International Journal of Future Generation Communication and Networking
Vol. 13, No. 3, (2020), pp. 2021–2036

Energy Efficient Approximate Multipliers for Error Resilient Applications: Literature Review

Divyasree Mikkili
Department of Electronics and Communication Engineering
Vignan's Foundation for Science, Technology & Research
Vadlamudi-522 213, Guntur, AP, India.
divyasree.mikkili@gmail.com

Sarada Musala
Department of Electronics and Communication Engineering
Vignan's Foundation for Science, Technology & Research
Vadlamudi-522 213, Guntur, AP, India.
sarada.marasu@gmail.com

Abstract

A multiplication of an arithmetic processor has a key impact on the power dissipation and speed. Many algorithms, such as like recognition and classification in data processing, do not always require precise results. In addition, many errors make little noticeable variations in practice, for instance image processing owing to human perceptual restrictions. Error-tolerant algorithms and their utilization inspired the progress of inexact multipliers which trade-off between power efficiency, area and speed. A lot of researchers are enhancing, the overall performance of multiplication process. The mechanisms for achieving the multiplier objectives have been discussed in this study article and the latest developments in the circuit of inexact multiplier of existing models quickly and presents a relative assessment of their error and circuit traits.

Keywords—*Accuracy, inexact computing, Multiplier****.***

I. INTRODUCTION

Today computing system leads to high performance, less area and low power consumption. To achieve these objectives an incipient computing prototype by negotiating arithmetic accuracy known as approximate computing is used. Many systems utilize domains, like big data analysis and multimedia accepts an inherent tolerance to a certain level of inaccuracies. There are numerous applications that give tolerable output regardless of exhibiting approximate /inexact/incorrect computations.

Inexact computing is a computation that provides a possible incorrect result rather than an exact result, for a situation where an approximate result is sufficient for a purpose. These applications are known as error resilient applications. For example, the limited sensitivity of human allows carrying out inexact calculations over the accurate in the applications of an image processing. One example is a search engine where no exact answer may exist for a particular query and hence, several numbers of answers are suitable. In the same way, occasional dipping of some frames in a video application can go unnoticed due to human limitations of recognition.

Inexact computing allows restricted approximation and provides unpredictable gain in energy and performance, but still achieves sufficient accurate results. The inexact computations for obtaining the energy-efficient designs can be at distinct levels of design concept such as algorithm/software, architecture, and circuit to.

ISSN: 2233-7857 IJFGCN

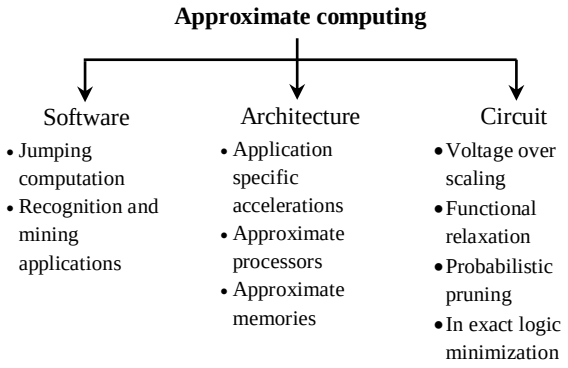

Fig. 1. Taxonomy of approximate computing

Functional approximation deals with the design of approximation of arithmetic units such as adders and multipliers, most of the approximate multipliers designed at the higher abstraction levels i.e. RTL, application and gate. In general multipliers are classified as serial and parallel. Parallel multipliers are faster than serial, and the architecture of parallel multipliers are of two different types, they are 1. Array Multipliers and 2. Tree Multipliers. Array multipliers are faster, but they are complex in design, they give large footprint. Tree multipliers are faster than array, they lead to small footprint. It consists of shift and add logics. Approximation of multipliers is obtained by three techniques. i. Approximation of operands. ii. Approximation of partial product generation. iii. Approximation of partial product tree.

II. MULTIPLIER CIRCUIT ARCHITECTONICS

The architectonics of a multiplier circuit is divided into three stages.

A. Approximation of Operands:

Multipliers based on Logarithmic Number Systems (LNS) have significant improvement over multipliers based on floating point (FLP) numbers and fixed point (FXP) numbers[1]. They proposed a logarithmic Multiplier, transforms operation of multiplication and division into operation of addition and subtraction respectively [2][3]. The logarithmic multiplication classification is shown in Fig. 2. LNS multipliers may be splitting in two groups, (1) lookup tables and interpolations based, and (2) algorithm based on Mitchell's.

i. Mitchel's Algorithim:

Mitchell suggested an algorithm called the Mitchel algorithm (MA), say the logarithm-based multiplication computation contains three steps:

ISSN: 2233-7857 IJFGCN

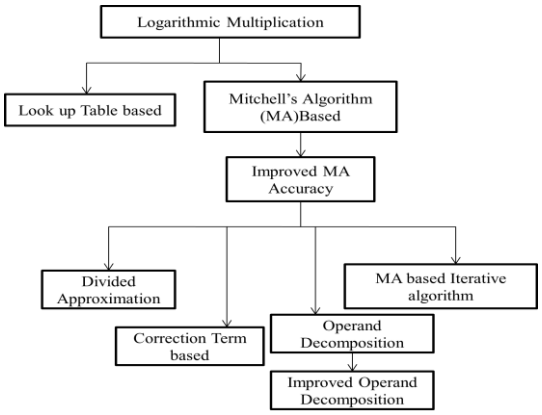

Fig. 2. Classification of Logarithmic multiplication

(1) logarithmic transformation of binary digit for logarithmic representations, (2) mathematic operations are implemented in the logarithmic domain, and (3) antilogarithmic transformation [3]. Block diagram for a computation of LNS multiplication is show figure3.

