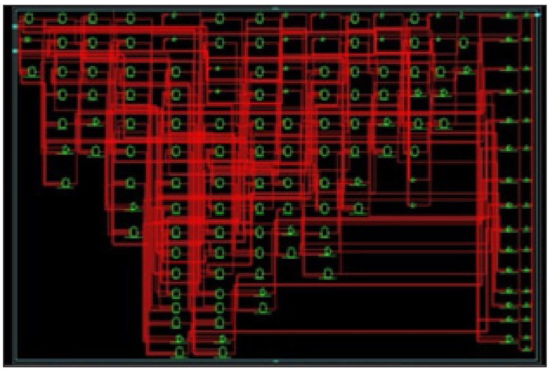

Figure 7

Design Summary

Device Utilization Summary (estimated values)			
Logic Utilization	Used	Available	Utilization
Number of Slices	50	4655	1%
Number of 4 input LUTs	90	9312	0%
Number of bonded IOBs	32	232	13%

Figure 8

Timing Summary

```
================================================================================
Delay:               14.100ns (Levels of Logic = 17)
  Source:            x<1> (PAD)
  Destination:       final_out<14> (PAD)

  Data Path: x<1> to final_out<14>
                                Gate     Net
    Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
    ----------------------------------------  ------------
     IBUF:I->O            10   1.218   1.057  x_1_IBUF (x_1_IBUF)
     LUT4:I0->O            3   0.704   0.706  m3/w4  (m3/w4)
     LUT2:I0->O            2   0.704   0.526  m3/w10  (m3/w)
     LUT4:I1->O            1   0.704   0.595  m14/w21  (m14/w21)
     LUT4:I0->O            2   0.704   0.622  m14/w23  (m14/w)
     LUT3:I0->O            0   0.704   0.000  m14/Mxor_s_Result1  (u<5>)
     MUXCY:DI->O           1   0.888   0.000  Madd_final_out_addsub0000_cy<5>  (Madd_final_out_a
     MUXCY:CI->O           1   0.059   0.000  Madd_final_out_addsub0000_cy<6>  (Madd_final_out_a
     MUXCY:CI->O           1   0.059   0.000  Madd_final_out_addsub0000_cy<7>  (Madd_final_out_a
     MUXCY:CI->O           1   0.059   0.000  Madd_final_out_addsub0000_cy<8>  (Madd_final_out_a
     MUXCY:CI->O           1   0.059   0.000  Madd_final_out_addsub0000_cy<9>  (Madd_final_out_a
     MUXCY:CI->O           1   0.059   0.000  Madd_final_out_addsub0000_cy<10>  (Madd_final_out_
     MUXCY:CI->O           1   0.059   0.000  Madd_final_out_addsub0000_cy<11>  (Madd_final_out_
     MUXCY:CI->O           1   0.059   0.000  Madd_final_out_addsub0000_cy<12>  (Madd_final_out_
     MUXCY:CI->O           1   0.059   0.000  Madd_final_out_addsub0000_cy<13>  (Madd_final_out_
     XORCY:CI->O           1   0.804   0.420  Madd_final_out_addsub0000_xor<14>  (final_out_14_O
     OBUF:I->O                 3.272          final_out_14_OBUF  (final_out<14>)
    ----------------------------------------
    Total                     14.100ns (10.174ns logic, 3.926ns route)
```

Figure 9

Conclusion

Partially modified multiplier products are used in this brief to suggest effective approximation multipliers. Use of an OR gate for changed products generates approximations. Remaining partial products can be reduced by using an approximate half-adder, full-adder and 4-2 compressor. In comparison to previous solutions, the designed approximation multiplier outperforms When compared to previous approximation multiplier designs, they are proven to be more precise.

It is possible to employ the proposed multipliers in applications with minimum loss in output quality while saving substantial power and area.

References

1. Gupta V, Mohapatra D, Raghunathan A et al. Low-power digital signal processing using approximate adders. *IEEE Trans Comput Aided Design Integr Circuits Syst* 2013; 32(1): 124-137.
2. King EJ, Swartzlander EE. Data-dependent truncation scheme for parallel multipliers. Proc. 31st Asilomar Conf. Signals, Circuits Syst. 1998; 1178-1182.
3. Cho KJ, Lee KC, Chung JG et al. Design of low-error fixed-width modified booth multiplier. *IEEE Trans Very Large Scale Integr (VLSI) Syst* 2004; 12(5): 522-531.
4. Mahdiani HR, Ahmadi A, Fakhraie SM et al. Bio-inspired imprecise computational blocks for efficient VLSI implementation of soft-computing applications. *IEEE Trans Circuits Syst* 2010; 57(4): 850-862.
5. Momeni A, Han J, Montuschi P et al. Design and analysis of approximate compressors for multiplication. *IEEE Trans Comput* 2015; 64(4): 984-994.
6. Narayanamoorthy S, Moghaddam HA, Liu Z et al. Energy-efficient approximate multiplication for digital signal processing and classification applications. *IEEE Trans Very Large Scale Integr (VLSI) Syst* 2015; 23(6): 1180-1184.
7. Zervakis G, Tsoumanis K, Xydis S et al. Design-efficient approximate multiplication circuits through partial product perforation. *IEEE Trans Very Large Scale Integr (VLSI) Syst* 2016; 24(10): 3105-3117.
8. Kulkarni P, Gupta P, Ercegovac MD. Trading accuracy for power in a multiplier architecture. *J Low Power Electron* 2011; 7(4): 490-501.
9. Lin CH, Lin C. High accuracy approximate multiplier with error correction. Proc. IEEE 31st Int. Conf. Comput. Design. 2013; 33-38.
10. Liu C, Han J, Lombardi F. A low-power, high-performance approximate multiplier with configurable partial error recovery. Proc. Conf. Exhibit. 2014; 1-4.
11. Venkatesan R, Agarwal A, Roy K et al. MACACO: Modeling and analysis of circuits for approximate computing. Proc. IEEE/ ACM Int. Conf. Comput.-Aided Design (ICCAD). 2011; 667-673.
12. Liang J, Han J, Lombardi F. New metrics for the reliability of approximate and probabilistic adders. *IEEE Trans Comput* 2013; 63(9): 1760-1771.
13. Suman S. Image enhancement using geometric mean filter and gamma correction for WCE iamges. Proc. 21st Int. Conf., Neural Inf. Process. (ICONIP). 2014; 276-283.