ADVANCED VERIFICATION METHODOLOGY FOR COMPLEX SYSTEM ON CHIP VERIFICATION

G.Renuka[1], V.Ushashree[2] and P.Chandrasekhar Reddy[3]

[1]Department of Electronics & Communication Engg., SREC , Warangal, Telangana, India.
[2]Department of Electronics & Communication Engg., JBIET, Hyderabad, Telangana, India.
[3]Department of Electronics & Communication Engg., JNTU, Hyderabad, Telangana, India.

ABSTRACT

Verification remains the most significant challenge in getting advanced SOC devices in market. The important challenge to be solved in the Semiconductor industry is the growing complexity of SOCs. Industry experts consider that the verification effort is almost 70% to 75% of the overall design effort. Verification language cannot alone increase verification productivity but it must be accompanied by a methodology to facilitate reuse to the maximum extent under different design IP configurations. This Advanced reusable test bench development will decrease the time to market for a chip. It will help in code reuse so that the same code used in sub-block level can be used in block level and top level as well that helps in saving cost for a tape-out of a chip. This test bench development technique will help us to achieve faster time to market and will help reducing the cost for the chip up to a large extent.

KEYWORDS

Advanced verification Methodology, Verification Simulation software, Test Bench.

1. INTRODUCTION

The complexity of the chip has increased in present years and integration of more numbers of components in a single Soc makes verification of any Soc design very critical. We need proper verification methodology for any Soc or IP. The object oriented programming (OOP) concepts in verification make it easy.In this paper, the problems regarding code reusability, faster time to market, flexibility are resolved by developing the test bench environment by an advanced Verification reusable methodology. Less energy consumption, reusability, better performance, lesser simulation time were the targets achieved by using this advanced methodology.

A test bench is an environment used to verify the correctness of a model as well as of a design. It also provides various functions including applying, creating, stimulus and verifying the correctness of interfacing and responses. Developing a test bench environment is the most time-consuming task for an advanced verification team.

DOI : 10.5121/vlsic.2015.6602

Stimulus is the most prevailing technique used in functional verification today and provides ability to verify the implementation before a device is manufactured which saves development time and effort to a huge extent. To simulate the DUT under a variety of test conditions including correct and faulty test inputs. Efficiency, flexibility and reuse are the goals in developing the test bench. Attaining these goals often makes test benches more difficult to use and more complex to create. Every test bench developer should make a trade-off between the time and effort to create and use the test bench versus the potential gain from making the test bench reusable, efficient and flexible.

To improve the reusability of a test bench the main focus was kept on the design-specific information in the test bench isolation and separating the functionality of the test bench.

Abstraction of the design information to a higher level along with utilization of standard interfaces were followed to improve the efficiency and flexibility of a test bench.

2. RELATED WORK

There are some existing methodologies such as 1) open verification methodology which was jointly developed by cadence and mentor graphics. 2) Verification methodology manual which was developed by Synopsys cadence and mentor graphics. Following are the related work Corresponding to the existing methodology.

In the paper "A Low-Cost and High-Performance Embedded System Architecture and An valuation Methodology''.2014 IEEE Computer Society Annual Symposium on VLSI, the authors proposes a low-cost and high performance bus-based architecture. Furthermore, an extended evaluation methodology is created in order to examine the circuit performance automatically and accurately. [1]

''Design and Implementation of Transaction Level Processor based on UVM''. 978-1-4673-6417-1/13 /$31.00 ©2013 IEEE. In this paper, the transaction level model based on UVM is established to accelerate the SuperV_EF01DSP's software development and it plays an important role as a golden reference model in the process of SuperV_EF01 DSP's verification. In contrast with RTL model, the simulation speed of TLM is about 20-timesfaster [2]

"Practical and Efficient SOC Verification Flow by Reusing IP Test case and Test bench". 978-1-2990-3/12 /$31.00 ©2013 IEEE. In this paper, an efficient flow to reuse IP test bench and test case in SOC verification is presented. The successful applications in project have demonstrated that this reusability flow can decrease the complexity of SOC verification by fully. [3]. Reusing IP's test bench and verification IPs, and provide unified and straightforward flow to import IP test case to SOC without changing IP's test scenario.[4]', in this paper the functional coverage is divided into assertion and cover group coverage. Assertion coverage is not100% as there remain few assertions which are meant to check whether the IP gives error response in an erroneous request. But erroneous scenarios cannot be generated as there is no such functionality added in the RTL.

