Multiplication Technique in Residue Number System

Tukur Gupta, Shamim Akhter, Shaheen Khan, Saurabh Chaturvedi

***Abstract:** This paper presents the modulo multiplication technique in residue number system (RNS) using Vedic mathematics. Residue number system supports fast mathematsical computation. In this paper, the use of the combination of RNS and Vedic mathematics has improved the computation time for modulo multiplication operation. The proposed modulo multiplier is implemented in VHDL and synthesized using Xilinx ISE 14.1.The performance comparison analysis in terms of area, power and delay is done between the proposed technique and direct computation. The performance of the multiplier circuit has been compared using the 32 nm standard cells available in Synopsys Design Compiler. The presented Vedic modulo multiplier is efficient in terms of speed for large input data sizes.*

***Index Terms:** HDL design, Vedic mathematics, residue number system (RNS), lookup table (LUT), modular multiplication*

I. INTRODUCTION

High-speed multiplier is necessity in all multiplication operation, including digital signal processing, image processing and others. Therefore, a lot of focus has been put on the development of area efficient and fast multipliers. Vedic multiplier [1], [2] is considered as one of the most efficient binary multipliers because of its modular approach. The architecture of a Vedic multiplier is based on the vertical and crosswise algorithm of the ancient Indian Vedic mathematics.Residue number system (RNS) [3] is a non-weighted number system. The parallel nature for the mathematical operations in RNS results in a faster computation. In RNS, an integer number is represented as a set of residues, and these residues are processed separately. In this paper, the Vedic multiplication technique proposed in [2] has been applied on RNS, which avoided the requirement of larger sized multiplier units. The organization of this paper is as follows: Section II presents the analysis and design for the Vedic-based multiplication operation in RNS domain. Section III deals with the design strategies for different moduli set. Section IV compares the proposed modular multiplication technique with the direct computation method. Finally, the conclusions are drawn in Section V.

Revised Manuscript Received on October 30, 2019.

* Correspondence Author
Tukur Gupta*, Ph.D. Scholar, Department of Electronics and Communication Engineering (ECE), Jaypee Institute of Information Technology, NOIDA, India
Shamim Akhter, Assistant Professor, Department of ECE, Jaypee Institute of Information Technology, NOIDA, India
Shaheen Khan, Assistant Professor, Department of ECE, Mewat Engineering College (Waqf), Haryana, India
Saurabh Chaturvedi, Assistant Professor, Department of ECE, Jaypee Institute of Information Technology, NOIDA, India

II. VEDIC-BASED MULTIPLICATION IN RNS

Modulo multiplication of two inputs X and Y in RNS is represented by $|X.Y|_m$, where m is the modulo number. Direct computation requires a multiplier to compute X.Y followed by mod-m operation to obtain the final result. The implementation of a multiplier circuit demands a large hardware in terms of logic gates. The computation of mod-m requires a lookup table (LUT), where the address size of an LUT is related to the number of bits in the input data. As an example, for 8-bit multiplication, an LUT requires 16-bit address. The LUT content width is decided by the value of moduli set m. The requirements of multiplier and respective LUT size for direct modulo multiplication are given in Table I.

Table I. Multiplier and LUT requirements for m = 7

No. of input bits (N)	Multiplier size (bits)	LUT address size (bits)
4	4×4	8
8	8×8	16
16	16×16	32
32	32×32	64

In this paper, the multiplication based on Vedic mathematics is proposed to reduce the requirements of multiplier and LUT. In the proposed approach, the input bits are divided into two equal parts, i.e. for N×N multiplication, divide the multiplicand and multiplier into two parts, consisting of (N to N/2+1) bits and (N/2 to 1) bits. As shown in Fig. 1, input data X and Y are divided as X_MX_L and Y_MY_L. Thereafter, mod-m is computed for all the four subdivided parts followed by Vedic-based cross product computation.

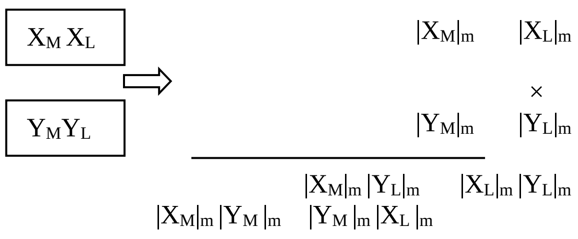

Fig. 1. Vedic-Based Cross Product Computation.

