VERIFICATION OF UART IP CORE USING UVM

Supriya Kamshette [1], Sangamesh A Gama[2], Bhemrao Patil[3]

[1]ECE Dept, Bidar, Karnataka, India
[2]Assistant Professor, CSE Dept, BKIT Bhalki, Bidar, Karnataka, India
[3]Assistant Professor, CSE Dept, BKIT Bhalki, Bidar, Karnataka, India

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. This core is designed to be maximally compatible with industry standard designs[4]. The key 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. Register level and functional compatibility. FIFO operation. The design is verified using UVM methodology. The test bench is written with regression test cases in order to acquire maximum functional coverage.

***Keywords:** UART, UVM, FIFO, WISHBONE INTERFACE*

---***---

1. INTRODUCTION

The data transmission takes place in between the chips, inside the chips and in between the systems also. As it is asynchronous clock there will be 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.

2. 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 UARTS is based on asynchronous serial mode of transmission [1]. Hand shaking between the UARTs 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.

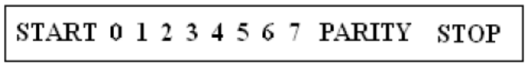

Fig -1: Bit Details

The architecture of UART consists of

2.1 Wishbone Interface

The wishbone interface is the standard computer bus interface that allows communication between the integrated circuits. The wishbone bus permits 8-bit and 32-bit data transfer.

Table -1: Wishbone Interface Signal

PORT	WIDTH	DIRECTION
CLK	1	INPUT
WB_RST_I	1	INPUT
WB_ADDR_I	5 or 3	INPUT
WB_SEL_I	4	INPUT
WB_DAT_I	32 or 8	INPUT
WB_WE_I	1	INPUT
WB_STB_I	1	INPUT
WB_CYC_I	1	INPUT
WB_DAT_O	32 or 8	OUTPUT
WB_ACK_O	1	OUTPUT

2.2 Interrupt Registers

This register is used to enable and identify the interrupts. There are two types of interrupt registers. They are
a) Interrupt Enable Register
b) Interrupt Identification Register.

The interrupt enable register enables and disables with interrupt generation by the UART. It is of 8-bit width. The interrupt identification register enables the programmer to retrieve the current highest priority pending interrupt. BIT-0 indicates that an interrupt is pending when its logic 0.when it is 1 the interrupt is not in pending. The width of the register is 8-bit.

2.3 Control Registers

There are two Control registers. They are
a) FIFO control register
b) 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.

2.4 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].

2.5 Divisor Latches

The divisor latches can be accessed by setting the 7th bit of LCR to 1.Restore the bit to zero after setting the divisor latches in order 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/O operations 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 then LSB last.

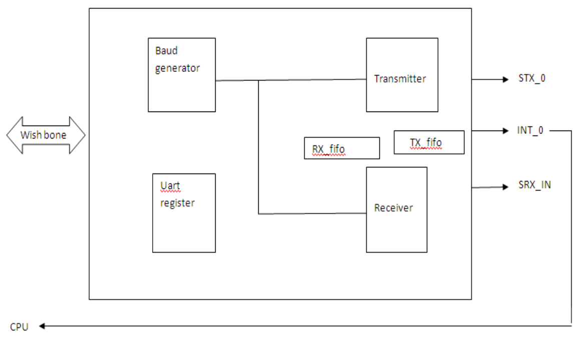

Fig -2: Block Diagram of UART IP Core

3. VERIFICATION METHODOLOGY

The UVM methodology test bench environment consists of

3.1 Agent Top

TB consists of two Agent top each of which consists one agent for UART A and for UART B, both are active so it consists of MONITOR, DRIVER, SEQUENCER, and CONFIG class. Sequences will be outside of TB, it will be connected to sequencer in run_phase of test case using start method. The sequences will generates stimulus and gives to Driver via Sequencer. This each agent is connected to DUT through static interface in top.

3.2 Driver

Driver drives the signals which received through the sequencer into the pins of DUT.

