RESEARCH ARTICLE OPEN ACCESS

# Optimized Design and implementation of IEEE-754 Floating point processor

G.Srinivasulu[1], Dr.V.Thrimurthulu[2], G.Rajesh[3]
[1]II M.Tech VLSI SD, CR Engineering College, Tirupathi, Chittoor(Dist), A.P, India,
[2]Professor, Head of ECE Dept., CR Engineering College, Tirupathi, Chittoor (Dist), A.P, India,
[3]Assosciate Professor,ECE Dept., CR Engineering College, Tirupathi,Chittoor(Dist), A.P,India
[1]seenu893@gmail.com, [2]vtmurthy.v@gmail.com , [3]gundlapallerajesh@gmail.com

**Abstract**
Inpresent a days floating point arithmetic operations are very crucial in financial applications. In DSP applications floating point arithmetic implemented by using FFT algorithm. This paper presents optimized design and implementation of double precision floating point processor. This processor is dynamically configured, loaded and executed. This processor is binary compliant with the conventional microprocessor without interlocked pipelining system (MIPS). Designed the hardware to optimize the area and delay. The design is coded in verilogHDL at RTL and synthesized in virtex5 by using Xilinx ISE tool.
**Key words:** floating point; single precision; processor design

## I. INTRODUCTION

Floating-point arithmetic is mostly used in lot of areas, mainly in scientific computation; numerical processing and signal processing (like digital filters, FFT, image processing, etc.)[8] applications. Single-precision and double-precision formats represented by theIEEE-754.

By using IEEE-754 format is higher than that of fixed point representation with the same number of bits the range &precision of numbers that can be represented. Arithmetic operations Implementation for IEEE floating-point standard in hardware becomes an important part of almost all processors. Efficient implementation of floating-point arithmetic operation applications are always having for high-performance.

Due to development in VLSI technology nowadays we haveFPGA's with features of high speed, more number of embedded modules and more number of logic. These turn them suitable for implementing more difficult applications and also we can go for developed implementation of applications like floating point arithmetic. The performance of floating point arithmetic in FPGA is improved, Then FPGA is a desirable platform for scientific and real time applications [1].By embedding our floating point processor we can easily increase the speed of floating point application.

Our view is to create a generic, adaptable, embedded floating point processor, which over floating point applications will increase performance and save a significant amount of FPGA real estate when compared to implementations on current FPGAs. With this goal of flexibility in mind, this processor was designed so that it can be configured to perform several useful functions. Since multiplication and addition are two of the most widely used arithmetic operations, these operations are included in the ALU, these operations can be done in both in integer and floating-point mode.

Our processor has 512 MB of data memory,256 KB of program memory,32 number of 32 bit register file,32 bit A and B register.32 bit ALU,32 bit PC(program counter),32 bit IR(instruction register).It has two modes of operation floating point mode and normal integer mode. In floating point mode operations like adder, subtractor, multiplier and multiply-add are performed and it also handles 5 floating point exceptions. For effective implementation the ALU uses merged data path for floating point addition and multiplication and efficient algorithm for multiplier and adder design. The design is implemented in verilogHDL and synthesized for Xilinx virtex-5 device. Thedesign is synthesized using Xilinx ISE tool.

## II. SINGLE PRECISION FLOATING POINT NUMBER

Single-precision floating-point format is a computer number format that is specified in the IEEE-754-2008standard.fig:1 shows a single precision floating point format. Where sign bit denotes the sign of the number and also the sign of the mantissa as well. Exponent is an 8 bit signed integer from -128 to 127(2's Complement) and for 8-bit unsigned integer from 0 to 255 which is the accepted biased form in IEEE 754 single precision definition. In this case an exponent with value 127 represents actual zero. The true mantissa includes 23 fraction bits to the right of the binary point and an implicit leading bit (to the left of the binary point) with value 1 unless the exponent is stored with all zeros. Actually the total precision is 24 bits [6] thus only 23 fraction bits of the mantissa appear in the memory format.

