Design and Verification of UART IP Core using VMM

T. Krishna Kathik , T. Praveen Blessington, Fazal. Noor Basha, ALGN. Aditya, S. R. Sastry Kalavakolanu

Abstract:- In the earlier era of electronics the UART (Universal asynchronous receiver/transmitter) played a major role in data transmission. This UART IP CORE provides serial communication capabilities,which allow communication with modems or other external devices. Thiscore is designed to be maximally compatible with industry standard designs[4]. Thekey features of this design are WISHBONE INTERFACE WITH 8-BIT OR 32-BIT selectable data bus modes. Debug interface in 32-bit data bus mode. Registerlevel and functionalcompatibility. FIFO operation. The design is verified using VMM based on system verilog. The test bench is written with regression test cases in order to acquire maximum functional coverage.

Keywords:- UART,VMM,FIFO,WISHBONE INTERFACE.

I. INTRODUCTION

The data transmission takesplace in between the chips,inside the chips and inbetween the systems also.As it is asynchronous clock there willbe no methodology to establish the clock distribution techniques. There are three of modes in UART. They are

1)HALFDUPLEX MODE:In the half duplex mode either transmission or reception takes place at a time.

2) FULLDUPLEX MODE:In the full duplex mode both transmission and reception takes place at time.

3)LOOP BACK MODE: It is used for testing purpose. We will be connection the transmitter and receiver of same UART this helps to check the accuracy of it.

Manuscript received on April 26, 2012.

T. Krishna Karthik,is currently pursuing M.Tech in the area of VLSI in K L University, Vijayawada, AP, India.
(E-mail: mail4krishnakarthik@gmail.com).

T. Praveen Blessington, Presently working as an Associate Professor & research scholar in Department of ECE, KL University, Guntur, Andhra Pradesh, India. (E-mail: praveentblessington@kluniversity.in).

Dr. Fazal Noorbasha, Presently working as an Assistant Professor, Department of Electronics and Communication Engineering, KL University, Guntur, Andhra Pradesh, India.
(E-mail:fazalnoorbasha@kluniversity.in).

A.L.G.N.ADITYA is currently pursuing M.Tech in the area of VLSI in K L University, Vijayawada, AP, India.
(E-Mail: adityasind@gmail.com).

S. R Sastry Kalavakolanu Presently he is pursuing M.Tech VLSI Design in KL University. (E-Mail: sastryece52@gmail.com).

II. STRUCTURE OF THE PACKET

It is 9-bit data bus mode. The start bit is a active low signal and the stop bit is active high signal. The rest of six bits are of data bits and a parity bit .the parallel data from CPU is converted to serial data by UART and then transmission takes place. Communication between two or more UART'S is based on asynchronous serial mode of transmission[1]. Hand shaking between the UART's is done using the synchronizing bits.Each character is sent as a start bit,a configurable number of data bits,an optional parity bit and one or more stop bits.

START	0	1	2	3	4	5	6	7	PARITY	STOP

The architecture of UART consists of

1) WISHBONEINTERFACE:the wishbone interface is the standard computer bus interface that allows to communicate between the integrated circuits. The wishbone bus permits 8-bit and 32-bit data transfer.

PORT	WIDTH	DIRECTION
CLK	1	INPUT
WB_RST_I	1	INPUT
WB_ADDR_I	5or3	INPUT
WB_SEL_I	4	INPUT
WB_DATA_I	32or8	INPUT
WB_WE_I	1	INPUT
WB_STB_I	1	INPUT
WB_CYC_I	1	INPUT
WB_DATA_O	32 or 8	OUTPUT
WB_ACK_O	1	OUTPUT

Table.1 wishbone interface signals

2)INTERRUPT REGISTERS:This register is used to enable and identify the interrupts.There are two types of interrupt registers. They are 1)Interrupt Enable Register. and 2)Interrupt Identification Register. The interrupt enable register enables and disables with interrupt generation by the UART. It is of 8-bit width. Theinterrupt identification register enables the programmer to retrieve the current highest priority pending interrupt.BIT-0 indicates that an interrupt is pending when it's logic'0'.when it is'1' the interrupt is not in pending. The width of the register is 8-bit.

2)CONTROL REGISTERS: There are two

Controlregisters. They are1)FIFO control register. and 2)LINE control register.

The FIFO control register allows selection of the FIFO trigger level. The FIFO register is of 8-bit data width. The 0th bit should be always 0.The line control register allows the specification of the format of the asynchronous data communication used.A bit in the register also allows access to the divisor latches,which define the baud rate. Reading from the register is allowed to check the current setting of the communication.