"Early development of UVM Based verification environment of image signal processing design using TLM reference model of RTL" an international journal of advanced computer science and Applications.vol 5,No2,2014[5]. In this Paper the author had described that TLM/ System C reference model of the design is the key component to enable the early development of Verification Environment without waiting for RTL to be available. UVM based early verification Environment is developed using Environment is developed both with Host interface and Core using Virtual Register Interface (VRI) approach. Testing of features of Verification Environment at TLM abstraction level runs faster and thus, it overall speeds up functional verification. Same environment can be reused from IP level to SOC level or from one SOC to another SOC with no/minimal change.

3. VERIFICATION METHODOLOGY

This verification methodology provides an appropriate framework to attain coverage driven verification (CDV).This CDV combines self checking test benches, automatic test generation and coverage metrics to appreciably minimizing the time spent to verify a design. The purpose of CDV is to ensure that thorough verification is done using up -front goal setting and eliminate the effort and time spent for creating hundreds of tests.

It also helps in receiving early notifications of errors and deploys error analysis to simplify debugging and runtime checking. The traditional directed testing flow is unlike than the CDV flow. Verification goals are set in CDV by using a controlled planning process. Then a test bench that generates and sends legal stimuli to the DUT is created. Coverage monitors are included to the environment for measuring progress and identify non-exercised functionality. For identification of undesired DUT behaviour, checkers are added. Simulations are carried out after both the test bench and coverage model are implemented. Using CDV, thorough verification of the design is achieved by changing the randomization seed or test bench parameters. Test constraints are included on top of the infrastructure so that the verification goals can be achieved quickly.

Some of the important constructs used in Advanced verification Methodology are as follows:

1. *Sequencer*:
   vm_sequencer#(s_item) sequencer
2. *Virtual Sequencer*
3. *TLM PORTS FIFO*:
   eg : Ports – Set of Methods Ex. Get(), Put(), Peek(), etc
4. *Call back*:
   eg:ClassDriver_callbackextendsvm_callback;
   endclass : Driver_callback
5. *Virtual Sequencer:*
6. *Configuration Database*:
   vm_config_db#(T)::set,vm_config_db #(T)::get
7. *Channel*
   vm_channel(vm_data)
8. *Atomic generator*
   vm_atomic_gen(Pack,"Atomic Pack Generator")

These important constructs helps in code reusability and reduce the time to market for a chip. Figure 1 shows the test bench architecture for this advanced verification methodology.

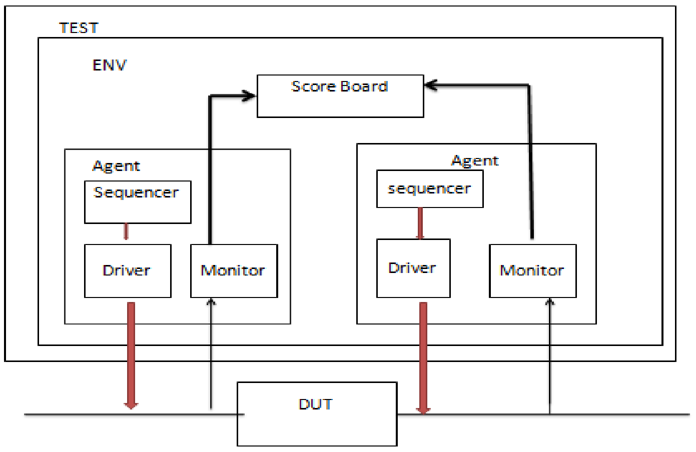

Fig. 1. Verification methodology Test Bench setup

The following are the different Verification Components used in testbench development for the SoC:

1. Data Item (Transaction)
2. Driver (BFM)
3. Sequencer
4. Monitor
5. Agent
6. Environment

1. Data Item

The input to the DUT are Data items Examples includes instructions and bus transactions .The data item's specifications derives attributes and fields of a data item. Generally, many data items are generated and sent to the Design under test by smartly randomizing data item fields using System Verilog constraints which results in more number of tests and helps in maximizing coverage.

```
class Pack extends vm _transaction
rand bit [7:0] ra;
rand bit [7:0] da;
rand bit [7:0] data[];
rand bit [7:0] length;
rand byte fc;
endclass
```

