International Journal of Scientific Research in Multidisciplinary Studies
Vol.3, Issue.7, pp.17-22, July (2017)

Research Paper
E-ISSN: 2454-9312
P-ISSN: 2454-6143

Design and Implementation of High Speed Radix-2 CSD Based Floating Point Multiplier

M.Lakshmi Kiran[1*], K.V. Ramanaiah[2]

[1*]Dept. of ECE, YSREC of Yogi Vemana University, Yogi Vemana University, Proddatur, India
[2] Dept. of ECE, YSREC of Yogi Vemana University, Yogi Vemana University, Proddatur, India

[*]*Corresponding Author: lk.*ece@yogivemanuniversity.ac.in

Available online at: www.isroset.org

Received 02nd Jun 2017, Revised 17th Jun 2017, Accepted 11th Jul 2017, Online 30th Jul 2017

***Abstract*—** Since last few decades, Analog systems or especially Digital systems are fabricated with ICs. These systems are basically having multiplier as fundamental element. Thus design of multiplier is so important that digital systems will be designed efficiently. Furthermore floating point multipliers are preferable over normal multipliers, because unlike normal multipliers floating point multipliers would support very small and even very large numbers. Earlier many researchers focused on design of floating point multipliers which results to many algorithms such as Booth algorithm, Vedic sutras and CSD algorithm. Here, modified CSD algorithm is proposed which is having lower delay i.e. higher speed in comparison with existing CSD algorithm due to the usage of pipeline concept. All the algorithms of floating point multiplier discussed here are designd using verilog HDL and targeted on Xilinx ISE 14.5 Vertex-7.

Keywords— *CSD (Canonic Signed Digit), HDL (Hardware Description Language), ISE(Integrated Synthesis Environment)*

I. INTRODUCTION

The world today is moving towards digitalization i.e. existing analog systems in all fields are upgrading to digital systems and upcoming systems are implementing with digital systems through ICs with aiming towards miniaturization in size, reduction in power and improving speed levels. For achieving these targets, designers must focus their research on efficient design of Digital multiplier. Because they are the main building blocks of any digital system, such as floating point multiplier. Also, floating point numbers supports very small numbers to a very large numbers unlike normal numbers. There are many algorithms available for implementing floating point multiplier [8-12], such as Conventional method, Vedic algorithm [13-15], CSD algorithm [1-7] and etc. Conventional method considers the process as we do on a paper. Although it is simple to design and implement but it becomes complex as size of the numbers increases and it is suffered with a larger delay and a huge resource requirement. Booth algorithm reduces number of partial products in a digital multiplier. The principle idea in this algorithm is to replace addition due to a string of ones with a subtraction at the right end and add a one before left end of string. This algorithm becomes complex as the length of the number increases. CSD algorithm is modified version of Booth algorithm, which reduces number of partial products further so that power consumption is said to be reduced. Modified CSD algorithm accelerates the speed of computation along with low power consumption. Section-I contain the introduction of floating point multiplies, Section-II contains IEEE-754 single precision based floating multiplier [9] and process of conversion between fixed point numbers and floating point numbers using Xilinx Core-gen, Section-III contains existing methods of multiplication such as Conventional method, Vedic algorithm and existing CSD algorithm, Section-IV contains Modified CSD multiplication algorithm is discussed along with fixed point multiplier through Xilinx Core-gen, Section-V contains comparison of existing and proposed methods and Section-VI concludes the research work with future directions.

II. IEEE-754 SINGLE PRECISION FLOATING POINT MULTIPLIER

IEEE has introduced 2 standard number formats of floating point numbers [7]. They are IEEE-754 single precision floating point number and IEEE-754 double precision floating point number. These 2 formats contain sign bit, exponential bits and mantissa bits. Sign bit is a single bit in both formats for discriminating numbers whether signed or unsigned i.e. 1 for signed numbers, 0 for unsigned numbers. Exponential bits are 8-bits wide for single precision and 11-bits for double precision numbers. Mantissa bits are 23-bits

wide for single precision and 52-bits for double precision. The method of floating point multiplication is decided by the way the mantissas are multiplied. The conventional way of floating point multiplication is as shown in Fig.1. A one added to each mantissa at LSB before giving to multiplication block. Sign block calculates xor of two sign bits to get sign bit for the result. Exponential block adds two exponential bits including biasing exponents before addition and un-biasing the exponent after addition. Multiplication output of 48-bits and exponential block output are given to normalization block which generates 23-bit resultant mantissa. With this final multiplication output is said to be produced. Normalization block checks various special cases involved in the FP such as not a number, infinite number, zero number, de-normal value, normal value, over flow.

