"Efficient Low-Power Arithmetic Unit Design Using Booth Multiplication Technique"

Rana S Mahajan[1] ,Vinod S Mahajan[2], Sachin B Borse[3]

[1]Assistant Professor, Department of E&TC Engineering, Sir Visvesvaraya Institute of Technology, Nashik, Maharashtra, India

[2]Assistant Professor, Department of Computer, D N Patel College of Engineering ,Shahada, Maharashtra, India

[3] Assistant Professor,Department of E&TC Engineering, Late G N Sapkal College of Engineering, Nashik , Maharashtra, India

Abstract -Low power consumption and smaller space are a number of the foremost vital criteria for the fabrication of DSP systems and superior systems. Optimizing the speed and space of the number may be a major style issue. However, space and speed are sometimes conflicting constraints in order that up speed results principally in larger areas. In our project we have a tendency to try and verify the most effective resolution to the current downside by comparison some multipliers. the world and speed of the number is a vital issue, increment in speed ends up in massive space consumption and contrariwise. Multipliers play important role in most of the superior systems. Performance of a system depends to a good extent on the performance of number so multipliers ought to be quick and consume less space and hardware. this concept forced United States of America to check and review regarding the Booth's algorithmic rule.

Keywords: Squaring, Vedic Mathematics, Yavadunam Sutra, Bit reduction.

I. INTRODUCTION

Multiplication and Division are most basic and frequently used operations in a CPU. These operations also form the basis for other complex operations. With ever increasing need for faster clock frequency it becomes imperative to have faster arithmetic unit. In this work a replacement system of arithmetic – sacred writing arithmetic is employed to implement operations. sacred writing arithmetic is predicated on sixteen formulas with the aim of simplification of long and cumbersome arithmetic. sacred writing arithmetic contains multiple algorithms for one operation. within the current work these algorithms square measure evaluated for his or her quality in binary arithmetic. appropriate algorithms square measure enforced for Multiplication, Squaring and Division in

Verilog. Multiplication and Squaring square measure any utilized in style of trigonometric (function circular function) and cos function implementations. As a student in Republic of India, sacred writing arithmetic could be a name that is detected repeatedly with relation to the techniques for determination arithmetic drawback mentally. one in every of the most functions of sacred writing arithmetic is to rework the tedious calculations into easier, orally manageable operation while not a lot of facilitate of pen and paper. Any normal human will perform mental operations for terribly tiny magnitude of numbers and thence sacred writing arithmetic provides techniques to resolve operations with massive magnitude of numbers simply. sacred writing arithmetic provides quite one technique for basic operations like multiplication and division. for every operation there's a minimum of one generic technique provided together with some strategies that square measure directed towards specific cases simplifying the calculations any.Today‟s CPUs square measure more and more performing on higher frequencies with reduction in size of semiconductor. Arithmetic and Logic Unit - ALU is one in every of the foremost necessary and demanding blocks in mainframe. thence it's imperative to possess quick and economical ALU. Division is that the most time intense amongst the essential mathematical calculation. In today‟s computing technology functions like trigonometric function and of times needed and are enforced in hardware. With these considerations, it is always important to have fast and efficient mechanism to implement mathematical functions. Vedic mathematics provides algorithms to simplify the mathematics and hence is perfect solution for the problem stated.

2. GOAL OF STUDY WORK

The goal of this work is to understand and implement the formulas of Vedic Mathematics for binary numbers and assess their use in Computer Arithmetic. The different mathematical functions explored are

- Multiplication – 8 bit, 16 bit and 23 bit multiplication are implemented and synthesized, timing, area and power of the hardware designed is evaluated.

- Squaring – Similar to multiplication 8 bit, 16 bit and 23 bit i.e. single precision squaring hardware design are implemented and evaluated.

- Sine and Cosine functions are implemented by polynomial approximation and interpolation method and performance is observed.

- Division algorithm for a 16 by 8 bit division is implemented in Verilog and performance is evaluated.