Fig. 1. IEEE 754 Double precision format where the formula to calculate the real number represented

| 1 | 11 | 52 |
|---|---|---|
| s | e | m |

Using the IEEE 754 format is

$$X= -1^s \times 1.f \times 2^{e-1023}$$

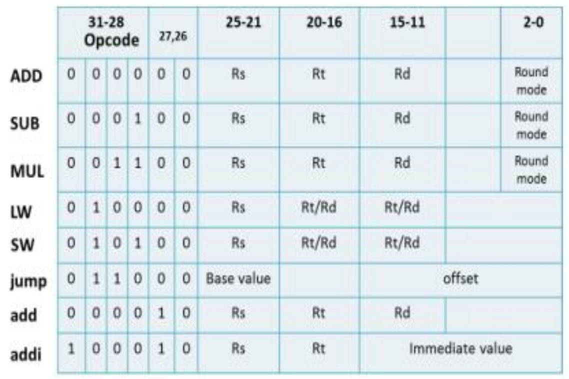
## III. PROCESSOR DESIGN

### A. Instruction set architecture

The first step in design is choosing an efficient Instruction Set Architecture (ISA) for our processor. Here we use MIPSISA (instruction set architecture).MIPS contains a load-store RISC (Reduced Instruction Set Computer) instruction set which contains three operands. Remaining of the design is divided in to two parts:

• Data path: According to the program instructions performs the data operations.

• Controller design: According to the program instructions, controller controls the data path, memory and I/O.



| | 31-28 Opcode | | | | 27,26 | | 25-21 | 20-16 | 15-11 | | 2-0 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| ADD | 0 | 0 | 0 | 0 | 0 | 0 | Rs | Rt | Rd | | Round mode |
| SUB | 0 | 0 | 0 | 1 | 0 | 0 | Rs | Rt | Rd | | Round mode |
| MUL | 0 | 0 | 1 | 1 | 0 | 0 | Rs | Rt | Rd | | Round mode |
| LW | 0 | 1 | 0 | 0 | 0 | 0 | Rs | Rt/Rd | Rt/Rd | | |
| SW | 0 | 1 | 0 | 1 | 0 | 0 | Rs | Rt/Rd | Rt/Rd | | |
| jump | 0 | 1 | 1 | 0 | 0 | 0 | Base value | | offset | | |
| add | 0 | 0 | 0 | 0 | 1 | 0 | Rs | Rt | Rd | | |
| addi | 1 | 0 | 0 | 0 | 1 | 0 | Rs | Rt | Immediate value | | |

Fig. 2. Instruction set architecture

**Instructions included are:**

| Instruction | operation |
|---|---|
| LW | load data from data memory to the file register |
| SW | The data in the file reg. is stored to data memory |
| ADD | Floating point addition is performed on two operand from the file reg. and the result is stored back to the destination reg. |
| MUL | Floating point multiplication is performed on two operand from the file reg .and the result is stored back to the destination reg. |
| SUB | Floating point subtraction is performed on two operand from the file reg.and the result is stored back to the destination reg. |
| add/sub/mul | Normal addition/subtraction/multiplication is performed on two operand from the file reg.and the result is stored back to the destination reg. |
| addi/muli/subi | Normal addition/subtraction/multiplication is performed on two operand one from the file reg other as an immediate value. and the result is stored back to the destination reg. |
| bne/beq | Branching instructions. |
| jmp | Unconditional control transfer instruction. |

Here the Instructions are 32 bit long, with a four bit opcode. And my instruction set uses 3 address instruction format. According to the Instruction format the execution of an arithmetic or logic instruction will be done i.e., whether we have an R-type or I-type instruction. The 27th bit of the instruction responsible for the modes of operation(floating point/integer mode).In case of floating point instruction the last three bits used to select the rounding mode of the floating point operation.

### B. Control unit design

The control unit of the MIPS single-cycle processor examines the instruction opcode bits [31:26] and decodes the instruction to generate 12 control signals to be used in the data path. These control signals and their operation are shown intable:1.

| Signal | Control operation |
|---|---|
| PCen | To enable the PC register. |
| P_memrd | To read the instruction from program memory |
| P_memwr | To write the instructions in to the program memory. |
| IRen | To enable the IR register |
| write | To write data in to the file register. |
| ACCen | Used to select signal for mux at the input to the B register. |

| | |
|---|---|
| ALUfz | 3 bit control signal to the ALU which select the desired operation to be performed by the ALU |
| D_memrd | To read data from data memory |
| D_memwr | To write data in to the data memory |
| memtoreg | Used to select the write data to the file register. |
| Dst_sel | Used to select the destination register address |

Here to generate the control signal the controller uses an FSM. Here the FSM consist of one initial state and 3 operating states. The states are start, fetch, decode and execute.

