International Journal of Research in Advent Technology, Vol.6, No.11, November 2018
E-ISSN: 2321-9637
Available online at www.ijrat.org

Energy-Efficient Approximate Multiplier Design using Bit Significance-Driven Logic Compression

C.Gangaiah Yadav[1], Dr. K.S.Vijula Grace[2]
Research Scholar[1], Assistant Professor[2]
ECE, NIU, Kumaracoil, Tamil Nadu, India[1, 2]
Email: ganga.486@gmail.com[1], vijulaks@yahoo.co.in[2]

Abstract-Approximate arithmetic is one of the recently emerged techniques as a promising paradigm for many imprecision-liberal applications. It can offer considerable reductions in circuit difficulty, delay (late) and energy production by relaxing exact needs. In this paper, we suggest a novel energy-efficient rough multiplier design using a significance-driven logic compression (SDLC) approach. Fundamental to this approach is an algorithmic and architecture damage reduction of the partial product rows based on their progressive bit significance. This is followed by the independent resizing of the out coming product terms to decrease the number of product rows. As such, the difficulty of the multiplier in terms of logic cell computations and lengths of unfavorable paths is desperately decreased. The suggested rough multiplier constructed in Verilog HDL and composited using ISIM simulator in Xilinx ISE 14.3. Additionally, we signify the energy-accuracy trade-offs for different degrees of reduction, attained through fabric/frame gathering. In assessing the success of our approach, a case study of picture handling application exhibited up to 68.3%energy depletion with very minute damages in picture quality expressed as peak signal-to-noise ratio (PSNR).

Index Terms- Approximate Arithmetic's, Significance Driven Logic Compression (SDLC), Logic Clustering

1. INTRODUCTION

There is a determinant request for higher calculation presence at low energy rate for upcoming applications. The basic assumption of precise calculating is to exchange conventional difficulties and energy-uneconomic data handling chunks by low-complexity ones with reduced logic counts. As a result, successive fragment area and energy consumption are decreased at the cost of inaccuracy introduced to the handled data. Investigation has shown that the seniority of modern applications such as digital signal processing, computer vision, automation, multi-media (intermedia handling) and statistical analytics have some level of liberality to such apprehension. This can be griped as a chance for well-organized machine construction for current and future generations of application-specific structures.

Multipliers are the critical arithmetic components in many of these applications, for two important reasons. Firstly, they are specified by complex logic design, being one of the most energy challenging data processing units in modern micro chips. Secondly, compute-intensive applications difficult exercise great number of multiplication performance to calculate outcomes. These elements have induced close awareness in approximate multiplier design investigation, since developments made in the capability of a multiplier are anticipated to considerable crush on overall structure capability. In the territory of rough circuits the multipliers can be enlarge classified as moderations of either timing or functional behaviors. Firstly, timing performance can be improved using assistive contribute voltage measuring developments which requires additional error repayment system to decrease the error estimate. Secondly, useful techniques deal with logic compression abilities and can be operated by relaxing the need for exact Boolean identity in favor of vitality and circuit area compressions. For example, shortening multiplier product terms allows for the eradication of some of the small remarkable partial product terms. As more columns are eradicated, further energy depletion is attained however, faults also developed.

Standard re-arrange with low-difficulty conjunctional logic is another successive skill. This allows for developing larger energy-capability multipliers using small rough/imprecise ones. The key concept of the above studies is to attain compressed logic difficulty, which is also the main goal of our work. A typical (N×N) exact multiplier produces N2 product terms, which are then gathered as a final outcome of size 2N. The exactness of this product depends largely on the importance of bits; conserving higher-significance bits is likely to generate an result closes to the accurate outcome than that of lower-importance bits. In our work, we hold this chance to make the following key offerings:

1) We suggest a novel energy-efficient approximate multiplier design approach using bit significance-driven logic compression (SDLC).

2) At the basics of our approach is a configurable logic gathering of product terms suitably selection for a given energy-accuracy dealing, followed by retrade using their independent belongings to decrease the output number of product terms.

To the best of our knowledge, this is the first validation of a methodolical logic reduction-based estimate multiplier design approach. The rest of the paper is arranged as follows. Section 2 gives the premature work done; Section 3 initiates the suggested rough multiplier construction. And Section 4 provides the fault analysis connected with different bit-widths of the suggested multiplier. Section 5 provides the inventive outcomes. Finally, Section 6 terminates the paper.