3. RELATED WORK

Many have proposed the use of Vedic mathematics methods for multiplication, but many of them restrict the implementation to 8 bits. [5] Implements a 2, 3, 4 and 8 bit multipliers by Nikhilam formula and also concludes that multiplication by generic formula Urdhvatiryagbhyam is slower because of the structure. [6] Argues that there's no distinction between multiplication done by sacred text and standard technique. [7] Implements complex quantity number exploitation sacred text arithmetic strategies and concludes that it's appropriate for top speed complicated arithmetic circuits. [8] Implements cubing circuit by Anurupyena formula by totally different structures for eight bit and infers sacred text carry save cubing is best. [3] Uses sacred text arithmetic formula for 2×2 multiplication and uses it as a block in higher breadth multiplication, for synthesis XC2S100-5tq144 was used. It will be determined that each one the previous work is said to solely multiplication and most of the numbers ar enforced exploitation formula Nikhilam or Anurupyena – a corollary of Nikhilam with all multiplier implementation in hot water eight bits or lower. [9] And [2] use multiplication in applications like FFT and elliptic curve cryptography.

4. ARITHMETIC OPERATION OF MULTIPLICATION

Multiplication is the most important arithmetic operation in signal processing applications. All the signal and data processing operations involve multiplication. As speed is usually a constraint within the multiplication operation, increase in speed are often achieved by reducing the quantity of steps within the computation method. The speed of number determines the potency of such a system. In any system style, the 3 main constraints that confirm the performance of the system ar speed, space and power demand. sacred writing arithmetic [1] was reconstructed from the traditional Indian scriptures (Vedas) by Hindu Bharati avatar Tirthaji prince (1884-1960) once his eight years of analysis on Vedas. sacred writing arithmetic is especially supported sixteen principles or word-formulae that ar termed as sutras. this can be a awfully attention-grabbing field and presents some effective algorithms which may be applied to varied branches of engineering like computing and digital signal process. desegregation multiplication with sacred writing arithmetic techniques would

lead to the saving of process time. Thus, desegregation sacred writing arithmetic for the number style can enhance the speed of multiplication operation The multiplier architecture is based on Urdhva Tiryagbhyam [4] (vertical and cross-wise algorithm) sutra. An illustration of Urdhva Tiryagbhyam sutra is shown in Figure 1.

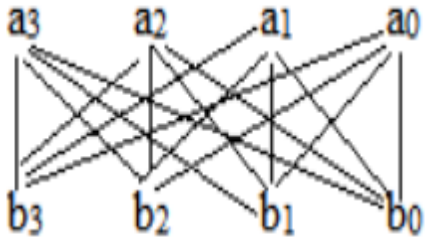

Figure 1. Illustration of Urdhva Tiryagbhyam sutra.

The 4x4 multiplication has been done in a single line in Urdhva Tiryagbhyam sutra [1], whereas in shift and add (conventional) method, four partial products have to be added to get the result. Thus, by using Urdhva Tiryagbhyam Sutra in binary multiplication, the number of steps required calculating the final product will be reduced and hence there is a reduction in computational time and increase in speed of the multiplier. Consider two 4-bit binary numbers a3a2a1a0 and b3b2b1b0. The partial products [5] (P7 P6 P5 P4 P3 P2 P1 P0) generated are given by the following equations:

i. P0= a0b0

ii. P1= a0b1 + a1b0

iii. P2 = a0b2 + a1b1 + a2b0+ P1

iv. P3= a0b3 + a1b2 + a2b1 + a3b0+ P2

v. P4 = a1b3 + a2b2 + a3b1 + P3

vi. P5 = a1b2 + a2b1 + P4

vii. P6 = a3b3 + P5

viii. P7 = carry of P6

5. DESIGN OF 4X4 MULTIPLIER

The 4-bit Vedic multiplier is designed using Urdhva Tiryagbhyam sutra and carry-skip technique for partial product addition. In a normal Vedic multiplier, the carry from each partial product addition is given to the next partial product bit calculation. In the proposed architecture (Figure 2), the carries are not only rippled to the next partial product bit calculations but also to the subsequent bits using carry skip technique so as to reduce the carry propagation delay.