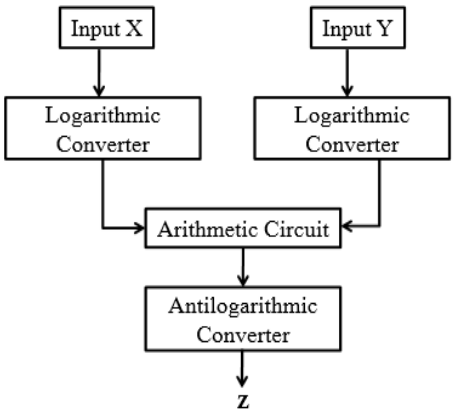

Fig. 3. Block diagram of Mitchel's Algorithm

Table 1: Logarithmic Arithmetic Operations

Binary Operation	**Logarithmic Operation**
P=X1×X2	Log(P)=Log(X1) + Log(X2)
P=X1÷X2	Log(P)=Log(X1) – Log(X2)
P=X1+X2	$Log(P)=Log(X1)+Log_2(1+2^{(Log(X2)-Log(X1))})$
P=X1-X2	$Log(P)=Log(X1)+Log_2(1-2^{(Log(X2)-Log(X1))})$

For example A is multiplicand and B is multiplier let us assume A as $(18)_{10}$ and B as $(60)_{10}$then the product 1080, by MA is obtained as follows

A= 18_{10} = 00010010, B= 6010 = 00111100

log 18 = 100.0010, log 60 = 101.1110

log 18 + log 60 = 10000000000 = 1024 then the error percentage is calculated as

ISSN: 2233-7857 IJFGCN

$$Error\% = \frac{actualvalue - obainedvalue}{actualvalue} * 100 \qquad (1)$$

Error %= (1080-1024/1080)*100 = 5.18

%accuracy = 100-Error = 94.82

The MA gives large error and less accuracy, but it provides less area, low power and with minimum delay, so to improve the accuracy and minimization of error rate, introduce Mitchel iterative algorithm and further Operand decomposition techniqe.

ii. Iterative Mitchel's Algorithm:

The Mitchel's iterative algorithm is also known as correction term. In this the error is corrected with the iterations, [4].

$$P_{approx}^{(0)} = 2^{(k1+k2)} + (A_1 - 2^{k2}) + (A_2 - 2^{k2})2^{k1}$$

$$P_{true} = P_{approx}^{(0)} + (A_1 - 2^{k1})2^{k2} \cdot (A_2 - 2^{k2})$$

After first approximation the absolute error is

$$Er^{(0)} = P_{true} - P_{approx}^{(0)} = (A_1 - 2^{k1}) \cdot (A_2 - 2^{k2})$$

If $Er^{(0)} \geq 0$, the two operands of the multiplier can obtained simply by removing the leading one, A1 and A2 is obtained, again by considering the new values of A1 & A2 until either of the operand is 0, the above algorithm is repeated.

$$Er^{(0)} = Cr^{(1)} + Er^{(1)}$$

Where $Cr^{(1)}$ is the imprecise value of $Er^{(0)}$.$Er^{(1)}$, the absolute error when $Er^{(0)}$ is approximated.

$$P_{true} = P_{approx}^{(0)} + Cr^{(1)} + Cr^{(2)}$$

Now sum up the imprecise value of $Er^{(0)}$ to the inexact Product P_{approx} as the correction term, therefore error of an approximation decreases. Here the correction terms executes similarly by pipelining the fault correction circuits and achieves smaller relative error.

iii. Operand Decomposition:

OD is a process to drop the error obtained by logarithmic multiplication. It decompose the input operands which reduce the switching off conventional multiplication[5]. Mitchell algorithm (MA) Operand Decomposition results by performing logical operations of the input. The decomposition breaks the input operands P and Q, as A=P OR Q, B=P AND Q, C=NOT(P) AND Q, D=P AND NOT(Q); P×Q= (A×B) + (C×D).

ISSN: 2233-7857 IJFGCN

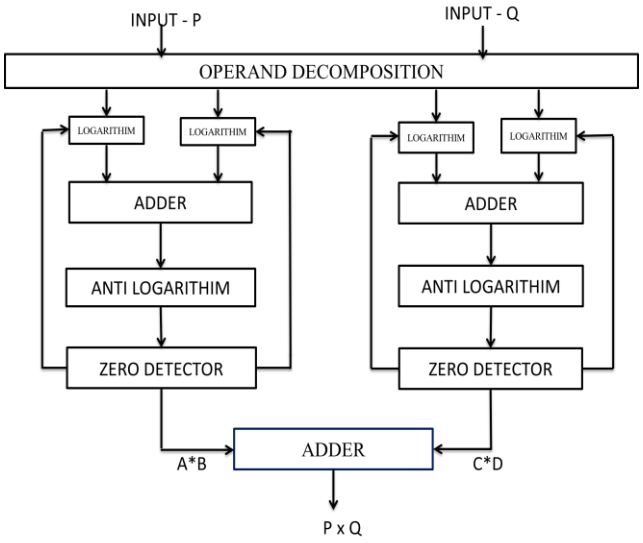

Fig. 4: Block diagram of Operand Decomposition

	00	01	11	10
00	000	000	000	000
01	000	001	011	010
11	000	011	1001	110
10	000	010	110	100

	00	01	11	10
00	000	000	000	000
01	000	001	011	010
11	000	011	111	110
10	000	010	110	100

Multiplication of Fixed point number systems gives complex circuit[6][7]. By introducing some errors LNS will overcome this circuit complexity, due to these errors, it exhibits less accuracy.