- START: All control signals are assigned to zero.
- FETCH: Control signals are assigned in way to fetch the instruction from program memory. The control signal active in this state are: PCen, memread, IRen
- DECODE: In this state the instruction is decoded and the data path control signals prepared for next cycle. The control signal active in: in this state are Ren, write, Bsel, ACCen, Ben.
- EXECUTE: In this state ALUget the data from the file Registerfor the desired operation and the result is send back to the destination register. The control signal active in this state are: ALUfz, d_memrd, d_memwr, memtoreg, dst_sel.

**C. Data path design**

Here the data path is based on MIPS (microprocessor without interlocked pipeline stages).It also utilizes the features of the Harvard architecture (separate memory for instruction and data).In this scheme instructions are executed in multi clock cycles. The data path consist of 512 MB of data memory,256 KB of program memory,32 number of 32 bit register file,32 bit A and B register.32 bit ALU with floating point support,32 bit PC register,32 bit IR register. To incorporate pipelining the data path is clearly divided into three section (fetch, decode, execute).And operation of each section is controlled by the control signal generated from the controller.

1) Fetch unit: The function of the instruction fetch unit is, by using the current value of the PC obtain an instruction from the instruction memory and for every next instruction the PC value will increases as shown in Figure: 3. The instruction fetch component contains the following logic elements that are implemented in Verilog: 16-bit program counter (PC) register, An adder to increment the PC by one, the instruction memory and an Instruction register.

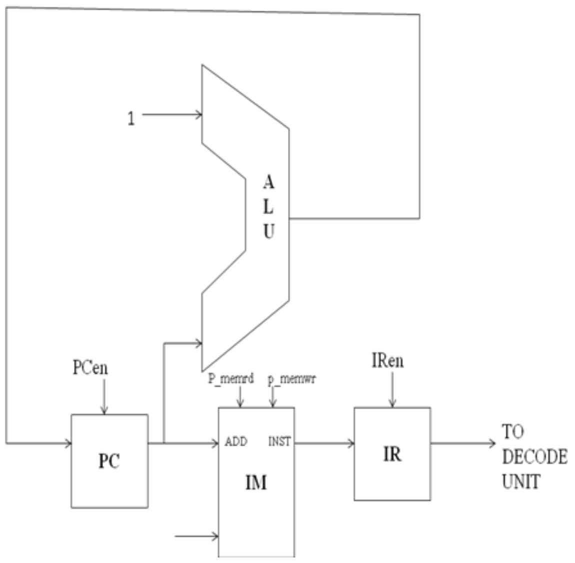


Fig3.Fetch

2) Instruction decode unit: The main function of the instruction decode unit is to decode the 32-bit instruction fetched in previous state(fetch state) to index the register file and obtain the register data Values as seen in Figure:4 . This unit also sign extends instruction bits [15 - 0] to 32-bit. The logic elements implemented in Verilog include multiplexers and a 32 bit register file,16 to 32 bit sign extender and A & B register.

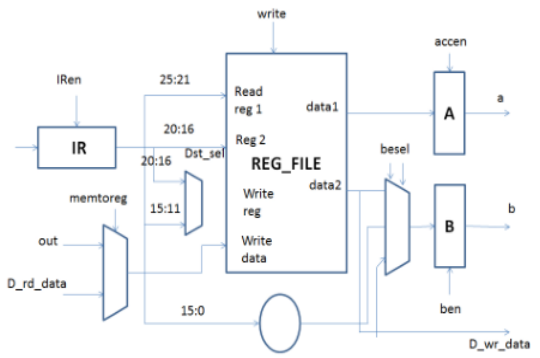


Figure4.Decode

3) Execution unit: The execution unit of the MIPS processor contains the arithmetic logic unit (ALU) which performs the operation determined by the ALUfz signal in the case of arithmetic operation. The branch address is calculated by adding the PC+1 to the sign extended immediate field shifted left 2 bits by a separate adder. And obtaining the address of data memory in case of load and store instruction. The logic elements implemented in Verilog include a multiplexer, an adder, the ALU and the ALU consist of data path for floating point arithmetic.fig:5 shows the data path for execute unit and the corresponding floating point ALU is shown in fig:6