2. Driver (BFM)

A driver is an active entity which emulates logic that drives DUT. The data items are repeatedly received by the driver and samples them drives it to the DUT. (If verification environment is created in the past, driver functionality can be implemented) For example, a driver controls data bus, address bus and the read/write signal to perform a write transfer.

```
Class driver;
   //  virtual interface for driver.
   // object for collecting data from generator and transfer to sb.
   // mailbox from generator to driver.
   // mailbox from driver to scoreboard.
     event drive_done;
Function new ();
// write virtual interface,mailboxes  and event in the argument list.
// connect the arguments with the respective variables inside class driver using  "this" operator.
Endfunction
```

3. Sequencer

An advanced stimulus generator is sequencer that controls the items which are provided to driver for execution. In default case, a sequencer behaves similar to a simple stimulus generator and also returns a random data item on request from driver. The default behaviour of driver allows to include constraints in data item class for controlling the distribution of randomized values. Generators are used to randomize arrays of transactions whereas sequencer is used to capture important randomization requirements.

```
 class instruct_sequencer extends vm_sequencer #(instruction);
function new (string name, vm_componentparent);
super.new(name, parent);
`vm_update_sequence_lib_and_item(instruction)
endfunction
`vm_sequencer_utils(instruct_sequencer)
endclass
```

4. Monitor

Monitor is a passive entity which samples DUT signals without driving them. Monitors gather coverage information as well as perform checking. A monitor: Collects transactions (data items). and extracts signal information from bus and next translates the information to a transaction which can be made available for other components and to the test writer as well.

Extracts events : The monitor checks the availability of transaction (information) structures the data, and emits an event to notify the availability of other components. It also captures status information so that it can be available to other components and to the test writer. Performs checking and coverage

```
Class monitor;
// virtual interface for monitor .
// transaction object from generator.
// object to be send to scoreboard.
//mailbox for scoreboard.
Event drive_done;
Function new();
// write virtual interface,mailbox and event as argument.
//connect the arguments with the respective variables inside class driver using  "this" operator.
 //create object to be send to sb.
Endfunction
```

5. Agent

Sequencers, drivers and monitors can be used independently. For decreasing the amount of work and knowledge as per the requirement of test writer, this methodology recommends the creation of a more abstract model known as agent. Agents can verify DUT devices. Some agents also initiate transactions to the DUT, for example master or transmit agents, while other agents respond to transaction requests which are known as slave or receive agents. Agents should be configurable to be as either active or passive. Transactions are driven according to test directives by Active agents. DUT activity is monitored by passive agents.

```
Class r_write agent extends vm_agent
//declare the instances of driver, monitor and sequencer
//include a flag is_active to control the agent
//use build() phase to connect the agent subcomponent
Vitual Function void build_phase(vm_phase phase);
Super..build_phase(phase);
Monitor=r_w_monitor::type_id::create("monitor",this);
If(is_active==vm_active)
Begin
-------
------
end
endfunction:build_phase
//use connect() phase to connect the agent's sub connect
```

6. Environment

The top-level component of the Verification Component is the environment (env). It can contain one or more agents, as well as a bus monitor. The env contains config. Properties which enables in customizing the topology and behaviour to make it reusable. For example, active agents can be converted to passive agents when verification env is reused in system verification units.

```
//  Include all the transactors in the environment class.
Class  communication based SOC_env;
        // declare virtual interface for both driver and receiver.
        // declare all the mailbox used in the environment.
        // declare all the handles of the transactors.
        //event declaration.
Function new();
      // declare both the virtual interface as arguments.
// connect with the interface using 'this' operator.
endfunction
Vitual task build();
begin
// create objects of all the transactors passing the correct   arguments.
end
Endtask
Virtual task reset_dut();
begin
// here you can initialize all the values of the signals.
end
```

Endtask

Several phases of this advanced verification methodology are as follows:

- Build phase is available in vm_component. It is used to construct all sub-components right from the Test case
  ```
  Function void build_phase(vm_phase phase):
  Super.build_phase(phase);
  Driver=r_write_driver::type_id::create("driver",this);
  Endfunction: build_phase
  ```
- Connect phase is used for connecting the ports/exports of the components.
- End_ of _elaboration: This phase is used for configuring the components if required.
- Start _of_ simulation: This phase is used for configuring the components if required.
- Run: Main body of the test is executed in this phase.
  ```
  fork
     run_phase();
     begin
         reset_phase();
         configure_phase();
         main_phase();
         shutdown_phase();
  end
   join
  ```