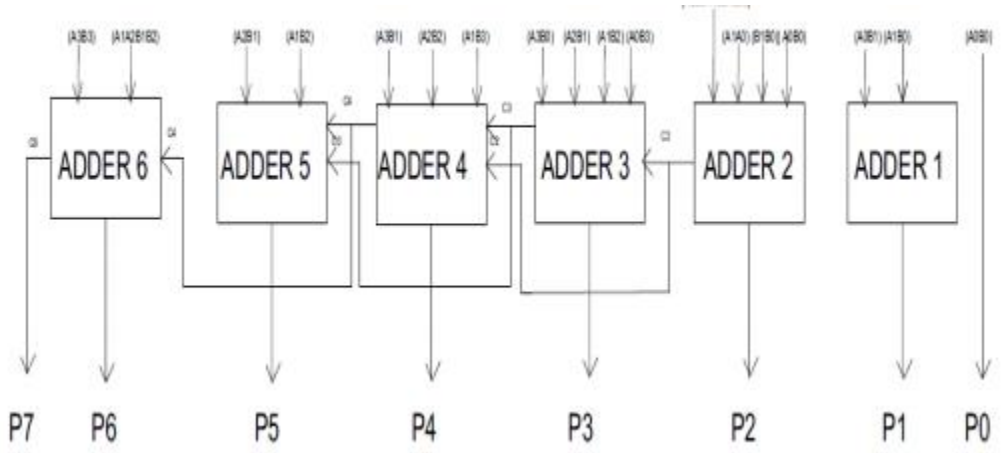

Figure 2. Proposed 4-bit Vedic multiplier using Urdhva Tiryagbhyam sutra and carry-skip technique.

The Vedic multiplier equations given in equation 1 are modified in the proposed multiplier as follows:

i. P0 = a0b0

ii. P1 = a1b0 + a0b1

iii. P2 = a2b0 + a1b1 + a0b2 + a0a1b0b1

iv. P3 = a3b0 + a2b1 + a1b2 + a0b3 + carry from P2

v. P4 = a3b1 + a2b2 + a1b3 + carry from P2 + carry from P3

vi. P5 = a3b2 + a2b3 + carry from P4 + carry from P3

vii. P6 = a3b3 + a1a2b1b2+ carry from P4

viii. P7 = carry from P6

The products P0, P1, P3 and P7 in equation 2 are same as in equation 1 but the other equations have been modified to incorporate carry skip technique in the adder blocks of figure 2. In P2 a new term a0b0a1b1 is added so that all the four terms are ready for addition at the same time to reduce the delay in waiting for carry from the previous stage. In P4 and P5 the carry from two previous stages are added thus carry propagation time is reduced. For example if all four terms of P2 are 1, the carry is directly transferred to P4. Similarly P6 is modified by adding carry from P4 and term a1a2b1b2.

6. PROPOSED MULTIPLIER

Booth's multiplication algorithm is a multiplication algorithm that multiplies two signed binary numbers in two's complement notation. The algorithm was invented by Andrew Donald Booth in 1950 while doing research on crystallography at Birkbeck College in Bloomsbury, London. Booth's algorithm is of interest in the study of computer architecture.

Booth's algorithm examines adjacent pairs of bits of the 'N'-bit multiplier Y in signed two's complement representation, including an implicit bit below the least significant bit, $y_{-1} = 0$. For each bit y_i, for i running from 0 to $N - 1$, the bits y_i and y_{i-1} are considered. Where these two bits are equal, the product accumulator P is left unchanged. Where $y_i = 0$ and $y_{i-1} = 1$, the multiplicand times 2^i is added to P; and where $y_i = 1$ and $y_{i-1} = 0$, the multiplicand times 2^i is subtracted from P. The final value of P is the signed product.