B. Approximation of Partial Product Generation:

The under-designed multiplier (UDM) in [8] uses an approximate 2 x 2 bit multiplier block that is acquired by modifying only one entree in the K-Map to its function. In this imprecise model, the accurate result for the multiplication of "11" and "11" is"1001" and is interpreted to "111" to recover one output bit. Assuming each input bit value is evenly likely, the error rate is of $1/(2)^4$= 1/16 for 2x2 multiplier. It is possible to build larger multipliers with 2×2 multiplier[8]. The adder tree is accurate even it introduces a fault while creating a partial products.

Larger multipliers are built with inexact 2x2 block to generate partial products by shifting and then add all the pps. Fig. 8, is an illustration of a 4x4 multiplier compose out from four 2x2 sections, where AL,XL and AH,XH are the lower bits and upper bits of two inputs A, X appropriately.

X1,X0

Y1,Y0

Fig. 5. K-Map for accurate multiplier

X1.X0

Y1,Y0

Fig. 6. K-Map for inaccurate multiplier

ISSN: 2233-7857 IJFGCN

But sometimes it acts as contrary in optimization of an adder tree. Since there are no adder or XOR gates to the inaccurate 2x2 building block[8].Hence immense multiplier blocks can be generated with 2x2 building sections, and enhances in terms of area and power that correlates with exact configuration.

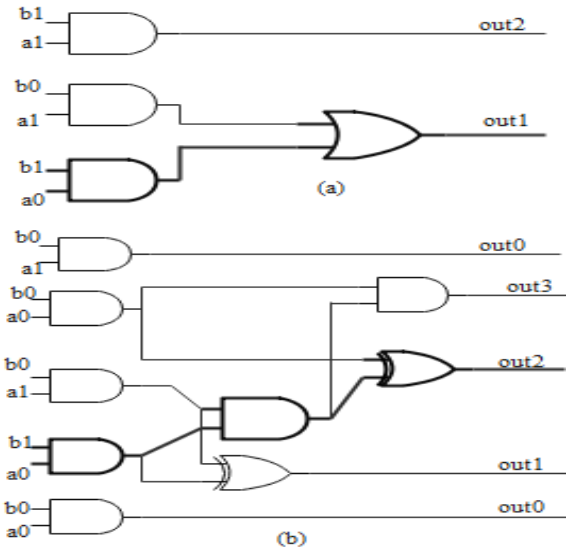

Fig. 7. The (a) inaccurate and (b) accurate 2x2multipliers, with the critical paths

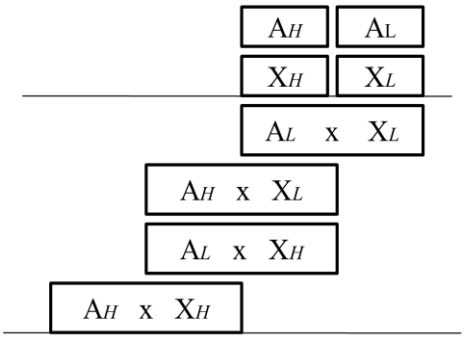

Fig. 8. Construction of large multipliers with smaller blocks

In an example the entire four bits taken as 2x2 multiplier, even if error occurs while creating the partial product, the product remains accurate.

C. Approximation in partial product tree:

There are so many techniques to reduce the partial products i)Broken Array multiplier(BAM). ii) Truncated multiplier (TRUM) iii) Error tolerant Multiplier(ETM). iv) Approximate wallace tree multiplier (AWTM), vi) Approximate counters or compressors(AC/ACM).

i. Broken Array Multiplier:

The basic construction of a 6×6 broken-array multiplier (BAM) in [9] is identical to the array multiplier structures. It comprises of identical cells called carry-save adder (CSA) array represented by squares, followed by a vector of bits integrated with adder (that are represented by rounded squares) to converts into normal binary formats from carry save redundant form. In Fig. 9. (a). The cell contains a 2-input AND gate to create a pp's, and a FA-(CSA) performs the addition of pp's to the successive sum. The CSA cells in the first row in Fig. 9. (a). can be replaced by one AND or NAND gate only. A BAM separates the CSA

ISSN: 2233-7857 IJFGCN

array and ignore certain cells which lead to a faster and smaller circuit, while providing inexact results. As illustrated in Fig. 9(a), quantity and location of ignored cells depend on two parameters introduced as: Horizontal Break Level (HBL) and Vertical Break Level (VBL). This implies that the CSA cells are not missing horizontally. Growing the HBL, slides down the horizontal oblique line and eliminate all cells that lies above the continuous line.