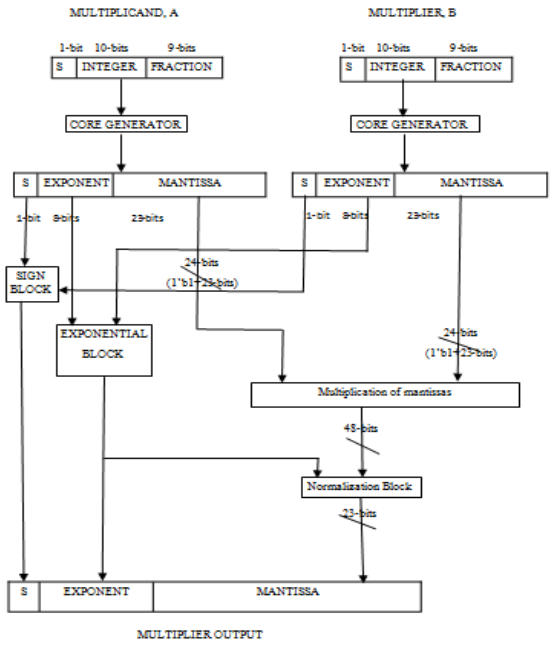


Figure 1. Block diagram for IEEE-754 single precision floating point multiplier along with core generator for fixed point number to floating point number converters.

Further, fixed point numbers multiplication can be done by using Xilinx Core generators. In Fig.1 fixed point multiplicand and multiplier numbers are given to core-gen to produce floating point numbers. Resultant multiplication output can be given to another coregent to get back fixed point number (which is not shown in Fig.1). Fixed point number contains a sign bit, variable width integer part and variable width fractional part. In this paper integer part width is taken as 10-bits and fractional part width is taken as 9-bits.

III. EXISTING MULTIPLICATION ALGORITHMS

In this section overview of existing multiplication algorithms are presented.

A. Conventional method

This is the direct method of multiplication as we do on a paper. Considering one bit at a time from LSB side of multiplier and then add shifted multiplicand to previous result if that bit is non-zero or no-operation otherwise. This process is repeated until MSB of multiplier. Here number of positions to be shifted depends upon the position of non-zero bits in the multiplier.

B. Vedic algorithm

This paper deals with a vedic sutra, Urdhva Tiryakbhyam [14] for multiplying mantissas of two floating point numbers. Urdhva Tiryakbhyam means crosswise and vertical multiplication. Each and every bit of both numbers (mantissas) are given to crosswise and vertical multiplication. Vedic multiplier is said to be having efficient area and time delay. Thus it finds great application in DSP algorithms.

In vedic algorithm, multiplication for any length of numbers can be done from multiplication of small length numbers. In single precision numbers, as mantissa is of 24 bits basic building block is of 6-bit vedic multiplication. 12-bit vedic multiplication can be done by using 6-bit vedic multiplication module, and then final 24-bit vedic multiplication for mantissas can be done by using 12-bit vedic multiplication module.

C. Existing CSD algorithm

The number of addition operations required is just one less than of the number of nonzero bits in the constant number. To reduce the number the power consumption and area, the constant number can be coded so that it contains minimum nonzero bits. This can be done by representing the number in canonic signed digit form. The characteristics of CSD representation are shown below [2].

- A CSD number doesn't contain consecutive bits are nonzero.
- The CSD number contains minimum number of nonzero bits, hence the name canonic.
- The CSD representation is unique for a given number.
- CSD numbers cover the range (-4/3,4/3), in which the values in the range [-1,1) are of great interest.
- The number of nonzero bits in the range of [-1,1) for W-bit CSD numbers is $W/3 + 1/9 + O(2^{-W})$
- Therefore CSD number contains around 33% fewer nonzero bits than normal numbers.

The algorithm for converting binary number to CSD number is presented below. The binary A is represented as $A= a_{W-1}a_{W-2} \ldots a_1a_0$ and its CSD number is $C= c_{W-1}c_{W-2} \ldots c_1c_0$.

$$a_{-1} = 0,\ y_{-1} = 0, a_W = a_{W-1}$$

for (i= 0 to W-1)
{
$Q_i = a_i$ xor a_{i-1}
$yi = y_{i-1}$ and Q_i
$ti = a_{i+1}$ and y_i
$c_i = 1- t_i - t_i$
}

Constant multiplication can be done by subtracting or adding partial products corresponding to positions of the nonzero bits in the constant multiplier. A CSD coded multiplier contains minimum number of nonzero bits. Thus it requires least number of subtraction or addition operations.