The representations of the number and products don't seem to be specified; generally, these area unit each conjointly in two's complement illustration, just like the multiplier factor, however any numeration system that supports addition and subtraction can work in addition. As declared here, the order of the steps isn't determined. Typically, it takings from LSB to MSB, beginning at i = 0; the multiplication by 2i is then generally replaced by progressive shifting of the P accumulator to the proper between steps; low bits is shifted out, and sequent additions and subtractions will then be done simply on the very best N bits of P.[2] There area unit several variations and optimizations on these details..... The algorithm is often described as converting strings of 1s in the multiplier to a high-order +1 and a low-order −1 at the ends of the string. When a string runs through the MSB, there is no high-order +1, and the net effect is interpretation as a negative of the appropriate value.

Find 3 × (−4), with **m** = 3 and **r** = −4, and $x = 4$ and $y = 4$:

- m = 0011, -m = 1101, r = 1100
- A = 0011 0000 0
- S = 1101 0000 0
- P = 0000 1100 0
- Perform the loop four times:
    1. P = 0000 110**0 0**. The last two bits are 00.
        - P = 0000 0110 0. Arithmetic right shift.
    2. P = 0000 011**0 0**. The last two bits are 00.
        - P = 0000 0011 0. Arithmetic right shift.
    3. P = 0000 001**1 0**. The last two bits are 10.
        - P = 1101 0011 0. P = P + S.
        - P = 1110 1001 1. Arithmetic right shift.
    4. P = 1110 100**1 1**. The last two bits are 11.
        - P = 1111 0100 1. Arithmetic right shift.
- The product is 1111 0100, which is −12.

The above-mentioned technique is inadequate when the multiplicand is the most negative number that can be represented (e.g. if the multiplicand has 4 bits then this value is −8). One possible correction to this problem is to add one more bit to the left of A, S and P. This then follows the implementation described above, with modifications in determining the bits of A and S; e.g., the value of **m**, originally assigned to the first x bits of A, will be assigned to the first x+1 bits of A. Below, the improved technique is demonstrated by multiplying −8 by 2 using 4 bits for the multiplicand and the multiplier:

- A = 1 1000 0000 0
- S = 0 1000 0000 0
- P = 0 0000 0010 0
- Perform the loop four times:
    1. P = 0 0000 001**0 0**. The last two bits are 00.

- P = 0 0000 0001 0. Right shift.

2. P = 0 0000 000**1 0**. The last two bits are 10.
   - P = 0 1000 0001 0. P = P + S.
   - P = 0 0100 0000 1. Right shift.
3. P = 0 0100 000**0 1**. The last two bits are 01.
   - P = 1 1100 0000 1. P = P + A.
   - P = 1 1110 0000 0. Right shift.
4. P = 1 1110 000**0 0**. The last two bits are 00.
   - P = 1 1111 0000 0. Right shift.

- The product is 11110000 (after discarding the first and the last bit) which is −16.

7. OBJECTIVE AND PERFORMANCE CITERIA

Our objective is to study about the performance of Combinational Multiplier and Booth multiplier using Simulator The performance of the any processor will depend upon its power and delay. The power and delay should be less in order to get a effective processor. In processors the most commonly used architecture is multiplier. If the power and delay of the multiplier is reduced then the effective processor can be generated [2]. Low power consumption and smaller area are some of the most important criteria for the high performance systems. Area of the circuit depends on the number of logic gates used in the circuit.