For an example, with m = 7, the modulo of an input can be represented using maximum of 3 bits. As depicted in Fig. 1, with m = 7, the subdivided parts are multiplied using a small sized multiplier because each input to the multiplier is of 3 bits.

The complete block diagram of the proposed architecture is demonstrated in Fig. 2 for m = 7. The partial products after Vedic multiplication followed by mod-7 are shown as: P_3, P_2, P_1 and P_0. The value of P_1 and P_2 is shifted by 1-bit and that of P_3 by 2-bit. This is because of the fact that the divided subparts are multiples of 16, therefore a factor of 2 is taken to compensate for the factor of $|16|_7$, i.e. 2, for P_1 and P_2. Similarly, for P_3, a factor of 4 is taken to compensate for $|256|_7$, i.e. 4.

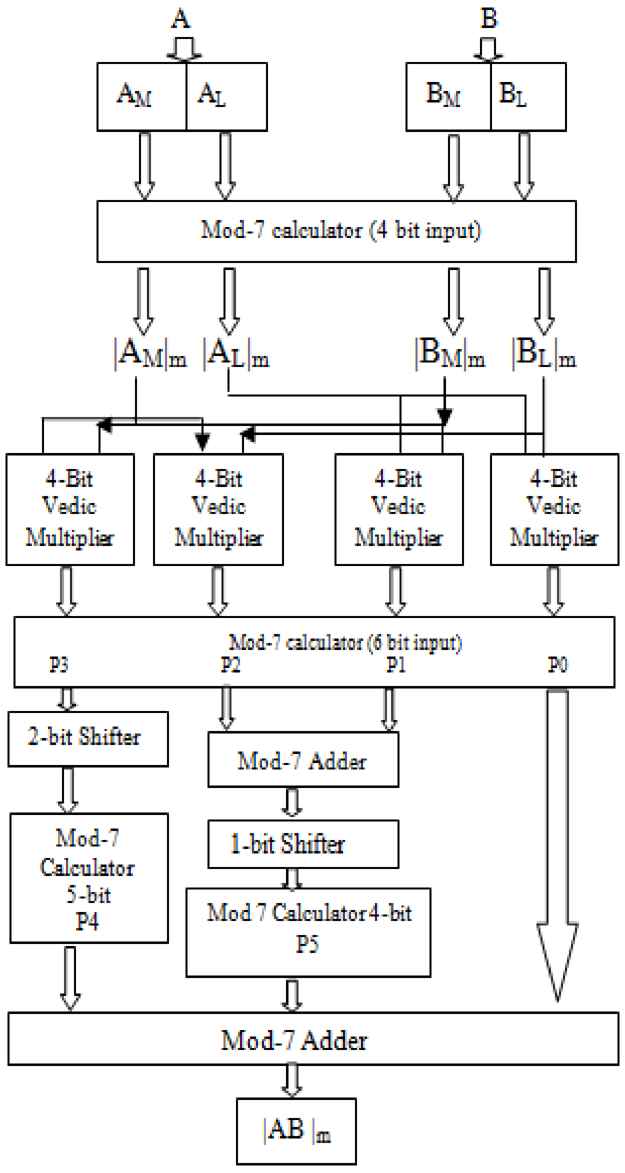

Fig. 2. Proposed RNS domain Multiplier for m = 7.

As an example, if input A = 10111111, then subdivided parts will be: A_M = 1011 and A_L = 1111. Similarly, if B = 11101001, then B_M = 1110 and B_L = 1001. These values are given to mod-7 calculator (in this case m = 7) as inputs.

Fig. 3 displays the architecture of the mod-7 calculator [3], and it can be extended to any modulo number with a modification in LUT size, contents and modulo adder.

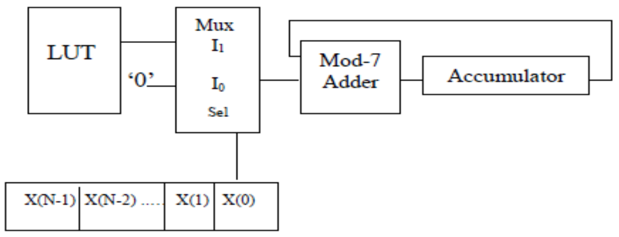

Fig. 3. Mod-7 calculator [3]

For 8-bit modulo multiplication, the subdivided parts of A and B are of 4-bit each, therefore LUT has only four address locations. The LUT contents for m = 7 are listed in Table II.