3) BAUDGENERATOR:The baud generator is responsible for generating a periodic baud pulse based on the divisor latch value which determines the baud rate for the serial transmission.This periodic pulse is used by transmitter and receiver to generate sampling pulses for sampling both received data and data to be transmitted. One baud out occurs for sixteen clock cycles. For sixteen clock cycles one bit data will be sent.
There are two debug registers they work in32-bit data bus mode. It has 5-bit address mode. It is read only and is provided for debugging purpose for chip testing. Each has a 256-byte FIFO to buffer data flow[3]. The use of FIFO buffers increases the overall transmission rate by allowing slower processors to respond, and reducing the amount of time wasted context switching. Besides data transfer, they also facilitate start/stop framing bits, check various parity options, as well as detect transmission errors[7].
4) DIVISOR LATCHES: The divisor latches can be accessedby setting the 7th bit of LCR to ‘1’.Restore the bit to zero after setting the divisor latches inorder to restore access to the other registers that occupy the same address. The two bytes form one sixteen bit register,which is internally accessed as a single number. In order to have normal operation two bytes are driven to zero on reset. The reset disables all serial I/Ooperations in order to ensure explicit setup of the register in the software. The value set should be equal to (system clock speed) /16*desired baud rate. The internal counter starts to work when the LSB of DL is return,so when setting the divisor,write the MSB first thenLSB last.

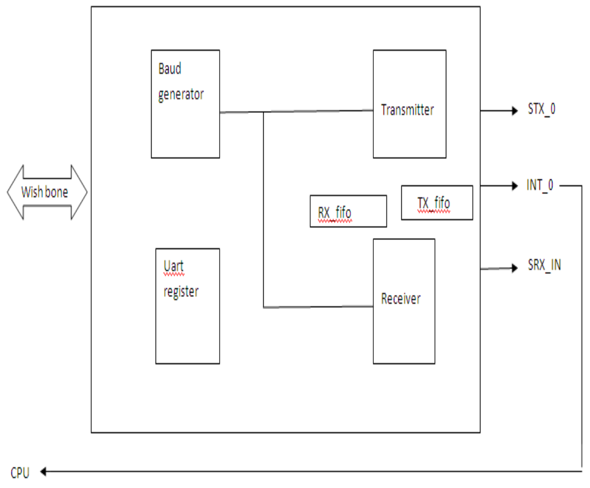


Fig.1:Block Diagram of UART IP Core

III. VERIFICATIONMETHODOLOGY

Verification is generally viewed as a fundamentally different activity from design.This narrowly focused language for verification and to the bifurcation.The VMM(verification methodology manual) that deals with System verilog it includes design Testbench,andassertionConstructsin a single language is that the test bench has easy access to all parts of environment.The value of an HVL is its ability to create high level,flexible tests,not its loop constructs or declaration style.

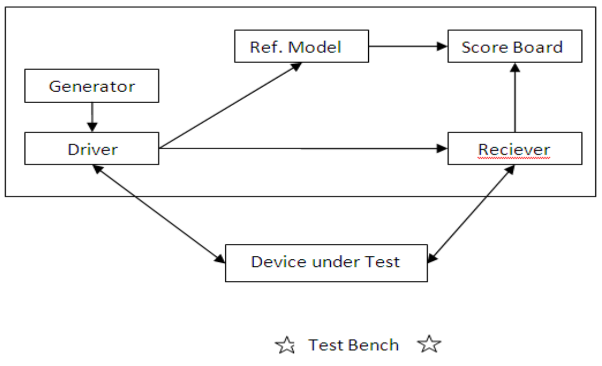


Fig.2:TEST BENCH MODULE

The system verilog test bench environmentconsists of

1) *GENERATOR*:The generator generates sequence of inputs and they are fed to thedriver. Generator is a class in which input data is randomized and driven to driver1 and driver2 through mailbox.

```
if(!data2rand.randomize())
............................
............................
        gen2drv.put(data2send);
        gen2rec.put(data2send);
```

2) *DRIVER:*Driver Drives the signals which received through the mailbox into the pins of DUT.