2. EARLIER WORK

Imprecise calculating can reduce the construction difficulty with a development in production and ability efficiency for damage flexible appeals**.** This brief deals with a new design approach for proceed towards of multipliers. The partial products of the multiplier are adjusted to initiate differencing possibility terms Logic difficulty of imprecision. Execution of multiplier consists three steps: generation of incomplete products, partial products depletion is varied for the collection of changed partial products based on their possibility. And finally, a vector combines addition to produce final outcome from the sum and carry rows produced from the reduction tree.

2.1 Approximation of Other Partial Products:

The collection of other partial products uses imprecise circuits like approximate half-adder, full-adder, are suggested for their collection. *Carry* and *Sum* are two outcomes of these approximate circuits. Since *Carry* has higher weight of binary bit, error in *Carr y* bit will give more by generating error difference of two in the output. Approximation is hold in such a way that the fixed difference between exact output and rough output is always maintained as one. Hence *Carry* outcomes are estimated only for the cases, where *Sum* is estimated. In adders, XOR gates tend to give to high Area and delay. For estimating half-adder, XOR gate of *Sum* is returned with OR gate as given. This generates an one error in the *Sum* calculation as seen in the Table I of estimate half-adder

$$Sum = x1|x2;$$

$$Carry = x1\&x2;$$

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

TABLEI. Truth table of Approximate Half Adder

In the estimation of full-adder, one of the two XOR gates is returned with OR gate in *Sum* calculation. This generates the error in last two cases out of eight cases. *Carry* is altered as in initiating one error. This generates more easily, while sustaining the difference between indigenous and approximate value as one. The truth table of approximate full-adder is given in Table II.

$$w = x1|x2;$$

$$Sum = w \oplus x3;$$

$$Carry = w\&x3;$$

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 II. Truth table of Approximate Full Adder

An 8-bit estimate multiplier is created by using this estimated half adders and Full adders as shown in fig.1.

3. PROPOSED APPROXIMATE MULTIPLIER

Our suggested estimation composed of two crucial steps. In the first, damage reduction is accomplished out through logic gathering. The resulting reduced terms are then resized using their independent belongings.

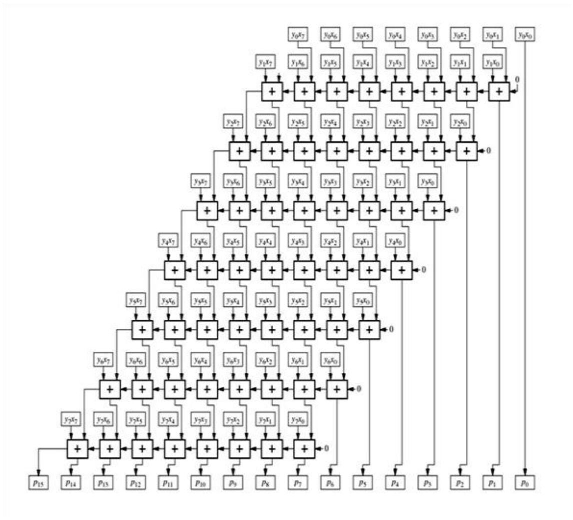


Fig.1.Approximate multiplier with approximate adders

3.1 Logic Compression

Collateral multiplication creation is generally separated into three suggestive phases: partial outcome evolution, assembling, and carry generation adder. In an (N ×N) multiplier, N^2 AND gates are employed in similar to produce the partial product bit-matrix. This matrix is then pillar-wise assembled to produce the final outcome by using carry generation adders the suggested approach begins by producing all partial outcomes using the similar number of AND gates, similar to subsisting multiplication. Before beginning to the collection stage, the number of bits in the partial product matrix is compressed by showing loss logic compression. The focus is to compress the number of rows in the partial product matrix, thereby attaining low-difficulty hardware before advancing to collection. Figure 2 shows the difference between the creation stages in exact and the suggested multiplication.