Table II. LUT Contents For Mod-7 For 4-Bit Input Data

LUT address	LUT contents
00	001
01	010
10	100
11	001

As discussed in [3] and shown in Fig. 3, the LUT contents are derived based on value of $|2^j|_m$, where j varies from 0 to 3 for m = 7. The value of input data sequence selects either the LUT content or '0' using a 2×1 multiplexer. For an example, consider the computation of $|B_L|_7$ with B_L = 1001, the sequence of data out from the multiplexer will be:

001 = > 000 => 000 => 001

The extracted value from the multiplexer is recursively used in the modulo-7 adder [3]. The modulo-m adder produces $|A+B|_m$. For this example, the result of iteration is given below, with an assumption that the accumulator is initially clear.

$|000 + 001|_7$ => 001
$|001 + 000|_7$ => 001
$|001 + 000|_7$ => 001
$|001 + 001|_7$ => 010

The final result is 010 which is 2 in decimal, and it represents the value of $|B_L|_7$, i.e. $|9|_7 = 2$. Various architectures of modulo-m calculator and adder are presented in [4]-[7]. The block diagram of a 4-bit modified Vedic multiplier is discussed in [2] and shown in Fig. 4 with the inputs A and B. This architecture requires four 2-bit binary multipliers, a carry save adder (CSA), an OR gate and a 2-bit increment-by-one (IBO) circuit. The outputs from the 2-bit multiplier are indexed from P_{15} to P_0. The final multiplication result is available at $S_7S_6.....S_1S_0$.

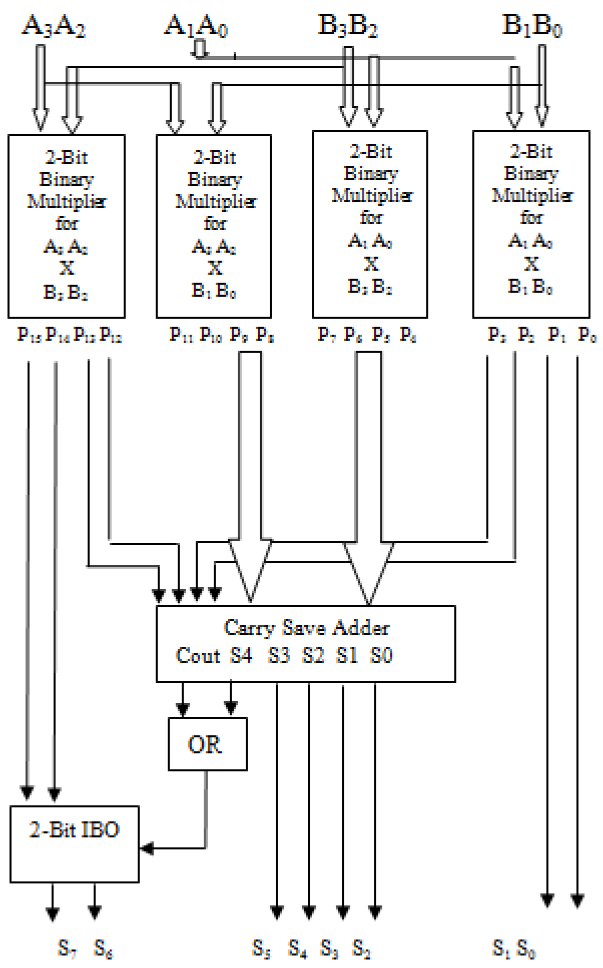
A_3A_2
A_1A_0
B_3B_2
B_1B_0
2-Bit Binary Multiplier for A_3A_2 X B_3B_2
2-Bit Binary Multiplier for A_3A_2 X B_1B_0
2-Bit Binary Multiplier for A_1A_0 X B_3B_2
2-Bit Binary Multiplier for A_1A_0 X B_1B_0
P_{15} P_{14} P_{13} P_{12}
P_{11} P_{10} P_9 P_8
P_7 P_6 P_5 P_4
P_3 P_2 P_1 P_0
Carry Save Adder
Cout S4 S3 S2 S1 S0
OR
2-Bit IBO
S_7 S_6
S_5 S_4 S_3 S_2
S_1 S_0