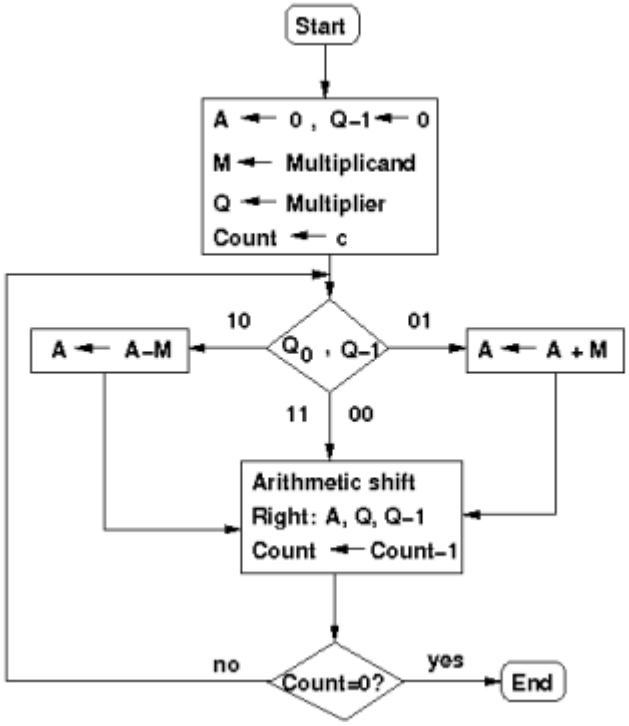

Figure-3 Design flow of Booth algorithm.

8. RESULT AND DISCUSSION

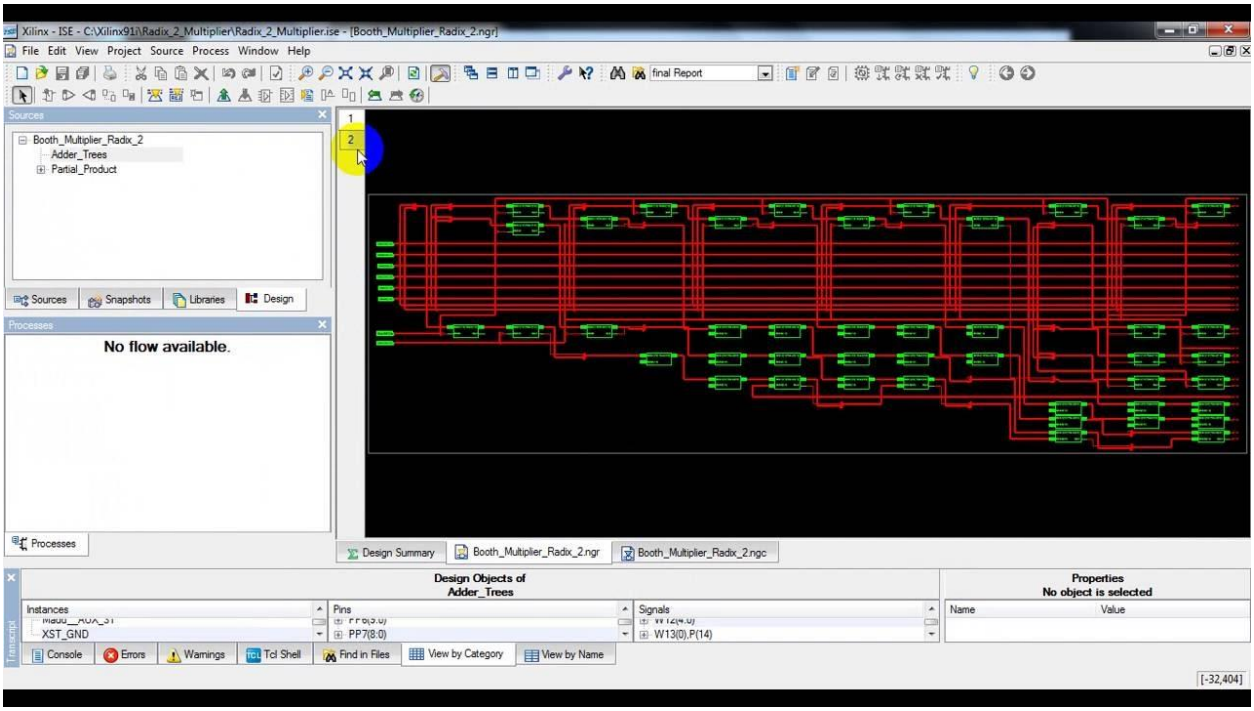

Figure-4 generated waveform of combinational multiplier.