- Virtual Interface is provided as there is a connection between Static and dynamic components
- Methods are provided for write and read operation
- virtual task write(input [2:0] addr, input [7:0] data);
- Virtual task read(input [2:0] addr, output [7:0] read_data);

A. Driver Configuration:

Divisor latch (MSB) reg. is programmed first and the LSB reg.

```
 virtual task DIVISOR();
 write(3'b011,  8'b10000011); // LCR
write(3'b001,  data2duv.divider2);
// divisor -- MSB
write(3'b000,  data2duv.divider1);
 // divisor -- LSB
 write(3'b011,  8'b00000011); // LCR
endtask
```

Programming IER, FCR, LCR

```
virtual task CONFIG();
 write(3'b001,  data2duv.IER);  // IE
 write(3'b010,  data2duv.FCR);
     // FCR
  write(3'b011,
     data2duv.LCR); // LCR
 endtask
```

Different test cases are written to program the above registers for different scenerios

FILLING TXFIFO

TXDATA(data2duv.wb1_dat_o,data2duv.no_of_txdata1)

Where wb1_dat_o is the data to be written no_of_txdata1 is number of data to be written

3)DUV:The device that is to be verified is called DUV. The DUV is drivenwith a group of inputs and the output is compared with the reference model in the score board. The functionality is verified using functionalcoverage.

4)RECEIVER:The receiver takes the information from theDUV and sends it to the score board.

Receiver configuration Process

1. Reset the DUT2.

2.BAUD rate Generator

LCR reg. 7th bit is programmed with 1

Divisor latch (MSB) reg. is programmed first and the LSB reg.

```
 virtual task DIVISOR();
   write(3'b011,  8'b10000011); // LCR
   write(3'b001,  data2rec.divider2); // divisor -- MSB
   write(3'b000,  data2rec.divider1); // divisor -- LSB
   write(3'b011,  8'b00000011); // LCR
endtask
```

NOTE: Even though the divisor latch values are randomized (constraint div_lat1 {divider1 inside {[2:6]};}), both receiver and driver is having the same values.

Programming IER, FCR, LCR

```
virtual task CONFIG();
   write(3'b001,  data2duv.IER);  // IE
   write(3'b010,  data2duv.FCR);  // FCR
   write(3'b011,  data2duv.LCR); // LCR
  endtask
```

Reading the Receiver FIFO

```
@(rc_if.rcv_cb.int_i2);
    $display (" UART2: Receiver Interrupt has come", $time);
    for(int i = 0; i< data2rec.no_of_txdata1; i++)
      begin
        read(3'b000, read1_data);
        $display("UART2: Data Read from receiver FIFO is = %h ", read1_data, $time);
        data2sb.red_data[i] = read1_data;
      end
        rc2sb1.put(data2sb);
        read(3'b101, read1_data); // LSR reading to about the status of reception
        $display("UART2: Data Read from LSR  is = %h ", read1_data, $time);
    $display("data reading is finished", $time);
```

SCORE BOARD:In Scoreboard comparing the inputs of the UART 1driver which are coming through mailbox with output of UART2 receiver.

In Scoreboard comparing the inputs of the UART 2driver which are coming through mailbox with output of UART1 receiver.

- Test cases define in program block.
- In this we import the package, build the environment, and run this environment.
- The logic inputs are randomized usingextended class of transaction.
- The extended classes are accessed using call-back.
- Different testcases have been created for verifying half duplex and full duplex modes.
- The number of transactions which are to be performed are specified and can be easily changed.

COVERAGE REPORT:

Functional Coverage:

- This measures how much of the design specification has been exercised.
- In our verification % specification has been exercised.

Code Coverage:

- This measures how much of the code has been executed.
- In this verification 100% of the code has been exercised.

FALF DUPLEX MODE VERIFICATION

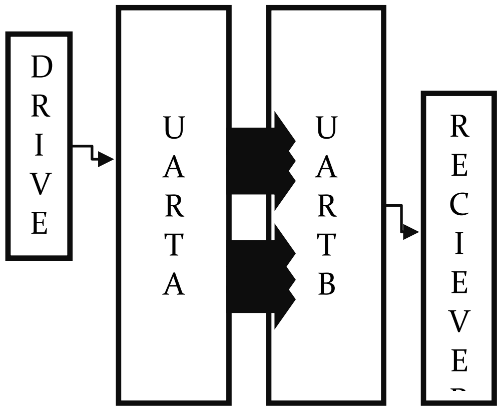


Fig.3:Half Duplex mode

1 Generator & 1 scoreboard.
Use 1 driver & 1 receiver.
Send Parallel data into UART A
UART A conv parallel to serial
UART B conv serial to parallel
Received parallel data.
Both data IN & OUT are compared & verified
FULL DUPLEX MODE VERIFICATION