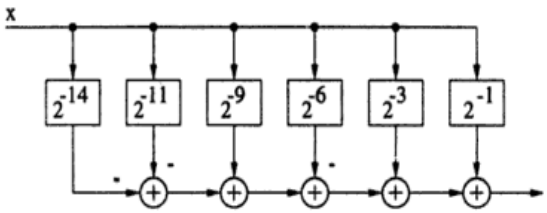


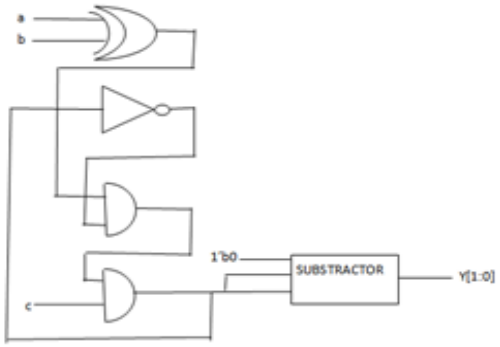
Figure .2 CSD multiplier using linear arrangement of adders [2].

The CSD multiplication of x X $0.101001\bar{1}0010\bar{1}00\bar{1}$ can be calculated as sown in Figure .2, in which partial products are accumulated linearly. But, this is not that much efficient way of implementation w.r.t. accuracy and time computation. These problems can be avoided by other arrangements such as Horner's rule based multiplication and tree-height reduction.

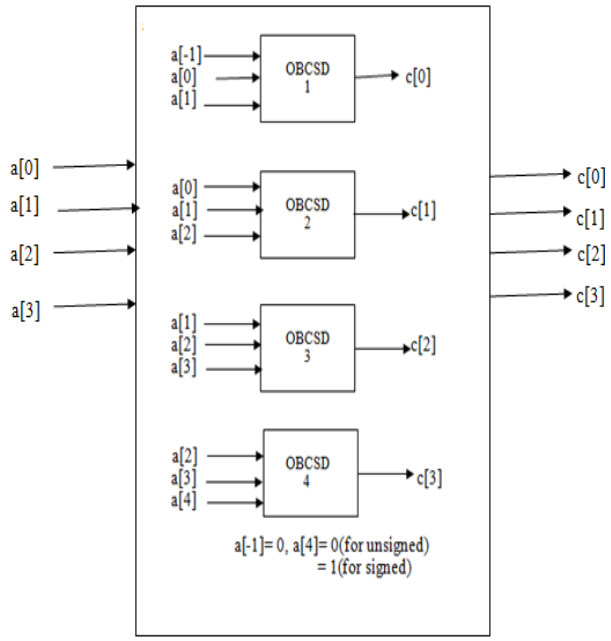
IV. PROPOSED CSD MULTIPLICATION ALGORITHM

In general, multiplication in CSD representation can be done in 2 steps, first converting the binary number of multiplier into CSD number followed by multiplication. In this paper CSD conversion process is little bit enhanced structurally as shown in Figure 3(a)&(b) for One Bit module and 4-bit module respectively.

Further, CSD multiplication using floating point numbers is also modified as shown Fig.4 unlike that of existing CSD multiplication or vedic multiplication or conventional multiplication, mantissas are not directly given to multiplication block. 23-bit of mantissa is converted into



(a)



(b)

Figure 3.(a) Logic diagram of One Bit CSD(OBCD) (b) Block diagram of 4-bit CSD conversion using OBCSD modules of (a)

CSD form with 48-bits and it is added with 2-bit of 01 at MSB which is equivalent to 1. Now it is of 50-bits supplied to Normalization block along with the output of exponential block to get 23-bit final mantissa. Sign bit and exponent bits computation remains same. With these modifications accuracy of the output is improved along with great reduction in time delay.