Likewise, no CSA cells are removed and as the VBL rises, the steep oblique edge shifts left, and all the cells on right-side of this VBL are ignored. A similar technique for creating new imprecise multipliers may be extended to other multiplier structures.

Equations (7) and (8) measure the maximum and minimum differences between the output of a BAM and an effective multiplier for the constant WL. It means that a BAM also sets restricted results in relation to an exact multiplier of the same WL. Equation (9) indicates the mean error in terms of VBL, HBL and WL. The BAM MRE (in percentage) for and variable values of HBL / VBL parameters[9]. Fig..9. Shows the BAM provides a mean error of zero. When the VBL and HBL rise, the BAM 's relative ME improves with varying proportions.

$$MAX_{BAM} = (2^{WL} - 1) \times (\sum_{i=0}^{HBL-1} 2^i) + 2^{HBL} \times (\sum_{i=0}^{VBL-HBL-1} (2^{VBL-HBL} - 2^i)) \quad (7)$$

$$MAX_{BAM} = 0 \quad (8)$$

$$ME_{BAM} = (\frac{2^{WL} - 1}{4}) \times (\sum_{i=0}^{HBL-1} 2^i) + 2^{VBL-2} \times (\sum_{i=0}^{VBL-HBL-1} (1 - 2^{-(i+1)})) \quad (9)$$

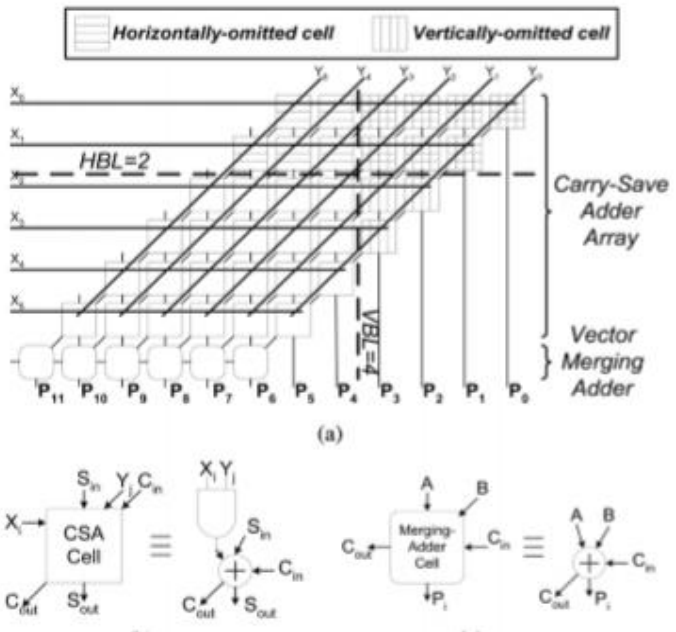

Fig. 9. (a)Structure of the BAM, (b) CSA cell, and (c)adder cell merging.

ISSN: 2233-7857 IJFGCN

ii. Truncated Multiplier:

Truncated multiplication is a good choice for sinking the delay, foot print on silicon and dissipation of power. In the multiplier matrix the less important partial product bits is omitted to achieve truncated multiplication, and additional gates are inserted to offset the errors arising from eliminating a partial multiplier matrix. Tree multipliers are more rapid than array multipliers, but local interconnects and structure has everlasting beneficial effects [10]. A truncated multiplier is a b × b multiplier with b output bits. In Fig.11. Partial products are portioned into two subsets. The less significant columns of the PPM, is known as least important part (LSP) the left over columns is known as most significant part (MSP). The over all output of the multiplier, P is given by

$$P = SMSP + SLSP \tag{10}$$

Where SMSP and SLSP represent the weighted sum of MSP and LSP elements respectively. For a required b output bits, the accurate option is to use the full rounded multiplier: it calculates all PPM, adds a constant on 2b bits to the result and only takes the first b bits of the result, and the next option is to discard the LSP's because they exhibit negligible contribution towards b output bits than MSP's. In terms of hardware efficiency this approach is very beneficial. The array of truncated multipliers is implemented using b = 8. "There are no cells in the LSP matrix, only half the number of cells in the final circuit as applied to the whole width (Fig.10). However, a straight forward analysis shows that the direct removal of the partial products from the LSP origins a very large error bounded by (n/2−1) lsb, where lsb is the weight of the least significant bit of the result". Numerous strategies that support this concept have been proposed[10],[11][12].

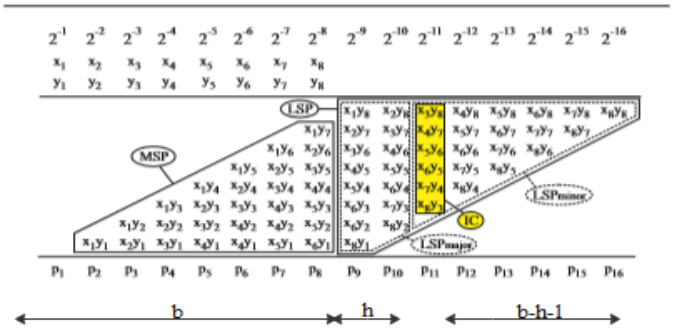

Fig. 10. PPM of the multiplier b=8. (b+ h + 1)th column is the IC vector(input correction) MSP and LSP of the PPMatrix is established by the b columnson both the sides.