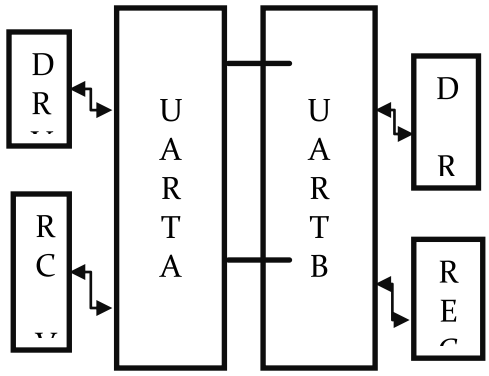


Fig.4:Full Duplex mode

- Uses 2 driver & receiver
- Send Parallel data into UART A
- UART B conv parallel to serial
- UART B conv serial to parallel
- Received parallel data.

- Both data IN & OUT are compared

IntheTestCase :

```
       class   uart_bb_trans_tc6   extends   uart_bb_trans;

        constraint   c7   {   fullduplex   ==   0;   }   //
            ...........................
        endclass
```

<u>Intherecceiverfile,</u>

```
 if(data2rec.fullduplex)

TXDATA(data2rec.wb2_dat_o,data2rec.no_of_txdata2)
;
```

<u>IntheDriverfile,</u>

```
 if(data2duv.fullduplex)
        rcv();
```

LOOP BACK MODE VERIFICATION:

Loop back mode -- contd.,

In the rtl-top level module,

```
 `ifdef LOOPBACK
       .stx_pad_o    (stx_loopback),
       .srx_pad_i    (stx_loopback),
        `else
       .stx_pad_o    ( stx_pad_12),
       .srx_pad_i    ( srx_pad_21),
`endif
```

In the Test case,

```
 constraint c8 { loopback == 1; } // To check the design in
       the Loopback
```

DIFFERENT SCENERIOUS OF TESTING

//TESTCASE 1 -- Basic Transaction with 14 datas and no parity and 1 stopbit, no stick and no break , 8 bits in the character

```
 class uart_bb_trans_tc1 extends uart_bb_trans;
  constraint c2 { FCR == 128; }
  constraint c3 { LCR == 3; }
  constraint c4 { IER == 0; }
  constraint c5 { no_of_txdata1 == 4; }
 endclass
```

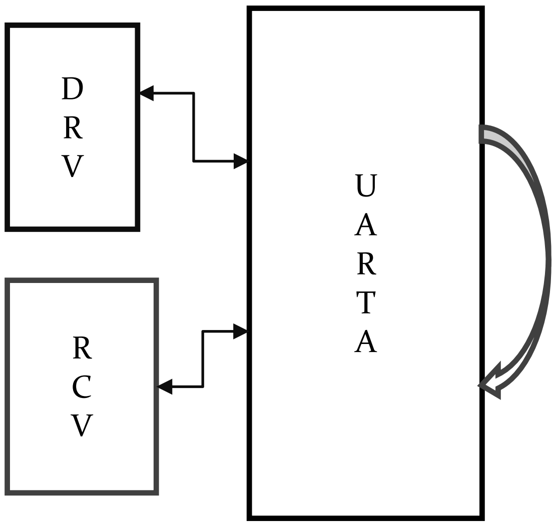


Fig.5:Loop Back Mode

//TESTCASE 2 -- changing the number of bits in the character among 5,6,7,8

```
 class uart_bb_trans_tc2 extends uart_bb_trans;
  constraint c2 { FCR == 128; }
  constraint c3 { LCR inside {[0:3]}; } // No. of bits in the
       charater is 5,6,7,8
  constraint c4 { IER == 0; }
  constraint c5 { no_of_txdata1 == 4; }
endclass.
```

IV. RESULTS:

The design has been done in verilog and verified using system verilog test bench.code coverage is done using QUESTAMODEL SIM