- Extract: all the required information is gathered in this phase
- Check: checks the results of the extracted information such as unresponded requests in scoreboard, read statistics registers etc.
- Report: It is used for reporting the pass/fail status

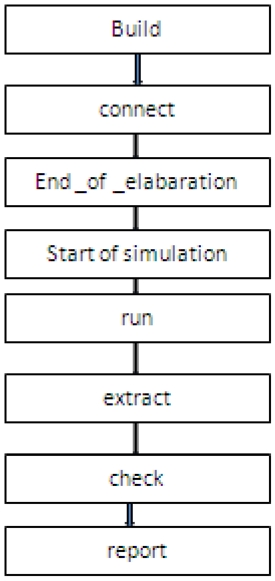

Fig. 2. Phases in advanced verification methodology

4. Results and Verification

Functional verification of communication based SOC has been carried out using advanced Verification Methodology. Verification methodology plays an important role in the functional verification of RTL design of the communication based SOC and yields the complete code coverage. Following the test Plan, the test cases are generated and then verified by developing the Verification IP. In the Sequencer, all the test cases are written as sequences using this methodology. The sequencer drives these sequences to the driver and to the Scoreboard. In the scoreboard, the comparison takes place between the actual output and the expected one. If the obtained output and the expected result matches then we conclude that the verification is completed successfully.

By using verification simulation software, the Verification of communication based SOC have been carried out and the log files for the test cases are generated with Coverage report. So the whole design is carried out using HDL and the verification is carried out by using advanced verification Methodology. The communication based SOC has been set as DUT for the functional verification and 95% code coverage has been obtained by using verification simulation software.

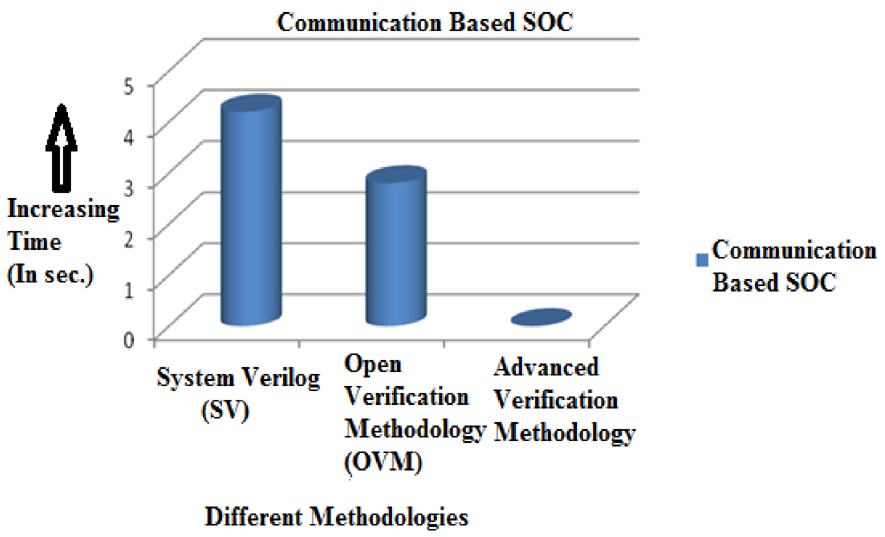

Fig. 3. Comparison graph for Different Simulation time

Fig.3. Shows the comparison between different verification methodologies i.e. System verilog, Open verification methodology and Advanced verification methodology. It is clear from the figure that Advanced verification methodology takes the minimum time for simulation with comparison to system verilog and OVM. Advanced verification methodology is more time efficient for reaching coverage goal compared to other methods.