V. SIMULATION AND SYNTHESIS RESULTS

The Single precision floating point multiplier for the proposed CSD algorithms are synthesized using XC7VX330T device of Virtex-7 family and coded with Verilog HDL. Its performance parameters are compared with floating multiplier of conventional method, Vedic algorithm [13-15] and Existing CSD algorithm [1]. Table 1 shows power requirement, time delay, device utilization of IEEE-754 single precision floating multipliers for various algorithms of conventional method, vedic algorithm, existing CSD algorithm and proposed CSD algorithm. From this table it clear that proposed CSD algorithm is better in power consumption and time delay as compared to remaining algorithms but at the cost of device usage. Even though existing CSD algorithm and vedic algorithm uses few resources they are not useful for many applications due to their larger delay. Conventional method may have shorter delay but uses more resources and hence it is not efficient for pipelined architectures. Xilinx core-gen can be used for converting fixed point numbers to floating point numbers and vice versa. Figure 5(a) shows simulation of fixed point multiplier with proposed CSD algorithm using number converters, Figure 5(b) shows synthesis results of floating point multiplier with proposed CSD algorithm.

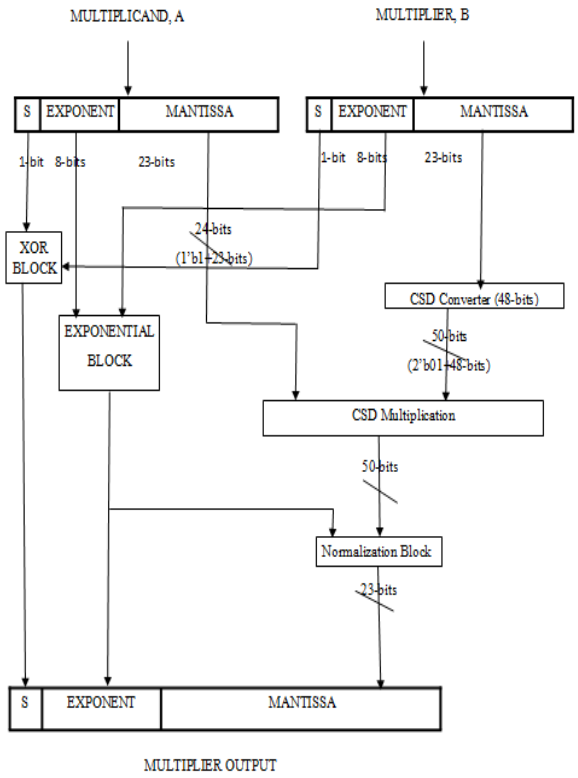
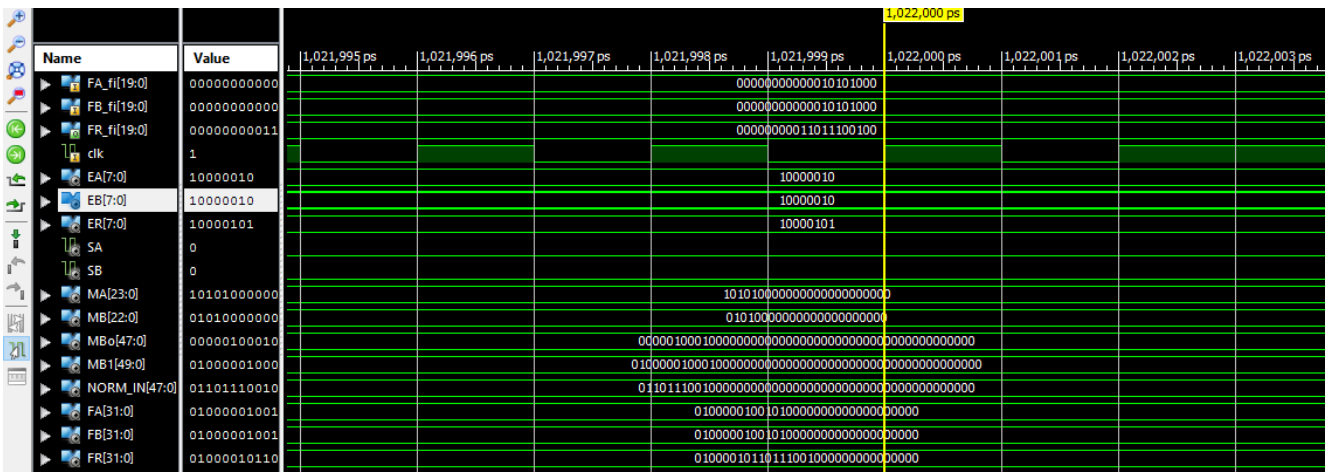