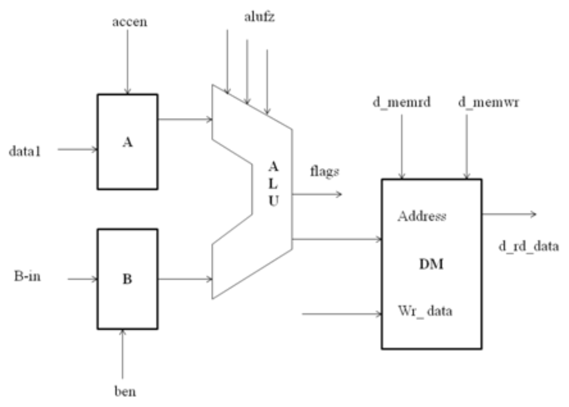


Figure5.Execute

## IV. FLOATING POINT ALU

The proposed ALU is a single precision IEEE-754 compliant integrated unit. It can handle basic floating point operations like floating point addition, subtraction, multiplication and multiply-add in floating point mode and 23 bit nor-mal addition, subtraction& multiplication in integer mode of operation. The mode of operation can be indicated by 27thbit of instruction and if the bit is set to one then floating point operation is performed. ALUfz control signal from the controller select the desired ALU operation corresponding to theinstruction. The input to the ALU is 32 bit value from A &B register. For floating point operation these are floating point numbers represented in IEEE-754 format and in the case of integer operation first 23 bit of these operand is used as input values to the ALU.

### A. Floating point addition/subtraction

As in [2] the conventional floating-point addition/subtraction algorithm consists of five stages exponent difference, pre-alignment, addition/subtraction, normalization and rounding. Given floating-point numbers and, the stages for computing are described as follows.

1) Find exponent difference. If, swap position of mantissas. Set larger exponent as tentative exponent of result.
2) Prealign mantissas by shifting smaller mantissa right by bits.
3) Add or subtract mantissas to get tentative result for mantissa.
4) Normalization. If there are leading-zeros in the tentative result, shift result left and decrement exponent by the number of leading zeros. If tentative result overflows, shift right and increment exponent by 1 bit.
5) Round mantissa result. If it overflows due to rounding, shift right and increment exponent by 1 bit.

### B. Floating point multiplication

As in[2]in standard floating point multiplication, six steps are required,

1) XOR the sign bits. XOR the sign bits of the floating point numbers to give the resulting sign bit.
2) Exponent addition. The exponents of two floating point numbers are added using a fixed point adder.
3) Mantissa de-normalization. The hidden bit 1 of 1.f is appended to the mantissas.
4) Mantissa multiplication. The mantissas are multiplied using a fixed point multiplier.
5) Normalization of mantissa and exponent.
6) Rounding. Rounding the mantissa after shifting.

From the above two algorithm we can see that floating point arithmetic require many operations. These operations contribute to area, delay and power consumption in the FPGA.So to optimize the design common data path in the Multiplier and adder are merged. Fig 6: shows the data path for a floating-point ALU. Only the main parts of the data path are shown for clarity.

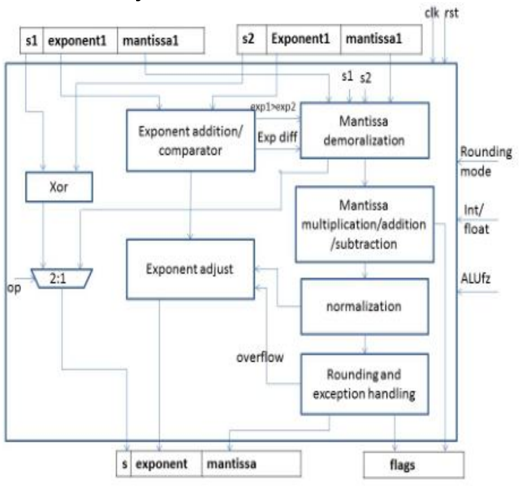


Fig6 floating point ALU

## V. ARCHITECTURE OPTIMIZATION

There are different optimization methods for floating point ALU. We can reduce the area, delay and power with the use of effective algorithm for the design. One method is to merge the common datapath.Area can be reduced by sharing resources between datapaths.fig:6 shows example of such a design. The floating point multiplier, adder and sub tractor arithmetic individually require normalization, rounding & exception handling unit [5][7]. By sharing the normalization, rounding &exception handling unit we can reduce the overall area by reducing the amount of redundant resources. Another method for optimization is use of effective algorithm for individual component design. From table: III and fig 7,8 we can infer that use of koggestone adder for mantissa addition can improve the speed. But it uses large area compared to other adders. So when performance is a main concern we can go for koggestoneadder. from table:III and fig 7,8 we can see that for area optimization Wallace tree multiplier can be used and to improve the speed we can use booth and Wallace tree multiplier. Another method for optimization is effective utilization of the resources to other application. Here 24 bit adders and multipliers for mantissa addition and multiplication is used for normal 24 bit addition and multiplication in the case of integer mode of operation.