Declare a virtual interface in the driver to connect the driver to the DUT. And we are using macros for printing, copying, comparing. Virtual Interface is provided as there is a connection between Static and dynamic components

a) Sequence Configuration

Divisor latch (MSB) reg. is programmed first and the LSB reg, as follows:

task body();
(addr=3,data = 8'b10000011); *//configure LCR 8*th *bit to access divisor latch register.*

(addr=1,data); *//setting divisor –MSB*

(addr=0,data); *//setting divisor-LSB*

(addr=3, data=8'b00000011); *// LCR 8th bit made zero for accessing other register.*

Programming IER, FCR, LCR

(addr=1, data); *// IE*
(addr=2, data); *// FCR*
(addr=0,data) *//THR*
(addr=3, data); *// LCR* ….etc

end task

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

3.3 DUV

The device that is to be verified is called DUV. The DUV is driven with a group of inputs and the output is compared with the reference model in the score board. The functionality is verified using functional coverage.

3.4 Monitor

The monitor extracts the signal information from the bus and translates it into transaction. Monitor is connected to scoreboard via TLM interfaces Analysis ports and exports.

3.5 Virtual Sequence/Sequence

Virtual sequencer is used in the stimulus generation process to allow a single sequence control activity via several agents. It is not attached to any driver, does not process items itself has reference to multiple sequencers only on virtual sequencers in an agent environment

3.6 Config Database

It is reusable, efficient mechanism for organizing configuration

3.7 Score Board

In Scoreboard comparing the inputs of the UART 1 monitor with output of UART2 receiver.

a) Test cases are written in virtual test class extended from uvm_base_test
b) In this we import the package, build the environment, and run this environment.
c) The logic inputs are randomized in using class of transaction which was extended from uvm_sequence_iem.

d) Different test cases have been created for verifying half duplex and full duplex modes.

Coverage Report

1) Functional Coverage:
  a) This measures how much of the design specification has been exercised.
  b) In our verification % specification has been exercised.

2) *Code Coverage*
  a) This measures how much of the code has been executed.
  b) In this verification 100% of the code has been exercised.

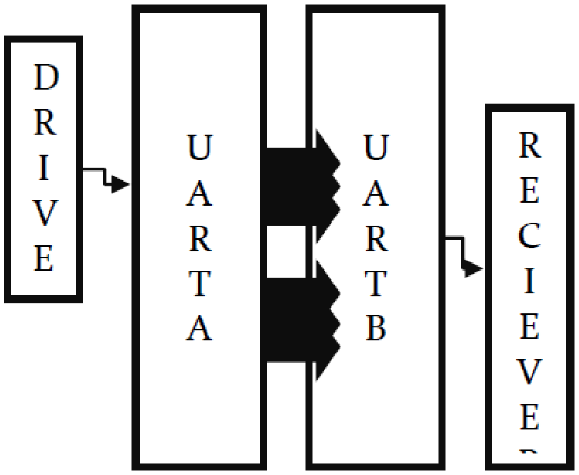

Fig -3: Half Duplex Mode

i. Sequences & 1 scoreboard.
ii. Use 1 driver & 1 receiver.
iii. Send Parallel data into UART A, UART A convert parallel to serial UART B conv serial to parallel Received parallel data. Both data IN & OUT are compared & verified.[3]

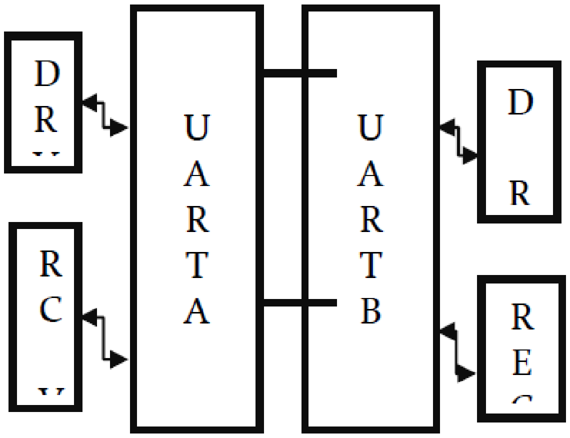

Fig -4: Full Duplex Mode

i. Uses 2 driver & monitor
ii. Send Parallel data into UART A
iii. UART B convert parallel to serial
iv. UART B convert serial to parallel
v. Received parallel data.
vi. Both data IN & OUT are compared [4]