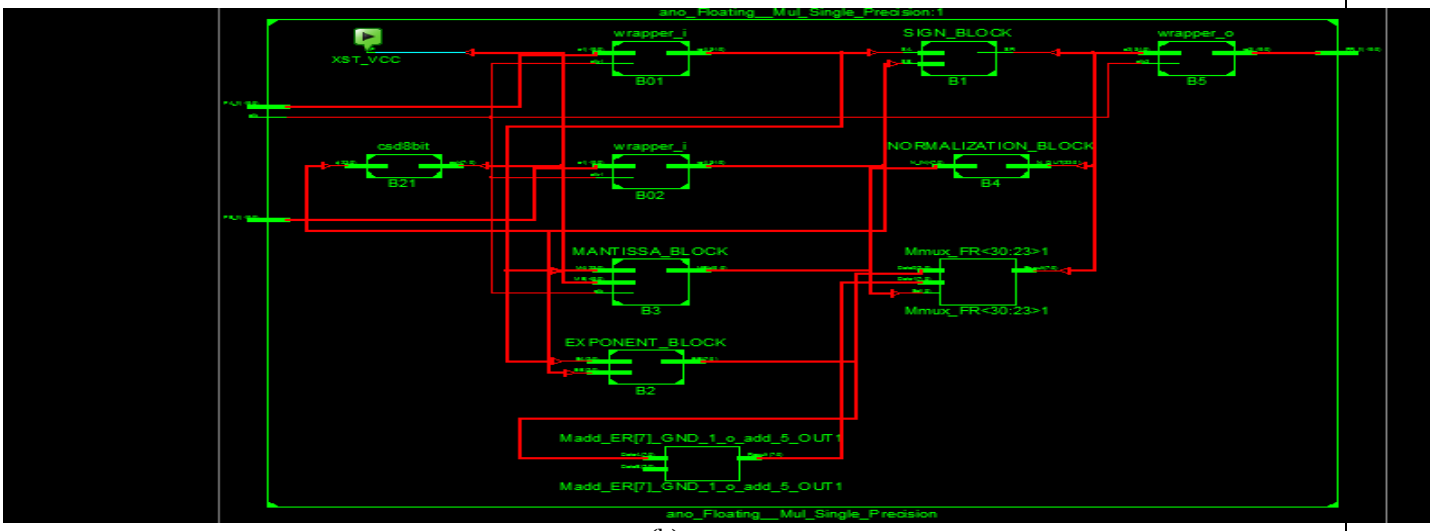
Figure 4. Block diagram of floating point multiplier using proposed CSD algorithm.

Table 1. Comparison of power requirement, time delay and device utilization for conventional method, vedic algorithm, existing CSD algorithm and proposed CSD algorithm of IEEE-754 single precision floating point multipliers.

Parameter	Conventional method	Vedic algorithm	Existing CSD algorithm	Proposed CSD algorithm
Number of Slice LUTs	1304 out of 63400 (2%)	1886 out of 204000 (0%)	904	3045 out of 298600 (1%)
Number of fully used LUT FF pairs	49 out of 1304 (4%)	0 out of 1886 (0%)	------	48 out of 3045 (1%)
Number of bonded IOBs	97 out of 210 (46%)	96 out of 600 (16%)	------	97 out of 400 (24%)
Total delay	5.117 ns	15.750 ns	61.2 ns	3.667 ns
Power requirement	42.8 mw	42.8 mw	42.8 mw	18.2 mw



(a)



(b)

Figure 5. (a) simulation results and (b) synthesis results of IEEE-754 single precision floating point multiplier of proposed CSD algorithm with fixed point number inputs and output.

VI. CONCLUSION

The IEEE-754 single precision floating multiplier with proposed CSD algorithm is synthesized on Xilinx Virtex-7 device, used Xilinx core-gen for conversion between fixed point-floating point and coded in Verilog HDL. Time delay and power requirement of proposed algorithm are better in comparison with conventional method, vedic algorithm and existing CSD algorithm but at the cost of resource utilization. Although existing CSD algorithm and vedic algorithm occupies least resources but it is slowest. Therefore proposed method can be efficiently used in high speed DSP applications.

REFERENCES

[1] Saroja.V Siddamal, R.M.Banakar and B.C.Jinaga "*Design of High-speed floating point multiplier*",in the 4th International Sympoiusm on Electronic Design, Test And Applications,China,2008.

[2] Keshab K. Parhi , "*VLSI DIGITAL SIGNAL PROCESSING SYSTEMS – DESIGN AND IMPEMENTATION*", A Wiley-Interscience Publication, **India**, pp. 505-511, 1999.