Constant Correction: An imprecise rounding process was developed [13]. It consists of an applying a constant at LSP's whose value is the average of the unformed bits. By adding a 1 as for true rounding, the final estimated product is rounded. For the 6x6 multiplier of Fig. 11. the correction term 1 is appended at the least significant sixth column, and again 1 is added for rounding [14].

Variable Correction: It can be expanded by shaping and inserting bit at the Products are not usually formed on right margin of an array[14].

ISSN: 2233-7857 IJFGCN

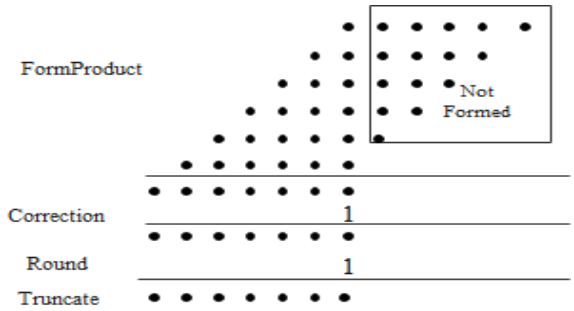

Fig. 11. Constant Correction Multiplication

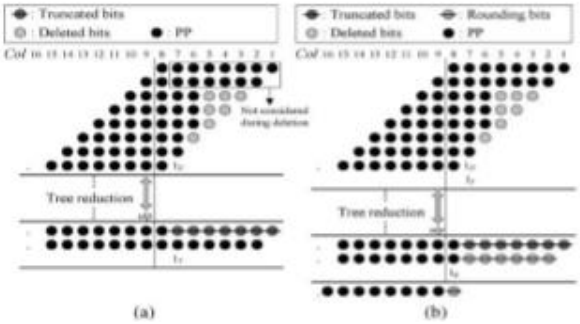

Fig. 12. Truncated multiplier with constant error correction and rounding

The sophistication of the method of variable correction is significantly greater than that of the fixed form of correction.

A. iii. Error Tolerant Multiplier:

A new algorithm for multiplication can be shown by an illustration in Fig. 13. First, the operands of the input are divided into a multiplication component comprising several higher significant bits and a non-multiplication part containing the lower significant bits. No need to be equal in length of each element [15]. The process of multiplication occurs at the point where the bits are separated and move concurrently in the two opposite directions.

In the case of Fig. 13. the two 12-bit input operands, the "101110011011" (2971) multiplicand and the "010011 00 1001" (1225) multiplier, are splitted into two equal sections each consists of 6 bits. The lower order bits (non multiplication part), here a special procedure is intended for not producing partial product and removes the carry propagation path. Every bit location of the non-multiplication component from left to right is checked and if one or both of the two operand bits are "1," the testing method is terminated and from that bit location all the bits are placed as "1."

As normal, the operation is performed from (LSB to MSB) right to left as the higher significant bits of an operands at the input which begins the multiplication portion. Hence, the circuit is designed in a conventional manner. Here, we maintained the conventional topology because the MSB's are weighted more than the LSB's [15]. These two operations perform simultaneously, multiplying the MSBs, the power consumption and overall delay is reduced. Fig.14. represents the architecture of 12 bits ETM.

ISSN: 2233-7857 IJFGCN

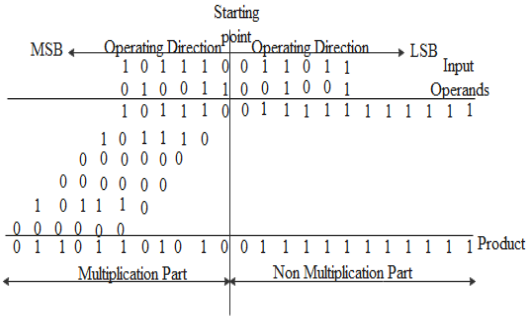

Fig. 13. Error Tolerant Multiplier

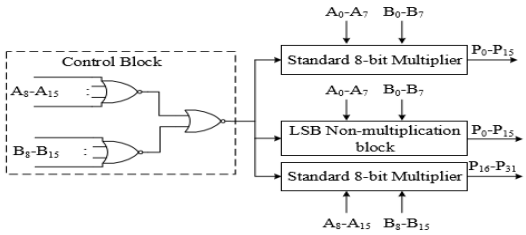

Fig. 14. Block diagram of l2-bits ETM.

iv. Approximate Wallace Tree Multiplier:

Approximate Wallace tree multiplier (AWTM) is depend on an inexact multiplication of bit-width and a "carry-in prediction process" In [16] proposed Four b/2-bit sub-multipliers enforce an n-bit AWTM, as shown in Figure 15, where four b/4-bit sub-multipliers further implemented the most important sub-multiplier AHBH. The AWTM is sub divided into four specific modes by the number of estimated n/4-bit sub-multipliers, while the other three multipliers (AHBL, ALBH and ALBL) are imprecise. A Wallace tree accumulates then the inexact partial products.