Fig. 4. Generalized block diagram for 4-bit modified Vedic multiplier

As shown in Fig. 2, the divided parts are of 4-bit each for an 8-bit multiplication. The mod-7 calculator gives 3-bit data because the values are less than 7. The output of mod-7 needs to be multiplied, therefore A_3 and B_3 are always '0' for the last input bit of each data of the 4-bit Vedic multiplier. The multiplication operation between A_3A_2 and B_3B_2 is reduced to ANDing of A_2 and B_2 at P_{12} and the terms P_{13}, P_{14} and P_{15} are '0'. Moreover, the multiplication output from each Vedic multiplier block is limited to the maximum value of 36, so the result can be accommodated in $S_5S_4S_3S_2S_1S_0$. Therefore, the OR gate and 2-bit IBO circuit can be removed. Similarly, the requirements for hardware optimization can be analyzed for other modulo numbers. As depicted in Fig. 2, the extracted 6-bit output from all the four Vedic multipliers is provided to mod-7 calculator. It produces P_3, P_2, P_1 and P_0 corresponding to mod-7 value. The value of P_1 and P_2 is added using a mod-7 adder, shifted left by 1-bit and then again passed through a mod-7 calculator. The obtained output is represented as P_5. Similarly, the value of P_3 is shifted left by 2-bit and then mod-7 is computed and represented as P_4. Finally, the values of P_0, P_4 and P_5 are added using a mod-7 adder to obtain the modulo multiplication result of $|X.Y|_m$. The proposed design is implemented in VHDL and synthesized using the Virtex-4 ML-402 board from Xilinx ISE 14.1.

III. DESIGN STRATEGIES FOR RNS VEDIC-BASED MULTIPLIER FOR DIFFERENT MODULI

The design proposed in Section II can easily be extended for larger input data sizes of multipliers (as a power of 2), such as N = 16 and N = 32. When N = 16, the subdivided parts will have the data size of 8-bit and the LUT should have eight address lines. The details of address and corresponding stored value in LUT are presented in Table III for 8-bit input data.

Table III. LUT contents for mod-7 for 8-bit data size

LUT address	LUT contents
000	001
001	010
010	100
011	001
100	010
101	100
110	001
111	010

The multiplier design proposed for m = 7 in Fig. 2 needs to be modified to work for other modulo numbers. The suggested modifications are shown in Table IV for different values of m and N.

Table IV. Suggested Modifications In The Proposed Design For Other Modulo Numbers

Mod (m)	Input data size (N)	Hardware requirements
3	N = 4, 8, 16, 32	2-bit multiplier, no need of shifting in middle and left stages w.r.t. Fig. 2
5	N = 4	2-bit multiplier, needs shifting left by 2-bit in middle stage and no shifting in left stage w.r.t. Fig. 2
	N = 8, 16, 32	Vedic multiplier of 4-bit, no need of shifting in middle and left stages w.r.t. Fig. 2
7	N = 8 or 32 $\longrightarrow$	Vedic multiplier of 4-bit, needs shifting left by 1-bit in middle stage and 2-bit in left stage w.r.t. Fig. 2
	N = 16 $\longrightarrow$	Needs shifting left by 2-bit in middle stage and 1-bit in left stage w.r.t. Fig. 2

The simulation results for the mod-7 modulo multiplier for N = 8, 16 and 32 are exhibited in Fig. 5. The design is tested by using different set of input data. The input and output signal values are represented in integer inside the waveform window for clarity.

