IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

MODIFIED COMPRESSOR BASED HYBRID MULTIPLIER FOR LOW POWER APPLICATIONS

Abinaya S[1], Janani A.M[2], Praveena S[3], Padmapriya K[4]

[123]UG student, Dept. of Electronics and Communication Engineering, AAA College of Engineering and Technology, Virudhunagar, Tamil Nadu, India.

[4]Assistant Professor, Dept. of Electronics and Communication Engineering, AAA College of Engineering and Technology, Virudhunagar, Tamil Nadu, India.

Abstract: Multi-digit multiplication is widely used for various applications in recent years, including numerical calculation, chaos arithmetic, primality testing. Systems with high performance and low energy consumption are demanded, especially for image processing and communications with cryptography using chaos. In this project, hardware design of multi-digit multiplication is described and its VLSI realization is evaluated in terms of the cost and performance. A new type of compressor-based multiplier is proposed for low power application. The results obtained by this study will help system designers for applications requiring multi-digit multiplication to select design alternatives including FPGA realization.

Key Words: Compressor ; Multiplier ; Low power applications ; Approximate Compressor

I. INTRODUCTION

A multiplier is one of the key hardware blocks in most digital signal processing (DSP) systems. Typical DSP applications where a multiplier plays an important role include digital filtering, digital communications and spectral analysis. Many current DSP applications are targeted at portable, battery-operated systems, so that power dissipation becomes one of the primary design constraints. Since multipliers are rather complex circuits and must typically operate at a high system clock rate, reducing the delay of a multiplier is an essential part of satisfying the overall design.

Multiplications are very expensive and slows the overall operation. The performance of many computational problems are often dominated by the speed at which a multiplication operation can be executed. Consider two unsigned binary numbers X and Y that are M and N bits wide, respectively. To introduce the multiplication operation, it is useful to express X and Y in the binary representation.

$X = \Sigma\ Xi\ 2i\ i = 0 \text{ to } M$ (1)

$Y = \Sigma\ Yi\ 2j\ j = 0 \text{ to } N$ (2)

With Xi , Yi Є {0,1}. The multiplication operation is then defined as follows:

$Z = X \times Y = \Sigma\ Zk\ 2k\ k = 0 \text{ to } M + N - 1$ (3)

$= (\Sigma\ Xi\ 2i\ i = 0 \text{ to } M)\ (\Sigma\ Yi\ 2j\ j = 0 \text{ to } N)$ (4)

$= \Sigma\ (\Sigma\ Xi\ Yi\ 2i+j)\ i = 0 \text{ to } M-1,\ j = 0 \text{ to } N-1$ (5)

.The simplest way to perform a multiplication is to use a single two input adder. For inputs that are M and N bits wide, the multiplication tasks M cycles, using an N-bit adder. This shift –and-add algorithm for multiplication adds together M partial products. Each partial product is generated by multiplying the multiplicand with a bit of the multiplier – which, essentially, is an AND operation – and by shifting the result in the basis of the multiplier bit's position. Similar to the familiar long hand decimal multiplication, binary multiplication involves the addition of shifted versions of the multiplicand based on the value and position of each of the multiplier bits. As a matter of fact, it's much simpler to perform binary multiplication than decimal multiplication. The value of each digit of a binary number can only be 0 or 1, thus, depending on the value of the multiplier bit, the partial products can A multiplier is one of the key hardware blocks in most digital signal processing (DSP) systems. Typical DSP applications where only be a copy of the multiplicand, or 0. In digital logic, this is simply an AND function.

.A faster way to implement multiplication is to resort to an approach similar to manually computing a multiplication. The entire partial product are generated at the same time and organized in an array. A multioperand addition is applied to compute the final product. The approach is illustrated in the figure. This set of operation can be mapped directly into hardware. The resulting structure is called an array multiplier and combines the following three functions: partial product generation, partial-product accumulation and final addition.