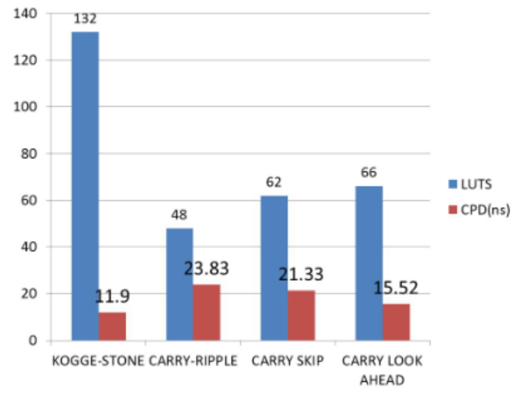


Fig 7: From synthesis of adder

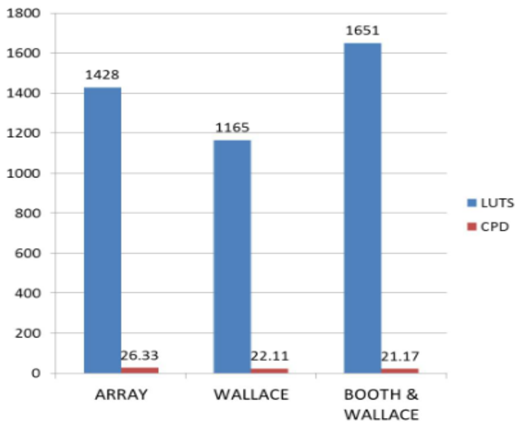


Fig 8: From synthesis of multiplier

MINIMUM DELAY

| | |
|---|---|
| multiplier | Wallace tree |
| adder | carry-ripple |

MINIMUM AREA

| | |
|---|---|
| multiplier | Booth and Wallace tree |
| adder | kogge-stone |

## VI. IMPLEMENTATION RESULT

The design is carried out in Verilog HDL, synthesized and simulated using Xilinx ISE software. The simulation is done in the Xilinx Virtex5 device [4]. Single precision floating point adder, subtractor and multiplier are realized. All exceptions and special cases are handled. The 24 bit mantissa adder in the floating point multiplier and 23 bit multiplier in the floating point multiplier module are made reconfigurable for integer mode of operation. All the operations are embedded in a single module to form a floating point ALU.With the help of thisALU single precision floating point processor is designed. Here each instruction takes3 cycle latency. Design is done in a way to accommodate a 3 stage pipelining. A simple program to add and multiply two floating point numbers is stored in program memory and corresponding floating point data is stored in Data memory.fig:9 show the ISA for the program. Andthe corresponding simulation result is shown in fig:10.Delay and area is calculated for the processor both for without optimized data path and with optimized datapath.Delay and Area estimation as per Synthesis results were 2.530ns (2.285ns logic, 0.245ns route) and 2 % of the virtex5 device area for processor with optimized data path and 3.150ns (2.830ns logic, 0.320ns route) and 2.5 % of the virtex5 device area for processor without optimized data path.

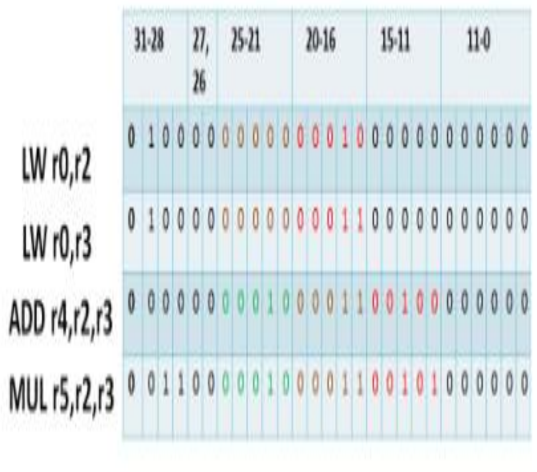


Fig.9. ISA for the program to add and Multiplication results.