[3] Gustavo A.Ruiz, Mercedes Granda, "efficient canonic signed digit recoding", *ScienceDirect Microelectronics journal*, Issue.42, pp.1090-1097, 2011.

[4] Rui Guo and Linda Sumners DeBrunner, "A novel fast canonical-sgned-digit conversion technique for multiplication,"in *Proc. ICASS*,pp.1637-1640, *2011*.

[5] Kripasagar Venkat, "Efficient multiplication and divison using MSP 430," Texas Instruments, Dallas, Texas,SLAA329, Sep. 2006.

[6] Shoab Ahmed Khan," Multiplier-less Multiplication by Constants," in Digital Design of signal processing systems: A practical approach, John Wiley & sons Publishing Company,2010,UK,pp.222–243,2011.

[7] IEEE standards board, "IEEE standard for Binary Floating---Point Arithmetic", New York:IEEE,1985.

[8] GH. A. Aty, Aziza 1. Hussein, I. S. Ashour and M.Mona, "High-speed, Area-Efficient FPGA-Based - Floating-point Multiplier", *Proceedings ICM 2003*, Egypt, pp.274-277, 2003.

[9] H.A.H.Fahmy and M.J.Flynn, "The case for a redundant format in floating point arithmetic," *16th IEEE symposium on computer Arithmetic,2003.*

[10] Ahmet Akkas, Michael J. Schulte, "A Quadruple Precision and Dual Double Precision Floating-Point Multiplier", *Proceedings DSD 2003*, Turkey ,pp.76-81, 2003,Turkey.

[11] Himanshu Thapliyal and M.B.srinavas, "A Novel Time-Area-Power Efficient Single Precision Floating Multiplier," *Proceedings* MAPLD 2005.

[12] A G. Even and P.-M. Seidel, "A comparison of three rounding algorithms for ieee floating-point multiplication," *Computers, IEEE Transactions on*, vol. 49,issue. 7, pp. 638–650, 2000.

[13] A. Kanhe, S. Das, and A. Singh, "Design and implementation of floating point multiplier based on vedic multiplication technique," in *Communication, Information Computing Technology (ICCICT)*, pp. 1–4, 2012.

[14] A. Itawadiya, R. Mahle, V. Patel, and D. Kumar, "Design a dsp operations using vedic mathematics," in *Communications and Signal Processing (ICCSP)*, pp. 897–902,2013.

[15] M. Ramalatha, K. Dayalan, P. Dharani, and S. Priya, "*High speed energy efficient alu design using vedic multiplication techniques*," in *Advances in Computational Tools for Engineering Applications, ACTEA '09 International Conference* , pp. 600–603, *2009.*

[16] Parth Mehta, Dhanashri Gawali," *Conventional versus Vedic mathematical method for Hardware implementation of a multiplier*", 2009 International Conference on Advances in Computing, Control, and Telecommunication Technologies, India, pp. 640-642, 2009.

[17] Sumit Vaidya, Deepak Dandekar," *Delay-Power performance comparison of multipliers in VLSI circuit design*", International Journal of Computer Networks & Communications (IJCNC), Vol.2, No.4, pp.47-56, 2010.

FIRST AUTHOR PROFILE

Mr. M.L.Kiran pursed B. Tech. from GPREC, Kurnool in 2009, M.Tech. from JNTUACEA, Ananthapuramu in 2014 with specialization in Digital Electronics and Communication Systems. He is currently Pursuing Ph.D (Full Time) on VLSI & DIP from Yogi Vemana University, Proddatur since 2016. He has worked as Asst. Professor in Department of Electronics and Communication Engg. in VITS, Proddatur from 2014 to 2016. He has published 4 research papers in reputed international journals and conferences and it's also available online. His main research work focuses on VLSI, Digital Image Processing and Communication systems. He has 04 years of teaching experience.

The author has obtained Ph. D degree in the year 2009, from J.N.T.U College of Engineering Hyderabad in the Specialization of VLSI Architectures for Neural Network Based Image Compression, M. Tech degree from J.N.T.U College of Engineering, Hyderabad, in the Specialization of Digital Systems and Computer Electronics, and B.E Degree from KBNCE Gulbarga College, Gulbarga University. Presently working as Head of the Department of Electronics and Communication Engineering, YSR Engineering College of Yogi Vemana University. Presently Guiding 7 Students for Ph.D. degrees. Professional Experience in teaching field is 24 Years. Areas of research interests are VLSI, Image processing, and Neural Networks. He has published more than 50 research papers in reputed international journals and conferences and it's also available online.