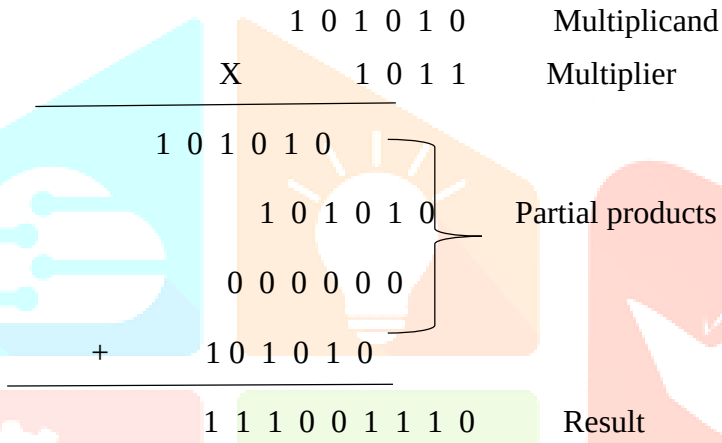


Fig 1 : Manual Multiplication

III. EXISTING AND PROPOSED SYSTEM

A. Existing System

we target the design of power–error efficient multiplication circuits. We differ from the previous works by exploring approximation on the generation of the partial products. The proposed method can be easily applied in any multiplier architecture without the need for a special design, in contrast to related works. In addition, the error imposed by perforation depends only on the configuration parameters and, in contrast to existing work, can be analytically calculated without the need for exhaustive simulations.

$$A \times B = \sum_{i=0}^{n-1} Ab_i 2^i, \; b_i \in \{0, 1\}.$$

$$A \times B|_{j,k} = \sum_{\substack{i=0, \\ i \notin [j, j+k)}}^{n-1} Ab_i 2^i, \; b_i \in \{0, 1\}.$$

The partial product perforation method for the design of approximate hardware multipliers is described. Consider two *n*-bit numbers *A* and *B*. The result of their multiplication $A \times B$ is obtained after summing all the partial products *Abi*, where *bi* is the *i* th bit of *B*. Thus The partial product perforation technique omits

the generation of *k* successive partial products starting from the *j* th one. A perforated partial product is not inserted in the accumulation tree, and hence *n* full adders can be eliminated. Applying the product perforation with *j* and *k* configuration values on the multiplication, $A \times B$ produces the approximate result For each architecture, the dot diagrams of the accurate and the respective perforated tree are presented. The dots represent the bits of the partial products that have to be accumulated, while the stages represent the delay of the reduction process followed by each tree. The dashed boxes with four dots are 4:2 compressors, those with three are full adders and those with two are either full- or half-adders. Through the proposed approximation technique, the power, area, and delay of the multiplication circuit are decreased, making, though, the computation imprecise. The higher the order of a perforated partial product, the greater the error imposed at the final result. In addition, since the addition is an associative and commutative operation, when more than one partial products are perforated, the total error results from the addition of the errors produced from the perforation of each partial product separately For each architecture, the dot diagrams of the accurate and the respective perforated tree are presented. The dots represent the bits of the partial products that have to be accumulated, while the stages represent the delay of the reduction process followed by each tree. The dashed boxes with four dots are 4:2 compressors, those with three are full adders and those with two are either full- or half-adders. Through the proposed approximation technique, the power, area, and delay of the multiplication circuit are decreased, making, though, the computation imprecise. The higher the order of a perforated partial product, the greater the error imposed at the final result. In addition, since the addition is an associative and commutative operation, when more than one partial products are perforated, the total error results from the addition of the errors produced from the perforation of each partial product separately that produces specific least significant bits (LSBs) of the accumulation tree, while the perforation skips the generation of partial products and thus decreases the number of operands to be accumulated. For example, in an 8-bit array multiplier, perforating a partial product removes eight full adders from the accumulation tree and reduces its delay. In order to attain similar circuit reduction using truncation, 6 LSB have to be truncated. However, truncating 6 LSB does not offer any delay reduction. Moreover, in this example, the truncation delivers, in all the cases, incorrect results, whereas the outputs of perforation are 50% correct. Finally, perforating one partial product (out of eight) results in a 12.5% loss of information while truncating 6 LSB (out of 16) results in a 37.5% information loss

B. Proposed System

The proposed design combines a Modified Compressor-Based Approximate Multiplier (MCBAM) and a Dynamic Approximate Compressor-Based Multiplier (DACBM) to achieve a balance between accuracy and power efficiency**.**