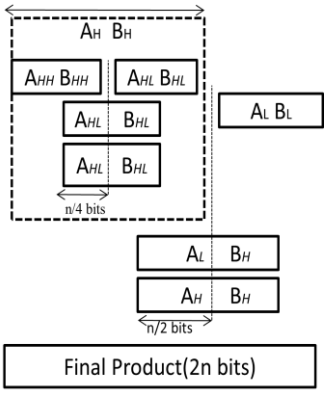

Fig. 15. Pipelined Inexact Multiplier

In order to ensure a high precision, the MSBs of the total 4b bit product (A × X) supposed to be precise to high level. As a result, they produced a multiplier AHXH as b×b precise multiplier and AHXL, ALXH, ALXL as b×b approximate multipliers. The b×b imprecise multipliers produce upper b bits as precise to higher level, that further causes the upper 2b bits among 4b bit product accomplish high precision [16]. Fig.16. Illustrates the carry-in prediction technique of these imprecise b×b multipliers[14].

The Carry-in Prediction – consider An example of two unsigned 16- bit multiplication (i.e. b =8):

"A=$(aedb)_{16}$ = $(44763)_{10}$

ISSN: 2233-7857 IJFGCN

$X=(e6e7)_{16} = (46823)_{10}$."

Approximate product out of ALXL, ALXH and AHXL. We need ALXL i.e. $(db)_{16}$, $(e7)_{16}$ to be estimated. As illustrated in Fig.16(a), We separate the entire multiplication into three sections: initial, b/2 (LSBs) are of exact value, whereas the second b/2 bits are placed to 1's, and thirdly is of correct calculation of the residual components in the PPM(partial product matrix) with a carry 'C' in addition resulting from the at least relevant position of the inexact part. So, the impression is to pre-calculate 'C' with some technique and start multiplication from the third and first parts concurrently. Similarly, by locating the bits in the second part, the no of additional operations involved are reduced, thereby reduces the expenditure of the hardware. Fig.16. (a) Additionally shows a column incorporates extreme amount of elements in multiplication. This theory employs a carry of at least 1 is indeed to reproduce to the subsequent line when there are two or more 1's in the vital column. The b/2 bits is set as 1's in the inexact part because carrying propagated from the vital column is probable to be more than 1, this minimize the error. Determining AHXL and ALXH in an identical approach, and summing all as shown in Fig. 16(b). as $(7CEBA7FB)_{16}$ imparts imprecise results. The exact value is $(7CED799D)_{16}$. In this case the relative error is pure 0.00.

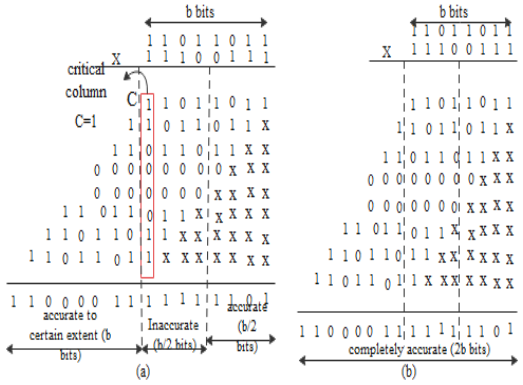

Fig. 16. An example of Carry-in Prediction (i.e. 16×16 multiplication) for b=8 bits: (a) inexact ALXL (b) Accurate ALXL.

v. Approximate Counters or Compressors:

Compressors are the most dominant building blocks for a rapid multiplier. It provides an improvement of partial product addition at the outlay of smallest power dissipation. Instead of fully summoning partial products with the support of the CSA / Ripple adder tree, a compressor structure can accomplish the same function in much less period and at the same moment remove the problems of upturn of area high power consumption [17].

This addition of partial products can not compensate as much for the reduction of delay allied with the critical path in conventional method comprises of half adders and full adders than counters or compressors.

The purpose for compressor's apparent preference over counters is the advantages it provides in terms of total number of transistors used, power and delay allied with the critical path (mainly includes XORs). Both MUXs and XORs can be incorporated in the compressor configuration.

The Compressors which reduce stages of the n product. A 4:2 compressor generates in a similar way , and a carry for the next stage. moreover, the carry in (Cin) of the new compressor produce (Cout). A 4:2 precise compressor consists two complete adder circuits in Fig.19,20. That describes a compressor configuration that uses two 2:1 multiplexers and three XOR-XNOR gates. The compressor logic equation outputs are as follows:

$$Sum = P \oplus Q \oplus R \oplus S \oplus C_{in} \quad \square\square\square\square$$

$$C_{out} = (P \oplus Q)C_{in} + (P \oplus Q)' P \quad \square\square\square\square$$

ISSN: 2233-7857 IJFGCN

$$Carry = (P \oplus Q \oplus R \oplus S) C_{in} + (P \oplus Q \oplus R \oplus S)' S$$

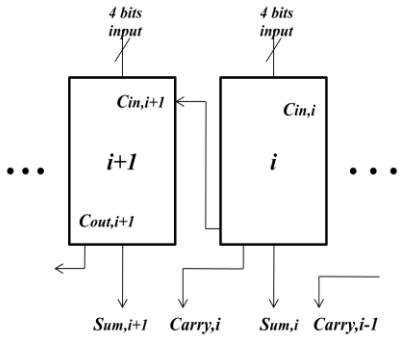

Fig. 17. Adjacent compressors with in a chain in the partial product reduction stage.

In this multipliers the two inaccurate compressor designs were used. The two designs are depends on modifications to the exact compressor's truth table to minimize hardware[18],[19],[20]. In design1, the carry signal is connected straightly to the Cin signal, and in order to minimize the hardware and thus reduces the delay, the sum and C_{out} signal columns are altered.