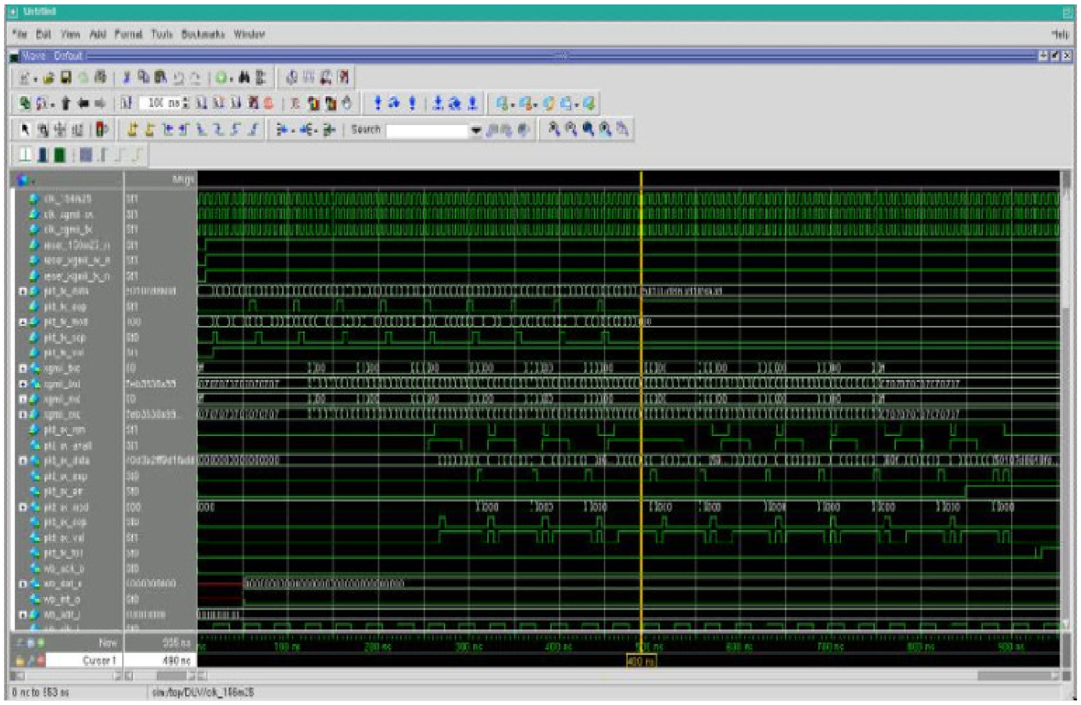

Fig. 4. Simulation Result

Fig. 4. Shows the simulation waveform of communication based SOC that has been carried out using advanced Verification Methodology. From the waveform, we can conclude proper transmission and reception of packet based data through the design under test (DUT).

TABLE I: Coverage Report

Weighted Average:				95.0%
Coverage Type ◂	**Bins**	**Hits**	**Misses**	**Coverage (%) ◂**
Branch	200	170	30	85. 00%
Total Assertion Attempted	13	13	0	100.00%
Total Assertion Failures	13	0	-	0.00%
Total Assertion Successes	13	13	0	100.00%

5. Conclusion

The specifications of Communication based SOC are verified successfully using Advance verification methodology on verification simulator and 95% code coverage has been extracted. For improvement of code coverage modification in the code has been done according to the need. The scoreboard also successfully compared the result of every transaction generated. This methodology provides the complete coverage of the RTL design so as to acquire the fault free

Protocol design of communication based SOC and that can also be implemented in real time systems. The verification flow in this research has not only reduced resources and efforts of SOC team to gather knowledge, develop test bench, test cases and debugging but also minimized the IP team's efforts as well.

REFERENCES

[1] A Low-Cost and High-Performance Embedded System Architecture and An Evaluation Methodology''.2014 IEEE Computer Society Annual Symposium on VLSI.

[2] 'Design and Implementation of Transaction Level Processor based on UVM''. 978-1-4673-6417-1/13 /$31.00 ©2013 IEEE.

[3] ''Practical and Efficient SOC Verification Flow by Reusing IP Testcase and Testbench''. 978-1-2990-3/12 /$31.00 ©2013 IEEE.

[4] ''UVM based STBUS Verification IP for verifying SoC Architectures''. 978-1-4799-4006-6/14/$31.00 ©2014 IEEE .

[5] "Early development of UVM Based verification environment of image signal processing design using TLM reference model of RTL" an international journal of advanced computer science and Applications.vol 5,No2,2014.

[6]' 'Generic System verilog UVM based reusable verification environment for efficient verification of Image signal processing IP/SOCs" an international journal of VLSI design &communication systems(VLSICS) vol3.No 6,Dec 2012.