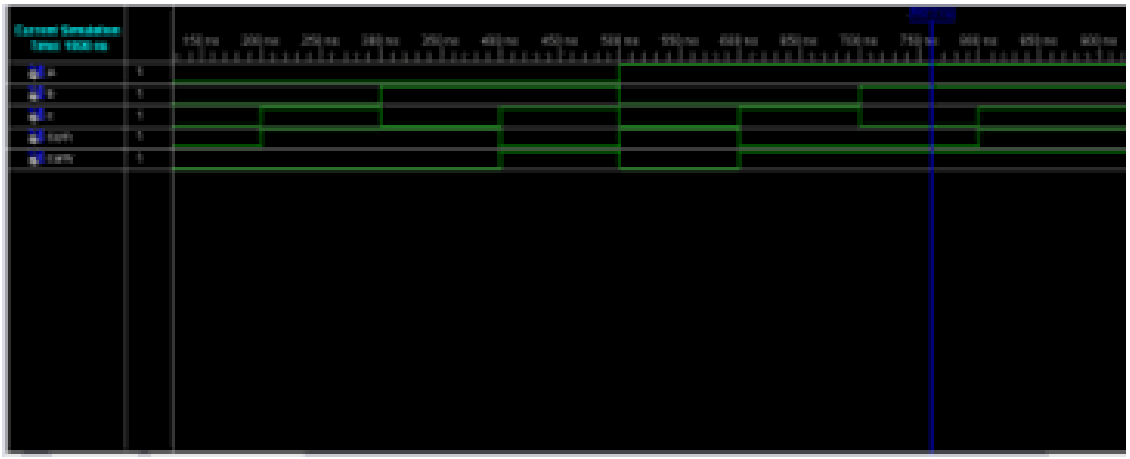

Figure-5 implementation of booth multiplier on simulator

Methods	Power(Microwatt)
Modified with 4-inputs NAND gate	40.01
Our result	29.21

The method of Booth multiplication reduces the numbers of adders and hence the delay required to produce the partial sums. The high performance of booth multiplier comes with the drawback of power consumption. The reason is large number of adder cells required that consumes large power [5].

9. CONCLUSION

It can be concluded that Booth Multiplier is superior in respect like area, Complexity. In booth multiplier number of gates is reduced and hence area of booth multiplier is less than combinational multiplier. However Combinational Multiplier gives optimum number of components required. Hence for less delay requirement Booth's multiplier is suggested. Further work can be carried out on this project in the power estimation section and to improve the speed or to minimize

the delay and power of multipliers. Booth Multiplier can be further extended to- • Modified booth multiplier • Radix -2 booth multiplier • Radix- 4 booth multiplier.

REFERENCES

[1] Arushi Somani, Dheeraj Jain, Sanjay Jaiswal, Kumkum Verma and Swati Kasht, "Compare Vedic Multipliers with Conventional Hierarchical array of array multiplier", International Journal of Computer Technology and Electronics Engineering (IJCTEE) Vol.2, No. 6, (2012),.

[2] Bengali S.,"Vedic Mathematics and Its Applications in Computer Arithmetic",http://repository.lib.ncsu .edu/ir/bitstream/1840.16/7232/1/etd.pdf, (2011).

[3] Deepa A. and Marimuthu C N., "A High Speed VLSI Architecture of a Pipelined Reed Solomon Encoder for Data Storage in Communication Systems", Asian Journal of Research in Social Science and Humanities,Vol.7,No.2,pp.228-238,(2017),.

[4] Dilli Kumar B, Bharati M, "A High Speed and Efficient Design for Binary Number Squaring using Dwandwa Yoga", International Journal of Advanced Research in Computer Engineering & Technology Vol. 1,No.4, (2012).