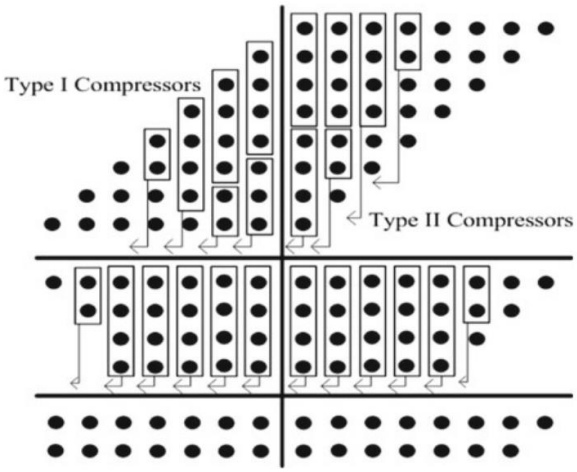


Fig 4: Proposed Multiplier

The proposed system introduces 'don't care' conditions to reduce the complexity of the compressor design in a multiplier. This approach can be particularly useful in approximate computing, where the goal is to save power and reduce hardware complexity by allowing some degree of inaccuracy in the computation. Here's a more detailed explanation of the modified compressor design and its relevance to approximate computing.

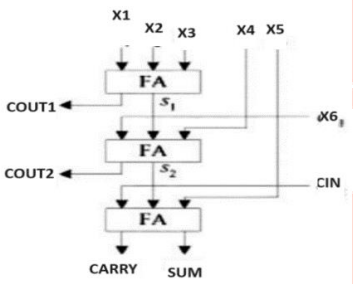


Fig 5: Proposed 6:2 compressor

The modified compressor design introduces 'don't care' conditions in the equations for the sum and carry outputs. These 'don't care' conditions indicate that the output value can be either 0 or 1, depending on the optimization goals and the specific computation requirements.

Sum and Carry Equations:

1. Sum Output:

- Original Equation: Sum = X1X2X4 + X1(X2 + X4) + X2X4 + X2X6

- Modified Equation with 'Don't Care': Sum = X1 + X2X4 + X2 + X2X4

2. Carry Output:

- Original Equation: Carry = X2 + X2X4 + X1X4 + X1X2X6

- Modified Equation with 'Don't Care': Carry = X1 + X2 + X3X4

Approximate Computing and 'Don't Care' Conditions:

In approximate computing, the goal is to achieve a balance between computational accuracy and resource efficiency. By introducing 'don't care' conditions, you are allowing certain bits in the computation to be

approximate or uncertain, which can lead to significant savings in terms of hardware resources and power consumption.

IV. RESULT

A.Modelsim Simulation:

Below are the results of our project that contains modelsim results and Xilinx synthesis results. Ensures bit-accurate results for test inputs, validating the correctness of the partial product generation and accumulation stages. There would be two inputs x and y . The output will also be generated. The multiplier module takes two binary inputs and produce output using the proposed architecture.

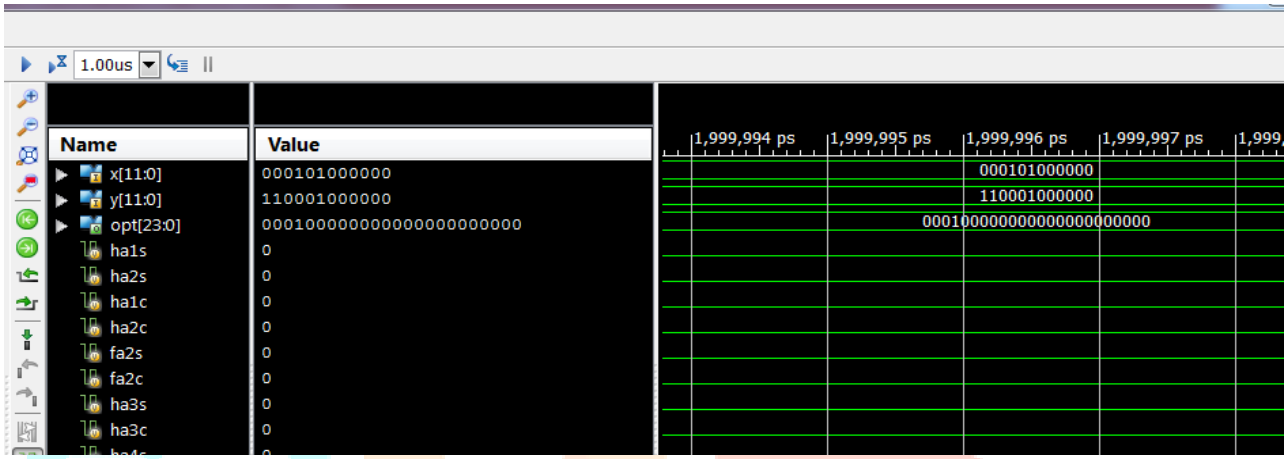


Fig 6: Product Output

Confirms that the proposed approximation strategies still produce results within acceptable error margins.The design meets low power , reduced area and acceptable accuracy. The multiplier takes two binary inputs and produces a binary output. Internally, the multiplication uses modified compressor with approximation logics in LSB.