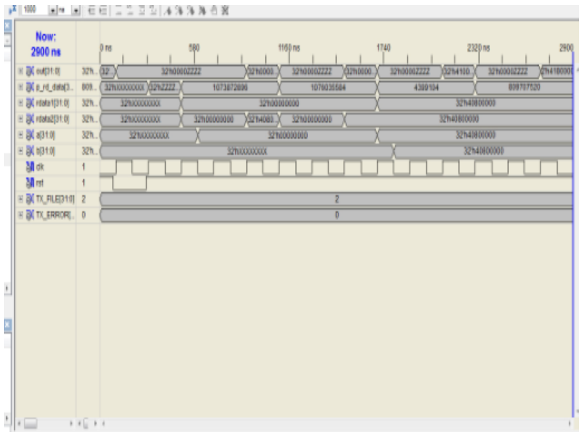


Fig. 10. Simulation result

## VII. CONCLUSION

This project mainly deals with development of a efficient Floating Point adder, subtractor and multiplier for ALU in Verilog. That ALU is used to design a double precision floating point processor. Here the processor uses MIPS and Harvard based architecture. The whole design is performed with the help of Xilinx and synthesized with Xilinx tools. The experimental Result shows that area and delay of the processor is reduced with the help of suitable hardware design for the datapath.A simple program to add and multiply two floating point numbers is stored in program memory and corresponding floating point data is stored in data memory. The simulation is done in the Xilinx virtex5 device.

By using the IEEE-754 floating point processor, we can improve the accuracy in representation of floating points which are crucial in financial applications. This project very useful in developing countries.

## REFERENCES

[1] C. H. Ho, C. W. Yu, P. H. W. Leong, W. Luk, and S. J. E. Wilton, Domainspecific hybrid FPGA: Architecture and floating point applications, inProc. Int. Conf. Field Program. Logic Appl. (FPL), 2007,pp. 196201.

[2] Yee Jern Chong and Sri Parameswaran, Configurable Multimode Embed-ded Floating-Point Units for FPGAs, IEEE Transactions 2010.

[3] A. Akkas, Dual-mode quadruple precision floating-point adder, in Proc.9th Euro micro Conf. Digit. Syst. Des. (DSD), 2006, pp. 211220.

[4] Xilinx Inc., Virtex-5 Family Overview - LX, LXT,and SXT Platforms,2007.

[5] P. C. Diniz and G. Govindu, Design of a field-programmable dual-precision floating-point arithmetic unit, in Proc. Int. Conf. Field Program.Logic Appl. (FPL), 2006, pp. 14.

[6] ANSIWEE std 754-1985, IEEE standard for binary Floating-point arithmetic, IEEE New York (1985).

[7] C.W. Yu, J. Lamoureux, S.J.E. Wilton, P.H.W. Leong and W. Luk.TheCoarse-Grained/Fine- Grained Logic Interface with Embedded Floating-Point Arithmetic Units. International Journal of Reconfiguring, 2008, Article ID 736203, 10 pages, 2008.

[8] Earl E. Swartzlander Jr.,andHani H.M. Saleh, FFT Implementation with Fused Floating-Point Operations, IEEE Transactions on computers, vol.61 no.2, february 2012.

## AUTHORS

Gollasrinivasulu received his B.Tech degree in Electronics & Communication Engineering from Shree Institute of Technical education, Tirupati (A.P), India, in the year 2012. Currently pursuing his M.Tech degree in VLSI System Design at Chadalawada Ramanamma Engineering College, Tirupati(A.P), India.. His area of research Includes Low power VLSI design.

Dr.V.Thrimurthulu M.E., Ph.D., MIETE., MISTE Professor & Head of ECE Dept. He received his Graduation in Electronics & Communication Engineering AMIETE in 1994 from Institute of Electronics & Telecommunication Engineering, New Delhi, Post Graduation in Engineering M.E specialization in Microwaves and Radar Engineering in the year Feb, 2003, from University College of Engineering, Osmania university,Hyderabad.,and his Doctorate in philosophyPh.D from central university in the year2012.He has done research work on Ad-hoc networks. He has Published 20+ papers in Various National & International Journals.

Gundlapalle Rajesh is currently working as associate professor in the department of elcetronics& communication engineering at chadalawada ramanamma engineering college, near tirupati, india. He has nearly 11 years of teaching experience. His extensive education includes b.tech. From jawaharlal technological university, hyderabad,india, plus m.e. in satyabama university, chennai, india. In addition to this, he is making research in the field of medical image processing.