3.2 Clustering a group of rows

The suggested multiplier arranges the partial product terms using different sizes of note-worthy-driven logic clusters. Each logic cluster selects a group of columns containing two bits starting from the least remarkable bits in successive partial products. In general, each 2×L logic collection is answerable for two operations') generating 2L partial product bits within two adjacent rows, *i.e.*, L pairs of perpendicularly arranged bits, by employing 2L AND gates. Then, ii) minimizing these 2L bits by half using L OR gates. Figure 3 decorates the usage of four sizes of logic collectors in 8-bit parallel multiplier. The first 2×7 logic congregator forms 14 partial products by using 14 AND logic gates and removes 7-bit value by using an array of 7 OR logic gates. The second 2 × 6 logic cluster reduces 12 partial products into 6 bits. In a same way the third and fourth logic clusters use 2×5 and 2×4 to reduce 10 and 8 partial products into 5 and 4 bits respectively. By doing so, each logic cluster reduces a gather of perpendicularly arranged bits within two consecutive partial products based on their developing bit importance.

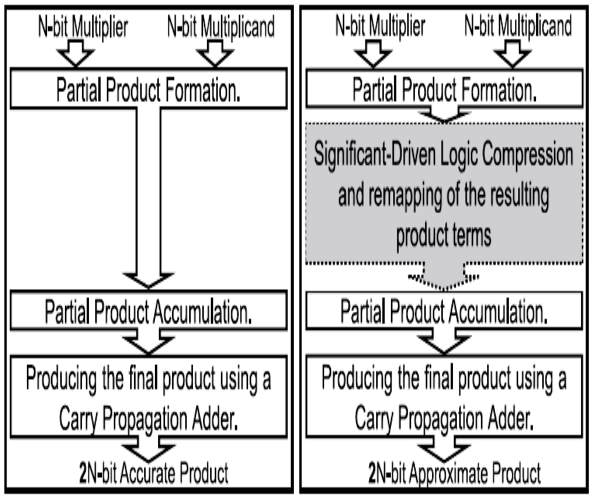

N-bit Multiplier
N-bit Multiplicand
Partial Product Formation.
Partial Product Accumulation.
Producing the final product using a Carry Propagation Adder.
2N-bit Accurate Product
N-bit Multiplier
N-bit Multiplicand
Partial Product Formation.
Significant-Driven Logic Compression and remapping of the resulting product terms
Partial Product Accumulation.
Producing the final product using a Carry Propagation Adder.
2N-bit Approximate Product

Fig.2.Process chart showing the difference between the major stages in: (a) Existing multiplication, and (b) the proposed approach to multiplication.

3.3 Generation of a reduced set of product terms

Using an array of OR gates in each logic group reduces the partial product terms by bisection. A compressed set of preliminary partial product matrix is thus ready to be assembled by appealing any suitable project of multiplication, such as carry-save array, Wallace and Dadda tree. In hypothesis, a two-input OR gate is enough to sum up two bits, *i.e.*, ‘0’+‘1’=‘1’+‘0’=‘1’ ‘0’OR‘1’=‘1’OR‘0’=‘1’ and also ‘0’+‘0’= ‘0’OR‘0’=‘0’. However, the OR gate unsuccessful to give an exact adding if the two inputs are “ones”, *i.e.*, ‘1’+‘1’_=‘1’OR‘1’, the difference value is ‘1’ as the adder backs ‘10’ and OR outputs ‘1’.

3.4 Significance-driven progressive cluster sizing

Since the main aim is to creating a capability-well organized multiplier with minute damages of exactness, the size of the logic collectors is reduced when going down in the

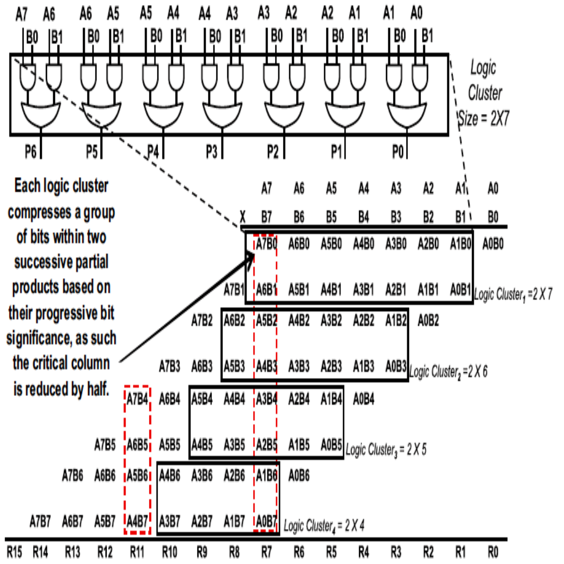

Logic Cluster Size = 2X7
Each logic cluster compresses a group of bits within two successive partial products based on their progressive bit significance, as such the critical column is reduced by half.
Logic Cluster1 =2 X 7
Logic Cluster2 =2 X 6
Logic Cluster3 = 2 X 5
Logic Cluster4 = 2 X 4

Fig.3.Despensation of four different sizes of logic clusters used to compress partial products based on their progressive bit-significance in (8 × 8) parallel multiplier architecture.