[5] Ghosh M, "Design and Implementation of Different Multipliers Using VHDL", http://ethesis.nitrkl.ac.in/66/1/moumita.pdf, (2017).

[6] Jagadguru Swami Sri Bharati Krishna Tirthaji Maharaj,"Vedic Mathematics", Delhi: Motilal Banarsidas Publishers Pvt.Ltd. , (2009).

[7] Kabiraj Sethi and Rutuparna Panda, "An improved Squaring circuit for binary numbers", International Journal of Advanced Computer science and Applications, Vol.03, No.02, (2012).

[8] Pabitra Kumar Mohapatra, Siba Kumar Panda, Sambita Dalal, Shibashis Pradhan,"VHDL implementation of a Novel Low power squaring circuit using YTVY algorithm of Vedic Mathematics ", International Journal of Emerging Trends in Science and Technology, Vol.02, No.3, pp.2139- 2146, (2015).

[9] Prabha S.Kasliwal, B.P.Patil and D.K.Gautam,"Performance evaluation of squaring operation by Vedic Mathematics,"IETE Journal of Research, pp.39-41, (2011).

[10] Shubham Mishra, Shailendra Kumar Dhakad,"A High Speed and device efficient FPGA based Squaring circuit", International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering, Vol.02, No.11, (2013).

[11] Deepa A. and Marimuthu C N.," VLSI Implementation of a Squaring Architecture Based On Yavadunam - An Algorithm of an Ancient Indian Vedic Mathematics", paper submitted to a journal and not yet selected.

[12] Anitha R., Pradeep Kumar M., "Design of Low Power Booth Multiplier Using Modified Booth Encoding Technique", International Journal of Engineering Research & Technology (IJERT), Vol.8, No.5, (2019).

[13] Kiran Kumar V., Lakshmi Devi K., "FPGA Implementation of High-Speed Modified Booth Multiplier", International Journal of Innovative Technology and Exploring Engineering (IJITEE), Vol.9, No.2, (2019).

[14] Singh A., Verma R., "Design and Analysis of Low Power Multiplier Using Booth Algorithm", International Journal of Advanced Science and Technology, Vol.29, No.3, pp. 456–462, (2020).

[15] Sharma P., Gupta S., "Performance Comparison of Array, Wallace and Booth Multipliers in VLSI Design", International Journal of Computer Applications, Vol.176, No.24, (2020).

[16] Reddy K. H., Prasad T. V., "Low Power and Area Efficient Modified Booth Multiplier Using VHDL", Journal of Electronics and Communication Engineering, Vol.15, No.4, (2020).

[17] Mehta D., Joshi H., "Design of Energy Efficient Multiplier Using Radix-4 Booth Algorithm", International Journal of Engineering and Advanced Technology (IJEAT), Vol.10, No.1, (2021).

[18] Kumar N., Singh P., "High Speed and Low Power VLSI Architecture of Booth Multiplier Using Compressor Adders", Microelectronics Journal, Vol.112, (2021).

[19] Patil S., Kulkarni A., "FPGA Based Implementation of Optimized Booth Multiplier for DSP Applications", International Journal of Reconfigurable Computing, Vol.2021, Article ID 6678923, (2021).

[20] Das S., Roy A., "Comparative Study of Low Power Multipliers Using Booth and Vedic Algorithms", Journal of Circuits, Systems and Computers, Vol.30, No.12, (2021).

[21] Choudhary R., Jain S., "Area and Power Efficient Multiplier Design Using Modified Booth Algorithm and Wallace Tree", International Journal of Electronics, Vol.109, No.3, pp. 389–402, (2022).

[22] Verma N., Tiwari M., "Design of Low Power High Speed Multiplier Using Hybrid Booth Encoding Technique", International Journal of Electrical and Computer Engineering (IJECE), Vol.12, No.2, pp. 1456–1464, (2022).