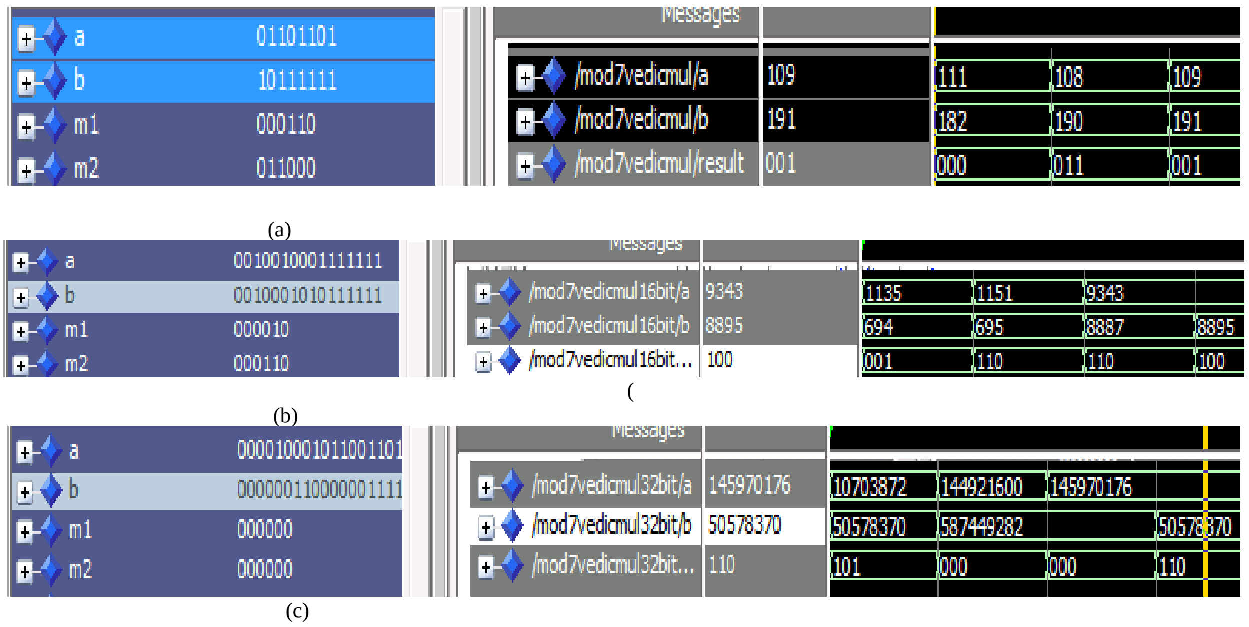
a 01101101
b 10111111
m1 000110
m2 011000
Messages
/mod7vedicmul/a 109 111 108 109
/mod7vedicmul/b 191 182 190 191
/mod7vedicmul/result 001 000 011 001
(a)
a 0010010001111111
b 0010001010111111
m1 000010
m2 000110
/mod7vedicmul16bit/a 9343 1135 1151 9343
/mod7vedicmul16bit/b 8895 694 695 8887 8895
/mod7vedicmul16bit... 100 001 110 110 100
(b)
a 0000100010110011101
b 0000001100000011111
m1 000000
m2 000000
/mod7vedicmul32bit/a 145970176 10703872 144921600 145970176
/mod7vedicmul32bit/b 50578370 50578370 587449282 50578370
/mod7vedicmul32bit... 110 101 000 000 110
(c)

Fig. 5. Simulation results of modulo multiplier for m = 7: (a) N = 8, (b) N = 16 and (c) N = 32.

IV. COMPARATIVE ANALYSIS

The binary adder is used as a fundamental building block in all the designs, including modulo adder, modulo calculator and Vedic multiplier. The basic ripple carry adder and carry look-ahead adder circuits are used in this work to compare the modulo multiplier architectures. Nonetheless, the synthesis results can be improved by using the advanced adders reported in [8]-[13]. The synthesis results of the proposed multiplication technique are compared with the direct computation using the Xilinx Virtex-4 ML-402 FPGA board. For direct modulo multiplication, the Vedic multiplier is used followed by the respective mod-7 calculator. In the work presented, this method is termed as Direct-I. In addition, the computation of $|A.B|_m$ can be done by first computing $|A|_m$ and $|B|_m$ and then computing mod-m of the product of $|A|_m$ and $|B|_m$. This method is termed as Direct-II.

Table V. Comparison Of Modulo Multipliers For Mod-7.

Input data size (N)	Technique	No. of slices	No. of LUTs	Delay (ns)
8	Proposed Direct-I Direct-II	79 113 59	143 202 106	12.96 15.63 16.02
16	Proposed Direct-I Direct-II	167 486 54	299 856 104	15.51 28.58 22.82
32	Proposed Direct-I Direct-II	151 1905 137	287 3341 262	18.81 53.78 27.08

The performance comparison has also been done in terms of area, delay and power using the 32 nm standard cell library in Synopsys Design Compiler. Table VI presents the comparative analysis for the high voltage threshold (HVT) library.

Table VI. Performance Comparison Using the 32 Nm HVT Standard Cell Library

Input data size (N)	Technique	Area (μm^2)	Delay (ns)	Power (μW)
8	Proposed Direct-I Direct-II	6802 3075 2717	6.14 7.22 5.96	280 188 131
16	Proposed Direct-I Direct-II	8607 9452 4526	7.99 14.7 9.71	470 655 285
32	Proposed Direct-I Direct-II	12242 33088 8156	11.75 29.67 17.19	830 2212 670

It can be observed that as the input data size increases, the proposed methodology for the modular multiplication is more efficient than the direct computation methods.

V. CONCLUSION

An efficient modulo multiplier based on Vedic mathmatics is proposed in this paper. A detailed design analysis is presented for mod-7 modulo multiplier for the input data sizes of 8, 16 and 32 bits. The design process can be extended for other modulo values after making the modifications suggested in Table IV. The proposed design is compared with direct computation, and the comparison results are presented in Tables V and VI. It can be inferred that the proposed design is efficient in terms of delay when the input data size is large, irrespective of the design plateforms, such as FPGA and standard cell. Furthermore, the use of different architectures of adders, including carry save adder, square root carry select adder and modified square root carry select adder, may be proposed for future work.