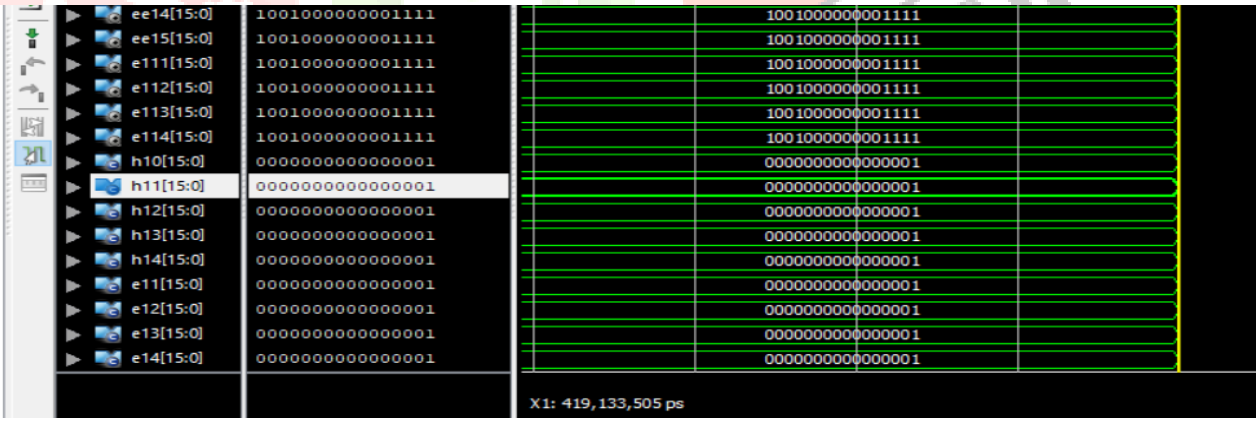
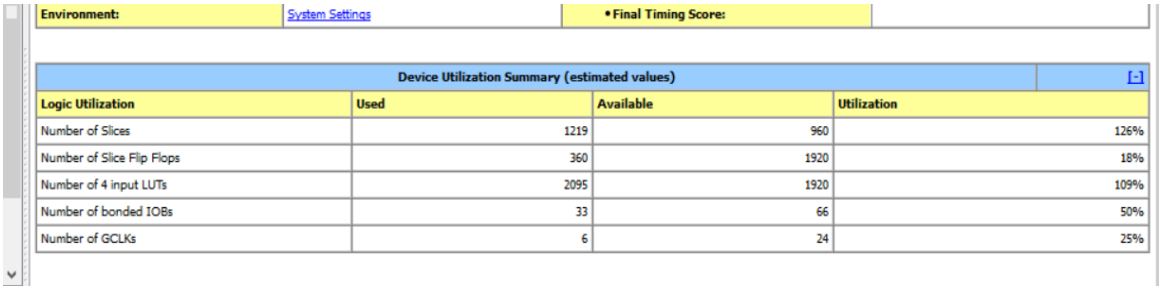


Fig 7: Final Output

The simulation output waveforms from Modelsim confirm that the logic behaves as expected for all the tested input combinations.

B. Xilinx Synthesis Report:

Existing Multiplier : Number of slices 1219 and utilization 126%.