Partial product matrix. The more remarkable bits are behaved towards with increasingly higher accuracy, while bits with lower importance are reduced by utilizing the SDLC approach. This permits the most remarkable product terms to be gathered on a carry-generation basis as in the regular multiplier. Thus, the exactness of the remarkable bits of the final outcome is less affected. Inspite of using the same number of AND gates as the accurate multiplier, this can be proceed towards will regularly compress the equipment difficulty of partial product collection.

3.5 Commutative Remapping

The logic compression step comprises the number of partial product terms. This compression can be hold to reduce number of rows prior to the accumulation sage. This can be attained by resizing the partial product terms based on the immediate belongings of the bits, *i.e.*, bits with the same weight are collected in the same column. Due to the compressed number of rows, the crucial path delay is extremely compressed (see Section IV). Figure .4indicates how the size of the partial product bit matrix occurrence of an (8 × 8) multiplier is compressed using the SDLC approach. The bordered packets mention to a group of bits selected by different sizes of logic collectors in which the height of the crucial column is reduced by bisect.

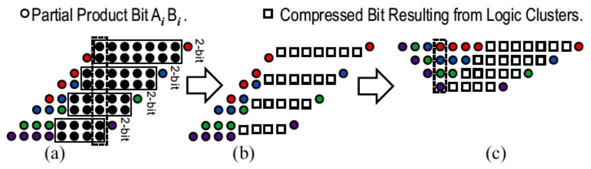
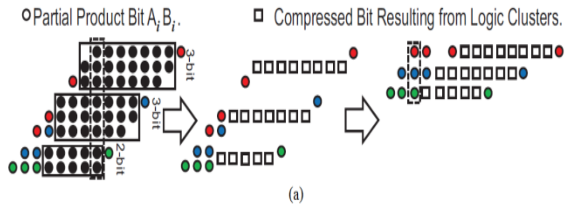

○Partial Product Bit A_jB_i.
□ Compressed Bit Resulting from Logic Clusters.
2-bit
2-bit
2-bit
2-bit
(a)
(b)
(c)

Fig.4. Dot notation shows the major two steps in SDLC approach in the case of (8×8) multiplier: (a) clustering a group of rows (b) generating a reduced set of product terms (c) ordered matrix after applying commutative remapping

4. VARIABLE LOGIC CLUSTER APPROACH

The suggested approach is efficient by affording higher levels of reduction by developing logic cluster depth. Figure 4 demonstrates the impact of increasing depth to 3 and 4 bits in the case of (8 × 8), showing the main points in logic compression and commutative remapping. As can be seen, with increased depth we can attain further compression in the partial product terms, leading to scarce rows for final gathering.


○Partial Product Bit A_jB_i.
□ Compressed Bit Resulting from Logic Clusters.
3-bit
3-bit
2-bit
(a)

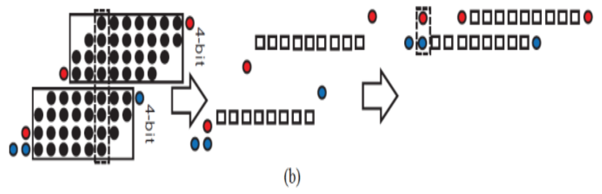


Fig.5. Dot notation showing the impact of increasing the depth of the logic clusters in the case of (8 × 8) multiplier: (a) clustering a group of bits within three successive rows (b) clustering a Group of bits within three successive rows

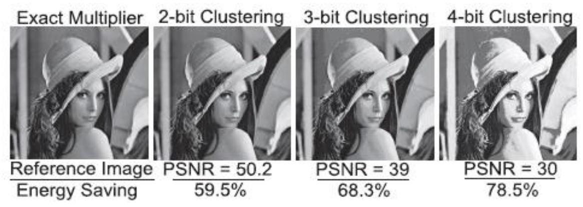
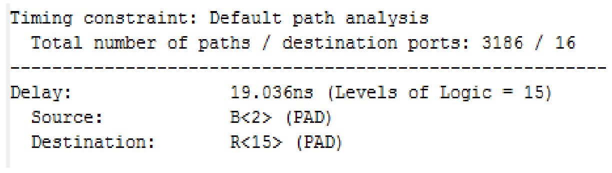


Fig.6. Output quality after applying blurs filtering to three different versions of the proposed (8 × 8) multiplier.