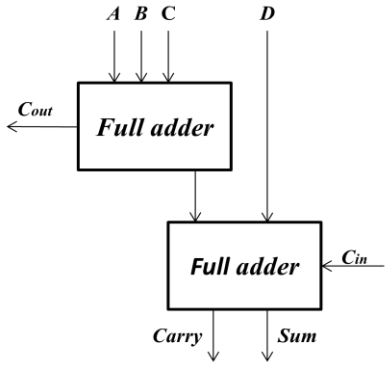

Fig. 18. An exact compressor by using two FA's.

For design1 the logic functions are as follows:

$$Sum = \left((P \oplus Q)' + (S \oplus R)'\right) C'_{in}$$

$$C_{out} = \left((P + Q)' + (S + Q)'\right)'$$

$$Carry = C_{in}$$

In Design2, C_{out} is removed completely from the circuit and gives better accurate output. The practical logic is as follows:

$$Sum = (P \oplus Q)' + (S \oplus R)' \quad (17)$$

$$Carry = ((P + Q)' + (S + R)')' \quad (18)$$

ISSN: 2233-7857 IJFGCN

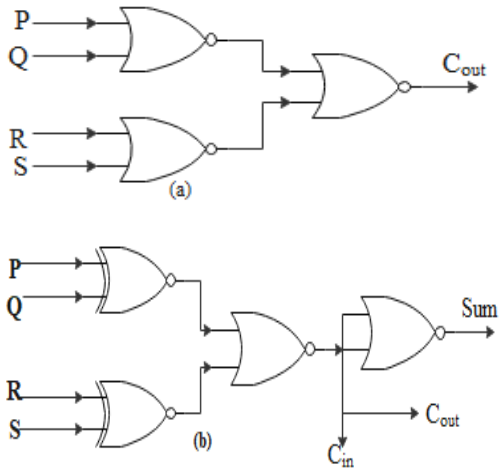

Fig. 19. Design-1approximate compressor circuits

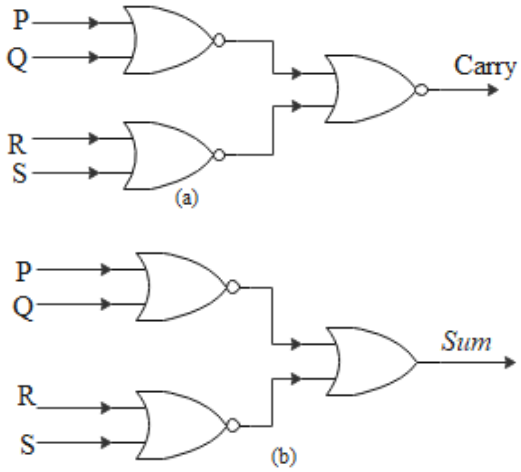

Fig. 20. Design-2 approximate compressor circuits a) carry b) sum

III. CONCLUSION

Inexact unsigned multipliers for both error and circuit parameters are assessed as comparative. Among the inexact multipliers considered, truncation is an efficient way to reduce the complexity of the circuit. This incurs, however, a strong ER, and NMED and MRED. UDM shows a low accuracy, Imprecise multipliers in the partial product tree appear to have low precision and reasonable hardware utilization when truncation is not used, whereas multipliers that use inexact counters or compressors are typically quite precise with reasonable consumption of hardware and high power dissipation.

ISSN: 2233-7857 IJFGCN

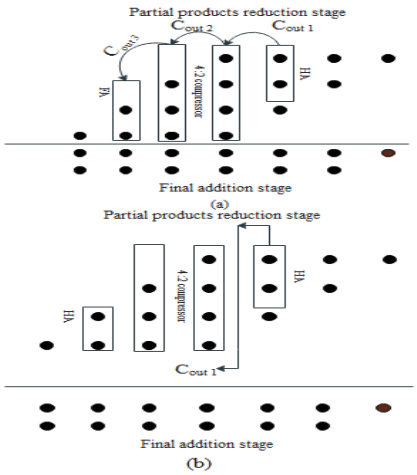

Fig. 21. Use of compressors for partial production reduction

Table 2. Comparison between methods and parameters of multipliers

Author	Method	Optimized Parameters
Approximation of Operands		
Mitchell [1]	Logarithmic Number System	Improves speed, but large error
Maclaren [2]	Iterative MA	71% Error is minimized and area51%
V. Mahalingam [5]	Operand Decomposition	Reduces switching activity and error is minimized
Durgesh [6]	Improved Operand decomposition	Reduces area & Improves speed
Approximation of partial product generation		
Kulkarni [8]	UDM	Power [30%-50%] mean error by [1.3%]
Approximation of Partial product tree		
Mahdiani [9]	BAM	Saves power, area, improves speed
Kyaw S.Kim [15]	ETM	Power by 93%
Bhardwaj [16]	AWTM (carry in prediction)	Saves power, area, improves accuracy

ISSN: 2233-7857 IJFGCN

Zervakis [17]	Pp perforation	Power[50%] Area[45%] Critical path delay[35%], RE
V.leon [20]	Approximation of high radix encoding	Energy & area saved by [55%]
Cong Liu [18]	Approximate R4BEM	Energy[59%], error
Weiqiang Liu[19]	(ARBC-1,ABRC-2)	Energy[45% - 64%],