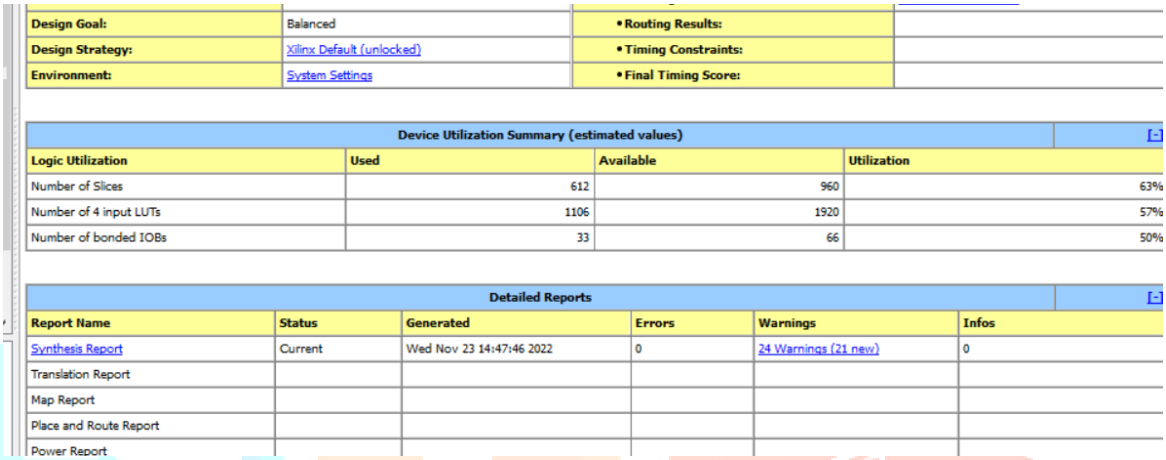


Environment:	System Settings	• Final Timing Score:	

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	1219	960	126%
Number of Slice Flip Flops	360	1920	18%
Number of 4 input LUTs	2095	1920	109%
Number of bonded IOBs	33	66	50%
Number of GCLKs	6	24	25%

Fig 4 : Existing Scheme result

Proposed Multiplier : Number of slices 612 and utilization 63%.



Design Goal:	Balanced	• Routing Results:	
Design Strategy:	Xilinx Default (unlocked)	• Timing Constraints:	
Environment:	System Settings	• Final Timing Score:	

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	612	960	63%
Number of 4 input LUTs	1106	1920	57%
Number of bonded IOBs	33	66	50%

Detailed Reports					[-]
Report Name	**Status**	**Generated**	**Errors**	**Warnings**	**Infos**
Synthesis Report	Current	Wed Nov 23 14:47:46 2022	0	24 Warnings (21 new)	0
Translation Report					
Map Report					
Place and Route Report					
Power Report					

Fig 5: Proposed scheme result

Fig 4 The Device Utilization Summary shows that the design exceeds available resources for slices (126%) and LUTs (109%), indicating overutilization. Fig 5 The Device Utilization Summary indicates efficient resource usage with slices at 63% and LUTs at 57%, showing the design fits well within available limit.

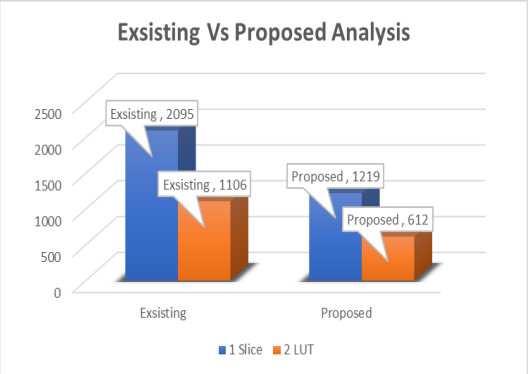
C.Comparison of Existing System and Proposed System Results:

The proposed has been simulated and the synthesis report can be obtained by using Xilinx ISE 12.1i. The various parameters used for computing existing and proposed systems with Spartan-3 processor.

- More slices used = More hardware area consumed. This is observed in traditional existing multipliers.
- Fewer slices = More optimized and compact design. This is observed in proposed hybrid multipliers.

S.No	Parameter	Existing	Proposed
1.	Slice	2095	1219
2.	LUT	1106	612

- 41.8% reduction in Slices.
- 44.6% reduction LUTs.



- Shows marked improvements in power , delay and area. Confirms the trade-off between precision and efficiency is well managed.
- Comparing to a conventional multiplier , our design used significantly fewer slices and LUTs.

V. CONCLUSION AND FUTURE SCOPE

A. Conclusion

In conclusion, designing a compressor-based multiplier using Verilog offers significant advantages in terms of speed and efficiency for performing multiplication operations in digital circuits. By carefully selecting the appropriate compressor design and optimizing the carry propagation logic, you can create a multiplier that meets your performance requirements while minimizing hardware resources. However, it's important to note that the specific design choices and trade-offs will depend on your project's requirements, such as speed, area constraints, and power consumption. Therefore, it's crucial to thoroughly test and validate your Verilog multiplier design to ensure it meets the desired functionality and performance goals. Additionally, as technology evolves, new design techniques and tools may become available, so it's important to stay updated with the latest advancements in digital design to create more efficient and optimized multipliers