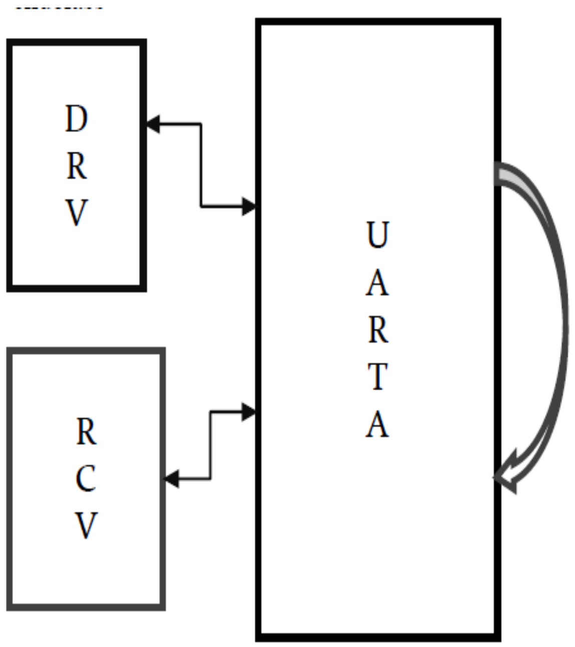

Fig -5: Loop Back Mode

In this mode only one UART will work by same UART we will transmit a data and also receives. By making MCR 4th bit as high. So internally connection will happen to work in[5][6].

4. TESTING

TESTCASE 1 -- Basic Transaction with 14 data's and no parity and 1 stopbit, no stick and no break, 8 bits in the character

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

5. RESULTS

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

6. CONCLUSIONS

The UART IP core working has been verified for different modes of operations i.e. HALF DUPLEX MODE , FULL DUPLEX MODE and LOOP BACK MODE. Currently I'm worked on coverage of all the functions. And also verified corner cases and the check the functionality of registers. The UART transmission from serial to parallel and vice-versa is attend and has been verified.

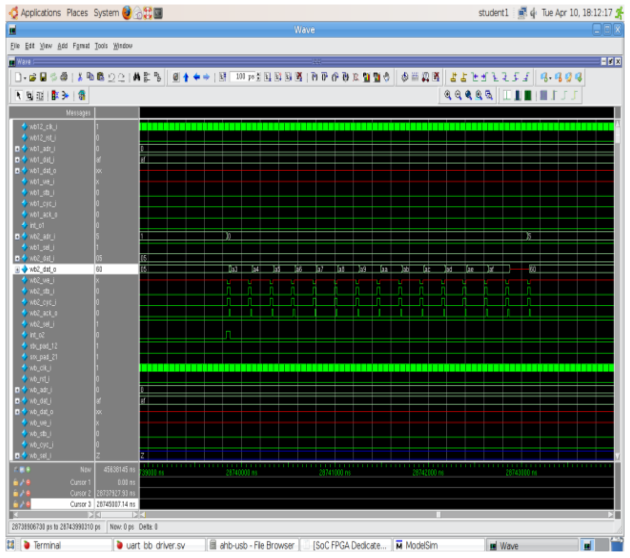

Fig -6: Simulated Waveform

REFERENCES

[1]. I.E.Sutherland'Micropiplines'Communication ACM, June 1989, Vol. 32(6), pp. 720 -738.
[2]. V.N. Yarmolik, Fault Diugnosis of Digital Circuits, JohnWiley & Sons, 1990
[3]. "PCI6550DUniversal Asynchronous Receiver/ Transmitter with FIFOs", National Semiconductor Application Note, June 1995.
[4]. M.S.Harvey, "Generic UART Manual" Silicon Valley. December 1999.
[5]. "PCI6550DUniversalAsynchronous Receiver/Transmitter with FIFOs", National Semiconductor Application Note, June 1995
[6]. Martin S. Michael, "A Comparison of the INS8250, NS16450 and NS16550AF Series of UARTs"National Semiconductor Application Note 493, April1989.
[7]. W.Elmenreicb, M.Delvai,TimeTriggered Communication with UARTs. In Proceedings of the 4'h IEEE International Workshop on Factory Communication Systems, Aug. 2002.