REFERENCES

[1] J.N. Mitchell, "Computer Multiplication and Division using Binary Logarithms," IRE Trans. Electronic Computers, Vol. 11, No. 6, pp. 512-517, Aug. 1962.
[2] M.J. Duncan, "Improved Mitchell Based Logarithmic Multiplier for Low Power DSP Applications," IEEE International Conference on System on a Chip Conf. (SOCC), Portland, USA. pp. 17-20, Sep. 2003.
[3] Y. Sun, and M.S. Kim, "A High-Performance 8Tap FIR Filter Using Logarithmic Number System, IEEE International Conference on Communications (ICC), Kyoto, Japan. June. 2011.
[4] Z. Babic, A. Avramovic, and P. Bulic, "An iterative logarithmic multiplier," Microprocessors and Microsystems, vol. 35, No. 1, pp. 23-33, Dec. 2011.
[5] V. Mahalingam and N. Rangantathan, "Improving Accuracy in Mitchell's Logarithmic Multiplication Using Operand Decomposition," IEEE Trans. on Computers, Vol. 55, No. 2, pp. 1523-1535, Dec. 2006.
[6] Durgesh Nandan, J.Kanungo and A. Mahajan, "An efficient VLSI architecture design for logarithmic multiplication by using the improved operand decomposition," International Journal of Engineering & Technology, vol. 58, pp. 134–141, April. 2017.
[7] Liu, W., Xu, J., Wang, D. and Lombardi, F., "Design of approximate logarithmic multipliers," Proc. Great Lakes Symposium on VLSI, Banff Alberta, Canada. pp. 47-52, May. 2017.
[8] P. Kulkarni, P. Gupta, and M. Ercegovac, "Trading accuracy for power with an underdesigned multiplier architecture," in 24th International Conference on VLSI Design, Chennai, India. pp. 346–351, Jan. 2011.
[9] 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. on Circuits and systems, Vol. 57, No. 4, pp. 850- 862, April. 2010.
[10] King Earl E. Swartzlander, "Truncated Multiplication with Approximate Rounding," IEEE, Thirty-Third Asilomar Conference on Signals, Systems, and Computers Pacific Grove, CA, USA. PP.1480-1483, Feb. 1999.
[11] M. J. Schulte and E. E. Swartzlander, Jr. "Truncated squares with Constant and variable Correction," SPIE - The International Society for Optical Engineering, Vol.4, pp. 1-11, oct. 2004.
[12] Chip-Hong Chang, Ravi Kumar Satzoda and Swaminathan Sekar,"A Novel Multiplexer Based Truncated Array Multiplier," IEEE International Symposium on Circuits and Systems, Kobe, Japan. pp.85-88, May. 2005.
[13] Y. Lim, "Single-precision multiplier with reduced circuit complexity for signal processing applications," IEEE Trans. on Computers, vol. 41, No. 10, pp. 1333–1336, Oct. 1992.
[14] R. Muthammal, S. Sandhya, "Design and Implementation of Truncated Multiplier in Fir Filter," IJAICE, Vol. 1 ,No. 3, pp. 53-56, Mar. 2015.
[15] K. Y. Kyaw, W. L. Goh, and K. S. Yeo, "Low-power high-speed multiplier for error-tolerant application," IEEE International Conference of Electron Devices and Solid-State Circuits (EDSSC), Hong Kong,\China, pp.1–4. Dec. 2010.
[16] K. Bhardwaj, Pravin S. Mane, J¨org Henkel, "Power- and Area-Efficient Approximate WallaceTree Multiplier for Error-Resilient Systems," Fifteenth International Symposium on Quality Electronic Design, IEEE conference, Santa Clara, CA, USA, pp. 263-269, Mar. 2014.
[17] G. Zervakis, K. Tsoumanis, S. Xydis, N. Axelos, and K. Pekmestzi, "Approximate multiplier architectures through partial product perforation: Power-area tradeoffs analysis," 25th Great Lakes Symp.VLSI, Pittsburgh, Pennsylvania, USA. pp. 229–232, May. 2015.

ISSN: 2233-7857 IJFGCN

[18] Honglan Jiang, Cong Liu, Leibo Liu, Fabrizio Lombardi, Jie Han, "A Review, Classification and Comparative Evaluation of Approximate Arithmetic Circuits," J. Emerg. Technol. Comput. Syst. Vol. 13, No. 4, pp. 60.1-60.36, July. 2017.
[19] Weiqiang Liu, Senior, Tian Cao, Peipei Yin, Yuying Zhu, Chenghua Wang, Earl E. Swartzlander Fabrizio Lombardi, "Design and Analysis of Approximate Redundant Binary Multipliers," ," IEEE Trans. on Computers, vol. 68, No. 4, pp. 804–819, June. 2019.
[20] Vasileios Leon, Georgios Zervakis, Dimitrios Soudris, and Kiamal Pekmestzi ," Approximate Hybrid High Radix Encoding for Energy-Efficient Inexact Multipliers," IEEE Trans On Very Large Scale Integration (Vlsi) Systems, vol. 26, No. 3, pp. 421–430, March. 2018.

ISSN: 2233-7857 IJFGCN