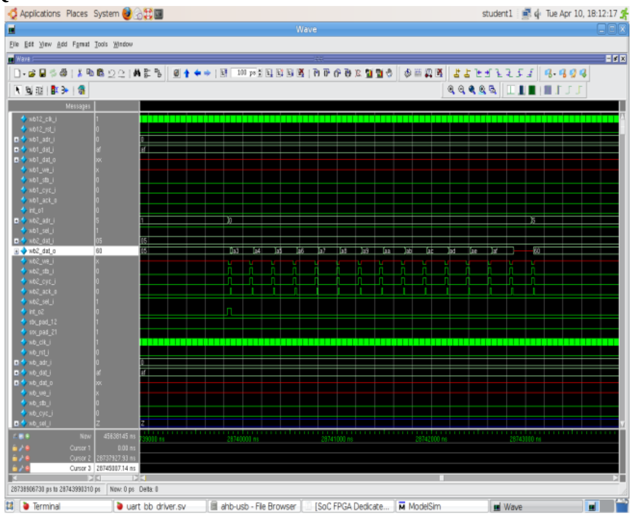
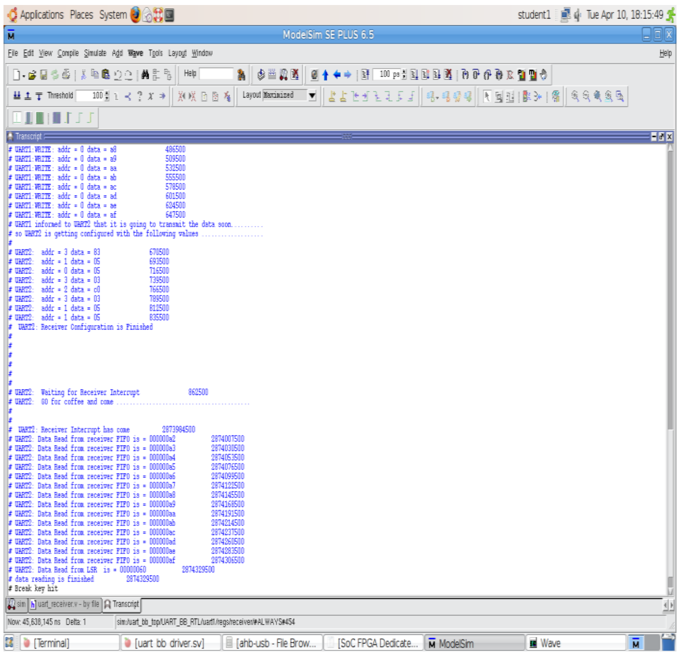


Fig .6:SIMULATED WAVE FORM



```
# UART1:WRITE: addr = 0 data = a8           486500
# UART1:WRITE: addr = 0 data = a9           509500
# UART1:WRITE: addr = 0 data = aa           532500
# UART1:WRITE: addr = 0 data = ab           555500
# UART1:WRITE: addr = 0 data = ac           578500
# UART1:WRITE: addr = 0 data = ad           601500
# UART1:WRITE: addr = 0 data = ae           624500
# UART1:WRITE: addr = 0 data = af           647500
# UART1 informed to UART2 that it is going to transmit the data soon..........
# so UART2 is getting configured with the following values ..................
#
# UART2:  addr = 3 data = 83           670500
# UART2:  addr = 1 data = 05           693500
# UART2:  addr = 0 data = 05           716500
# UART2:  addr = 3 data = 03           739500
# UART2:  addr = 2 data = c0           766500
# UART2:  addr = 3 data = 03           789500
# UART2:  addr = 1 data = 05           812500
# UART2:  addr = 1 data = 05           835500
#  UART2: Receiver Configuration is Finished
#
#
#
#
#
#
# UART2:  Waiting for Receiver Interrupt            862500
# UART2:  GO for coffee and come ..........................................
#
#
#  UART2: Receiver Interrupt has come        2873984500
# UART2: Data Read from receiver FIFO is = 000000a2        2874007500
# UART2: Data Read from receiver FIFO is = 000000a3        2874030500
# UART2: Data Read from receiver FIFO is = 000000a4        2874053500
# UART2: Data Read from receiver FIFO is = 000000a5        2874076500
# UART2: Data Read from receiver FIFO is = 000000a6        2874099500
# UART2: Data Read from receiver FIFO is = 000000a7        2874122500
# UART2: Data Read from receiver FIFO is = 000000a8        2874145500
# UART2: Data Read from receiver FIFO is = 000000a9        2874168500
# UART2: Data Read from receiver FIFO is = 000000aa        2874191500
# UART2: Data Read from receiver FIFO is = 000000ab        2874214500
# UART2: Data Read from receiver FIFO is = 000000ac        2874237500
# UART2: Data Read from receiver FIFO is = 000000ad        2874260500
# UART2: Data Read from receiver FIFO is = 000000ae        2874283500
# UART2: Data Read from receiver FIFO is = 000000af        2874306500
# UART2: Data Read from LSR  is = 00000060        2874329500
# data reading is finished        2874329500
# Break key hit
```

Now: 45,638,145 ns   Delta: 1   sim:/uart_bb_top/UART_BB_RTL/uart1/regs/receiver/#ALWAYS#454

Fig.7: TRANSCRIPT OF UART

V.CONCLUSION

The UART IP core working has been verified for different modes of operations i.e. HALF DUPLEX MODE , FULL DUPLEX MODE and LOOPBACK MODE.