B. Future Scope

The modified approximate compressor-based hybrid multiplier presents significant potential for future applications, particularly in the domain of low-power and error-tolerant computing. As the demand for energy-efficient hardware continues to grow, especially in IoT devices, wearable electronics, and biomedical instruments, this multiplier can play a vital role by reducing power consumption while maintaining acceptable accuracy levels. In fields like machine learning and image processing, where exact computation is not always necessary, this design offers an effective trade-off between performance and precision. Furthermore, its architecture can be scaled to higher bit-widths, making it suitable for more complex digital signal processing tasks. There is also considerable scope for real-time implementation on FPGAs and ASICs, enhancing its practical relevance in embedded systems. Future work could explore adaptive approximation techniques, allowing the multiplier to dynamically adjust its behavior based on workload requirements or energy constraints, thus improving its versatility and efficiency in diverse applications.

REFERENCES

[1] AmritaNanda,"Design and Implementation of Urdhva-Tiryakbhyam Based Fast 8×8 Vedic Binary Multiplier"IJERT, ISSN: 2278-0181,Vol. 3 Issue 3, March - 2014

[2] Poornima M, Shivaraj Kumar Patil, Shivukumar, ShridharKP,Sanjay H, "Implementation of Multiplier Using Vedic Algorithm", JITEE, ISSN:-2278-3075, Volume-2, Issue-6,May-2013.

[3] Premananda B.S, Samarth S. Pai, Shashank B, ShashankS.Bhat, "Design and Implementation of 8-bit Vedic Multiplier", IJAREEIE, Vol.2, Issue 12, ISSN: 2320-3765, Dec-2013.

[4] Anju& V.K. Agrawal,"FPGA Implementation of Low Power and High Speed Vedic Multiplier using Vedic Mathematics", IOSR-JVSP , e-ISSN: 2319 – 4200 ,2, Issue 5 (May. – Jun. 2013), PP 51-57

[5] Booth, A.D., "A signed binary multiplication technique," Quarterly Journal of Mechanics and Applied Mathematics, vol. 4, pt. 2, pp. 236– 240, 1951

[6] Jagadguru,Swami Sri Bharath, KrsnaTirathji, "Vedic Mathematics or Sixteen Simple Sutras From The Vedas", MotilalBanarsidas, Varanasi(India),1986

[7] Mrs. M. Ramalatha, Prof. D. Sridharan, "VLSI Based High Speed Karatsuba Multiplier for Cryptographic Applications Using Vedic Mathematics", IJSCI, 2007.

[8] L. Ciminiera and A. Valenzano, "Low cost serial multipliers for high speed specialised processors," Computers and Digital Techniques, IEEE Proc., vol. 135.5, 1988, pp. 259-265.

[9] P. Verma, K.K. Mehta, "Implementation of an efficient multiplier based on Vedic Mathematics using EDA Tool", International Journal of Engineering and Advance Technology (IJEAT) ISSN : 1 (5), 2012, 2249-8958.

[10] Harpreet Singh Dhillon and AbhijitMitra, "A Reduced– Bit Multipliction Algorithm for Digital Arithmetics", International Journal of Computational and Mathematical Sciences 2.2 @ www.waset.orgSpring2008

[11] P. D. Chidgupkar and M. T. Karad, "The Implementation of Vedic Algorithms in Digital Signal Processing", Global J. of Engg. Edu, Vol.8, No.2, 2004, UICEE Published in Australia

[12] Vahdat, Shaghayegh; Kamal, Mehdi; Afzali-Kusha, Ali; Pedram, Massoud (2019). *TOSAM: An Energy-Efficient Truncation- and Rounding-Based Scalable Approximate Multiplier. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, (), 1–13.* doi:10.1109/TVLSI.2018.2890712

[13] cui, xiaoping; Liu, Weiqiang; chen, Xin; Swartzlander, Earl; Lombardi, Fabrizio (2015). *A Modified Partial Product Generator for Redundant Binary Multipliers. IEEE Transactions on Computers, (), 1–1.*

[14] Li, Yin; Zhang, Yu; He, Wei (2020). *Fast Hybrid Karatsuba Multiplier for Type II Pentanomials. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 28(11), 2459–2463.*

[15] Tsoumanis, Konstantinos; Axelos, Nikos; Moschopoulos, Nikos; Zervakis, Georgios; Pekmestzi, Kiamal (2015). *Pre-Encoded Multipliers Based on Non-Redundant Radix-4 Signed-Digit Encoding. IEEE Transactions on Computers, (), 1–1.*