REFERENCES

1. S. Akhter, “VHDL implementation of fast N×N multiplier based on Vedic mathematics,” in Proc. 18th European Conference on Circuit Theory and Design, 2007, pp. 472-475.

2. S. Akhter and S. Chaturvedi, "Modified binary multiplier circuit based on Vedic mathematics," in Proc. IEEE 2019 International Conference on Signal Processing and Integrated Networks, 2019, pp. 234-237.
3. A. Omondi and B. Premkumar, Residue number systems: Theory and implementation, 1st ed. London: Imperial College Press, 2007.
4. A. A. Hiasat, "General modular adder designs for residue number system applications," IET Circuits, Devices & Systems, vol. 12, no. 4, pp. 424-431, Aug. 2018.
5. A. A. Hiasat, "High-speed and reduced-area modular adder structures for RNS," IEEE Transactions on Computers, vol. 51, no. 1, pp. 84-89, Jan. 2002.
6. S. Akhter, G. Raturi, and S. Khan, "Analysis and design of residue number system based building blocks," in Proc. IEEE 2018 International Conference on Signal Processing and Integrated Networks, 2018, pp. 441-445.
7. T. Gupta, S. Akhter, A. Srivastava, and S. Chaturvedi, "HDL implementation of five moduli residue number system," International Journal of Innovative Technology and Exploring Engineering, vol. 8, no. 9, pp. 689-693, Jul. 2019.
8. S. Akhter, S. Chaturvedi, and K. Pardhasardi, "CMOS implementation of efficient 16-bit square root carry-select adder," in Proc. IEEE 2015 International Conference on Signal Processing and Integrated Networks, 2015, pp. 891-896.
9. B. Ramkumar and H. M. Kittur, "Low-power and area-efficient carry select adder," IEEE Transactions on Very Large Scale Integration Systems, vol. 20, no. 2, pp. 371-375, Feb. 2012.
10. G. Singh, "Design of low area and low power modified 32-bit square root carry select adder," International Journal of Engineering Research and General Science, vol. 2, no. 4, pp. 422-431, Jun. 2014.
11. Y. He, C.-H. Chang, and J. Gu "An area efficient 64-bit square root carry-select adder for low power applications," in Proc. IEEE 2005 International Symposium on Circuits and Systems, 2005, pp. 4082-4085.
12. V. Kokilavani, K. Preethi, and P. Balasubramanian, "FPGA-based synthesis of high-speed hybrid carry select adders," Advances in Electronics, vol. 2015, pp. 1-13, 2015.
13. S. Akhter, V. Saini, and J. Saini, "Analysis of Vedic multiplier using various adder topologies," in Proc. IEEE 2017 International Conference on Signal Processing and Integrated Networks, 2017, pp. 173-176.

AUTHORS PROFILE

Tukur Gupta received her B.Tech. in Electronics and Communication Engineering (ECE) from the Gautam Buddha Technical University (Formerly, Uttar Pradesh Technical University, Lucknow) in 2010 and M.Tech. in Microelectronics and Embedded Technology from Jaypee Institute of Information Technology (JIIT), NOIDA in 2015. She is currently pursuing Ph.D. in VLSI Design from JIIT, NOIDA. She has 6 years of industrial and teaching experience. Her research interests include VLSI design and low-power design.

Shaheen Khan obtained his B.Tech. and M.Tech. degrees in ECE from ITM, Gurgaon, India and RV University, Udaipur, India in 2001 and 2006, respectively. He is currently pursuing his Ph.D. degree from Jamia Millia Islamia, New Delhi, India. His research area includes digital system design and signal processing. He is a Life Member of IETE.

Shamim Akhter did B.Tech. (ECE) from AMU, Aligarh (2001), M.Tech. (VLSI) from IIT Delhi (2003) and Ph.D. from JIIT NOIDA (2015). His research interest is VLSI signal processing.

Saurabh Chaturvedi obtained his B.Tech. degree in ECE from JIIT, NOIDA in 2005, M.Tech. degree in VLSI Design from the Guru Gobind Singh Indraprastha University, Delhi, India in 2008 and Ph.D. degree from the University of Johannesburg, South Africa in 2018. He is a Senior Member of IEEE.