The coverage of all the functions achieved is 95%. The corner cases have been verified and the registers have been checked forfunctionality. The UART transmission from serial to parallel and vice-versa is attend and has been verified.

REFERENCES

1. I.E.Sutherland'Micropiplines'*CommunicationACM,* June 1989, Vol. 32(6), pp. 720 -738.
2. V.N. Yarmolik, *Fault Diugnosis of Digital Circuits,* JohnWiley & Sons, 1990.
3. PCI6550DUniversalAsynchronousReceiver/Transmitter withFIFOs",NationalSemiconductor Application Note, June 1995.
4. M.S.Harvey,"GenericUARTManual"SiliconValley. December 1999.
5. PCI6550DUniversalAsynchronousReceiver/Transmitter withFIFOs",NationalSemiconductor Application Note, June 1995
6. Martin S. Michael, "A Comparison of the INS8250,NS16450 and NS16550AF Series ofUARTs"National Seiniconductor Application Note 493, April1989
7. W.Elmenreicb,M.Delvai,TimeTriggeredCommunication with UARTs. InProceedings ofthe 4'h IEEE International Workshop on FactoryCommunication Systems, Aug. 2002.

AUTHORS PROFILE

T. Praveen Blessington, Presently working as an Associate Professor & research scholar in Department of ECE, KL University, Guntur, Andhra Pradesh, India, where he has been engaged in teaching and research in VLSI & embedded designs. He is a member of VLSI Research Group, Department Curriculum Committee (DCC) Member. His interest is research and development in SOC, NOC Architectures, Low-Power VLSI & Embedded Systems. He has published and presented various International and National reputed journals and conferences. He is a life member of IETE, ISTE and SCIEI.
E-Mail: praveentblessington@kluniversity.in

Dr. Fazal Noorbasha, Presently working as an Assistant Professor, Department of Electronics and Communication Engineering, KL University, Guntur, Andhra Pradesh, India, where he has been engaged in teaching and research, VLSI Research Group Head, Department Curriculum Committee (DCC) Member. His interest of research and development is Low-power VLSI, High-speed CMOS VLSI SoC, Memory Processors LSI's, Digital Image Processing, Embedded Systems and Nanotechnology. He has published and presented over 35 Science and Technical papers in various International and National reputed journals and conferences. He is a Scientific and Technical Committee & Editorial Review Board Member in Engineering and Applied Sciences of World Academy of Science Engineering and Technology (WASET), Advisory Board Member of International Journal of Advances Engineering & Technology (IJAET), Life Member of Indian Society for Technical Education (ISTE-India), Member of International Association of Engineers (IAENG-China) and Senior Member of International Association of Computer Science and Information Technology (IACSIT-Singapore).
E-Mail: fazalnoorbasha@kluniversity.in

T. Krishna Karthik, :was born in Gudivada, Krishna(dist),AP, India. He received B.Tech. in Electronics & Communication Engineering from GEC,AP. India ,M.Tech from KL University, Vijayawada, AP, India. He has undergone 2 international conferences and 2 publishmentin IEEE. Email: mail4krishnakarthik@gmail.com

A. L. G. N. Aditya, was born in 31st Dec 1986He received B.Tech in Electronics and Communication Engineering from JNTU, Kakinada, AP. He is currently pursuing M.Tech in the area of VLSI in K L University, Vijayawada, AP, India. He is a coordinator of International Students Research Forum. His research interests include Analog VLSI Design, Neural networks, and biomedical electronics. He has undergone 8 international conferences and 1 publishment in IEEEE-mail: adityasind@gmail.com

S. R. Sastry Kalavakolanu, was born in A.P,India. He received the B.Tech degree in Electronics & communications Engineering from Jawaharlal Nehru Technological University in 2010. Presently he is pursuing M.Tech VLSI Design in KL University. His research interests include Low Power VLSI Design. He has undergone 3 International Journals and 1 publishment in IEEE. E-Mail: sastryece52@gmail.com