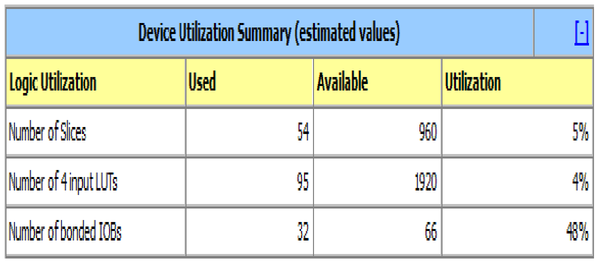
5. EXPERIMENTAL RESULTS

A. Delay report



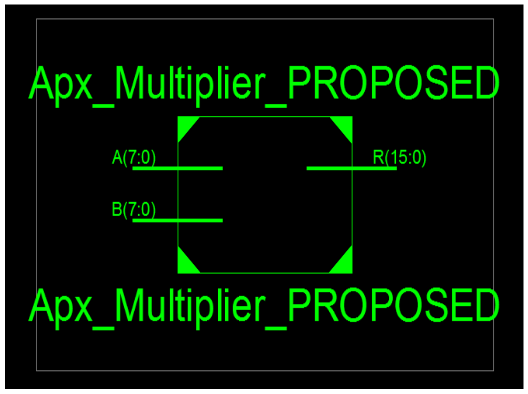
```
Timing constraint: Default path analysis
  Total number of paths / destination ports: 3186 / 16
-------------------------------------------------------------
Delay:                  19.036ns (Levels of Logic = 15)
  Source:               B<2> (PAD)
  Destination:          R<15> (PAD)
```

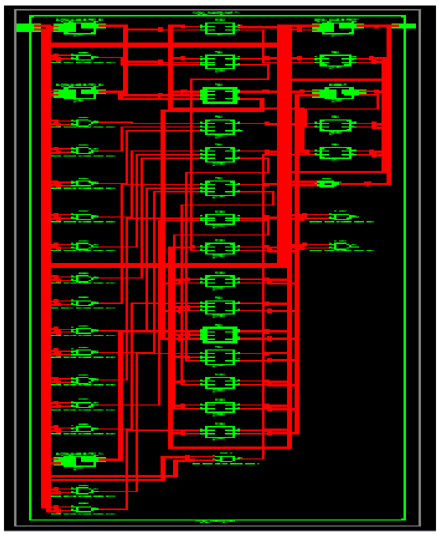
B. Summary report



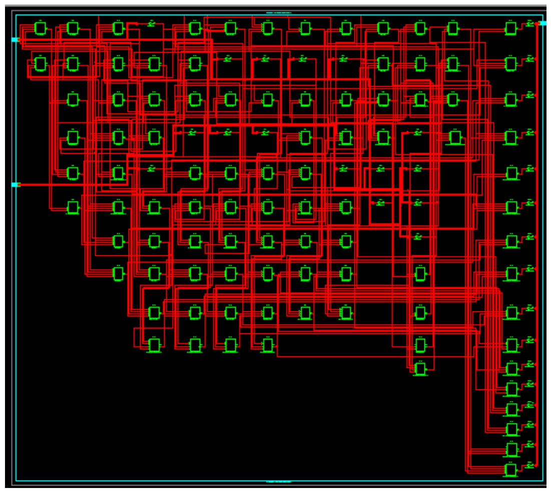
Device Utilization Summary (estimated values)			
Logic Utilization	Used	Available	Utilization
Number of Slices	54	960	5%
Number of 4 input LUTs	95	1920	4%
Number of bonded IOBs	32	66	48%