[7]' 'VMM BASED CONSTRAINED RANDOM VERIFICATION OF AN SOC BLOCK'' an International Journal of Advances in Engineering & Technology, Sept 2012. IJAET ISSN: 2231-1963, Vol. 4, Issue 2, pp. 167-172.

[8] ''A Runtime Verification Solution for the Functional Correctness of SoCs", Rawan Abdel-Khalek and Valeria Bertacco,Department of Computer Science and Engineering, University of Michigan.ISSN NO: 978-1-4244-6471,2010 IEEE.

[9] "UVM Based Testbench Architecture for Unit Verification" 2014 Argentine School of Micro-Nanoelectronics, Technology and Applications, ISBN: 978-987-1907-86-1, IEEE Catalog Number CFP1454E-CDR

[10] "UVM-based Verification of Smart-Sensor Systems", 2012 International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design, (SMACD), 978-1-4673-0686-7/12/$31.00 ©2012 IEEE

[11] "An efficient method for using transaction level assertions in a class based verification envioremnt", 2011 International Symposium on Electronic System Design.

[12] "Transparent Security-Sensitive Process Protection via VMM-Based Process", 2013 IEEE 37th Annual Computer Software and Applications Conference Workshops. 978-0-7695-4987-3/13 $26.00 © 2013 IEEE ,DOI 10.1109/COMPSACW.2013.38.

[13] "Design of Information Flow in Collaborative-VMM" 978-1-4673-5000-6/13/$31.00 ©2013 IEEE.

[14] “Efficient Online RTL Debugging Methodology for Logic Emulation Systems “,DOI: 10.1109/VLSID.2012.87 Publication Year: 2012 , Page(s): 298 - 303 Cited by: Papers (2) IEEE Conference Publications .

[15] ” How to automate millions lines of top-level UVM testbench and handle huge register classes“,SoC Design Conference (ISOCC), 2012 Internation,DOI: 10.1109/ISOCC.2012.6407127 Publication Year: 2012 , Page(s): 405 - 407 .IEEE Conference Publications .

[16] “A reconfigurable and scalable verification environment for NoC design”, 2013 Conference on DOI: 10.1109/CoNMedia.2013.6708540 Publication Year: 2013 , Page(s): 1 - 4 IEEE Conference Publications .

[17] ” A Layered Malware Detection Model Using VMM “,25-27 June 2012 Page(s):1259 - 1264 ,Print ISBN:978-1-4673-2172-3 INSPEC Accession Number:12980219 DOI:10.1109/TrustCom.2012.35 ,Publisher:IEEE .

[18]”Advanced Testbench Design using Reusable Verification Component and OVM”an International Journal of Computer Applications © 2013 by IJCA Journal ,volume 73-number 15 year of Publication 10.5120/12820-027

[19] ”Bryan Ramirez, Michael Horn “Parameters and OVM – Can’t They Just Get Along?" Proceedings of Design and Verification Conference & Exhibition (DVCon '11), 2011

[20]” Development of JTAG Verification IP in UVM Methodology”,IJSRD - International Journal for Scientific Research & Development| Vol. 1, Issue 8, 2013 | ISSN (online): 2321-0613.

AUTHORS

G.Renuka obtained her M.TECH degree from KITS, Warangal and pursing Ph.D. in Verification Methodology for SOC from Jawaharlal Nehru Technological University, Hyderabad. She has 10 years of teaching experience.

Usha Shree is working as Professor of ECE Department, DEAN (Academics) & Director-IQAC at J.B.Institute of Engineering & Technology(AUTONOMOUS), Hyderabad. She has an experience of 16 years in teaching and research put together. She is alumni of Jawaharlal Nehru Technological University, Anantapur, A.P, India. She obtained PhD in area of MEMS sensors and Embedded Systems. She is versatile in multidisciplinary specializations in allied branches. Her laurels include more than 50 publications at National and International reputed conferences and journals.

P.Chandra Sekhar Reddy is working as Professor& Student activity co-coordinator in ECE Department at Jawaharlal Nehru Technological University, Hyderabad, India. He has an experience of 27 years in teaching and research put together. He is alumni of Jawaharlal Nehru Technological University, Anantapur, A .P , India He is multitalented in multidisciplinary specializations in allied branches. He has guided 12 Research scholars of JNTUH and JNTU Anantapur, INDIA.