C. Technology and RTL schematics

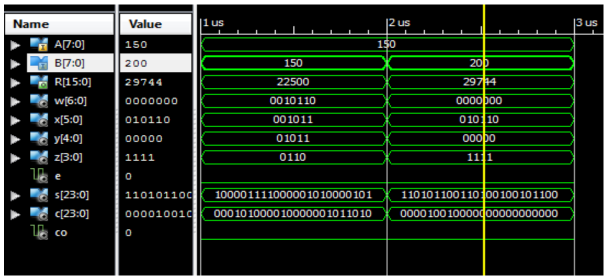




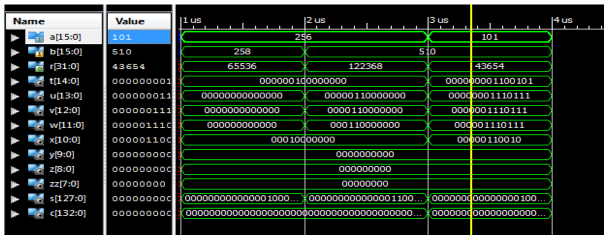
D. Technology Schematic



E. Simulation Results for 8bit multiplier



```
Name        Value        |1 us                         |2 us                         |3 us
A[7:0]      150          150
B[7:0]      200          150                           200
R[15:0]     29744        22500                         29744
w[6:0]      0000000      0010110                       0000000
x[5:0]      010110       001011                        010110
y[4:0]      00000        01011                         00000
z[3:0]      1111         0110                          1111
e           0
s[23:0]     11010110     100001111000001010000101      110101100110100100101100
c[23:0]     00001001     000101000010000001011010      000010010000000000000000
co          0
```



```
Name        Value        |1 us               |2 us               |3 us               |4 us
a[15:0]     101          256                                     101
b[15:0]     510          258                 510
r[31:0]     43654        65536               122368              43654
t[14:0]     000000001    000000100000000                         000000001100101
u[13:0]     000000011    00000000000000      00000110000000      00000001110111
v[12:0]     000000111    0000000000000       0000110000000       0000001110111
w[11:0]     00000111     000000000000        000110000000        000001110111
x[10:0]     00000110     00010000000                             00000110010
y[9:0]      00000000     0000000000
z[8:0]      00000000     000000000
zz[7:0]     00000000     00000000
s[127:0]    00000000     0000000000000001000...  0000000000000001100...  000000000000000100...
c[132:0]    00000000     00000000000000000000000000000000000000...  00000000000000000...
```

6. CONCLUSION

Here in this forecast we can design and developed a new approximate multiplier depend on significance driven logic compression which will generate approximate outcomes with less in error difference such that the behavior in terms of delay and gate count are compressed compared to conventional approximate multipliers.

REFERENCES

[1]. J. Han and M. Orshansky, "Approximate computing: An emerging paradigm for energy-efficient design," in *2013 18th IEEE European TestSymposium (ETS)*, pp. 1–6, May 2013.

[2].C. H. Lin and I. C. Lin, "High accuracy approximate multiplier with error correction," in *2013 IEEE 31st ICCD*, pp. 33–38, Oct 2013.

[3] G. Zervakis, K. Tsoumanis, S. Xydis, D. Soudris, and K. Pekmestzi, "Design-efficient approximate multiplication circuits through partial product perforation," *IEEE Transactions on Very Large Scale Integration(VLSI) Systems*

[4] K. Bhardwaj, P. S. Mane, and J. Henkel, "Power- and area-efficient approximate wallace tree multiplier for error-resilient systems," in *FifteenthISQED*, pp. 263–269, March 2014.

[5] C. Liu, J. Han, and F. Lombardi, "A low-power, high-performance approximate multiplier with configurable partial error recovery," in *2014DATE*, pp. 1–4, March 2014

[6] C. Solomon and T. Breckon, Fundamentals of digital image processing : a practical approach with examples in Matlab. Wiley-Blackwell, 2011.

[7] D. Mohapatra, V. K. Chippa, A. Raghunathan, and K. Roy, "Design of voltage-scalable meta-functions for approximate computing," in *2011*

[8] SuganthiVenkatachalam and Seok-Bum Ko, "Design of Power and Area Efficient Approximate Multipliers" IEEE Transactions On Very Large Scale Integration (Vlsi) Systems

C GANGAIAH YADAV doing research on VLSI. I received M.TECH [VLSI SYSTEM DESIGN] from JNTUA, Anantapuramu in 2012. Currently I'm working as an Assistant Professor in Balaji Institute of Engineering and management Studies at Nellore, Nellore District, AndhraPradesh, India.
Email: ganga.486@gmail.com

K.S. VIJULA GRACE working as an Assistant Professor in Noorul Islam University at kumaracoil, Kanyakumari District, Tamil Nadu, India.

Email:vijulaks@yahoo.co.in