Universal Asynchronous Receiver and Transmitter (UART)

Conference Paper · January 2016

DOI: 10.1109/ICACCS.2016.7586376

CITATIONS
22

READS
11,126

2 authors:

Umakanta Nanda
VIT-AP University, Amaravati
52 PUBLICATIONS **186** CITATIONS
SEE PROFILE

Sushant Pattnaik
Silicon Institute of Technology
5 PUBLICATIONS **53** CITATIONS
SEE PROFILE

2016 3rd International Conference on Advanced Computing and Communication Systems (*ICACCS* -2016), Jan. 22 – 23, 2016, Coimbatore, INDIA

Universal Asynchronous Receiver and Transmitter (UART)

Umakanta Nanda, Sushant Kumar Pattnaik
Department of Electronics and Communication Engineering
Silicon Institute of Technology
Bhubaneswar, India
umakanta.nanda@silicon.ac.in, sushanta.pattnaik@silicon.ac.in

Abstract— **All most all computers and microcontrollers have several serial data ports used to communicate with serial input/output devices such as keyboards and serial printers. By using a modem connected to a serial port serial data can be transmitted to and received from a remote location via telephone line. The serial communication interface, which receives and transmits the serial data is called a UART (Universal Asynchronous Receiver-Transmitter). RxD is the received serial data signal and TxD is transmitted data signal. In this project UART is implemented in virtex II pro FPGA chip due to low cost, high speed, reprogram ability and fast time to market.**

Keywords— UART; DTE; RS-232C; RxD; TxD

I. INTRODUCTION

An universal asynchronous receiver and transmitter (UART) is an integrated circuit which is programmed to control a computer's interface to its attached serial devices [3]. Specifically, it provides the system with the RS-232C Data Terminal Equipment (DTE) interface, enabling it to talk to and exchange data with modems and some other serial devices. Being a part of this interface, the UART also provides the basic operations as:

- Converts the bytes it gets from the computer along parallel circuits to a single serial bit stream for outbound transmission.
- For inbound transmission, converts the serial bit stream to the bytes that the system handles.
- Adds a parity bit after selection in outbound transmissions, checks the parity of incoming bytes (if selected) and rejects the parity bit.
- Adds start and stop delineators for outbound and helps to strip them from inbound transmissions.
- Handles interrupts from keyboard and mouse (which are serial devices with special ports).

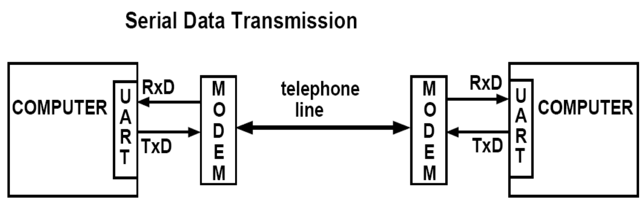


Fig. 1. Serial data transmission.

II. UART DESIGN

The UART block diagram consists of three main components, *Transmitter control, Receiver control and Baud rate generator*. When transmitting , the UART takes eight bits of parallel data, converts the data to a serial bit stream that has a start bit (logic ‘0’), 8 data bits, and a stop bit (logic ‘1’). When receiving, the UART initially detects a start bit, then receives a stream of 8 data bits and translates the data into parallel when it detects the stop bit. As no clock is transmitted, the UART must synchronize incoming stream of bits with the local clock [3].

The following six 8bit registers are used.

1- RSR- Receive shift register.
2- RDR- Receive data register.
3- TDR- Transmit data register.
4- TSR- Transmit shift register.
5- SCCR- Serial communications control register.
6- SCSR- Serial communications status registers [5].

Assume that the UART is connected to a microcontroller data and address bus so that the CPU can read and write to the registers. RDR [5], TDR, SCCR and SCSR are memory mapped. RDR, SCSR, SCCR can drive the data bus through tristate buffers.TDR and SCCR [5] can be loaded from the data bus.

Besides the registers, the three main components of the UART are the Baud rate generator, the receiver and transmitter control. The Baud rate generator divides the clock of the system down to provide the bit clock (bclk) with a period equal to one bit time and also bclkx8, which has a frequency eight times the bclk frequency. The TDRE (transmit data register empty) bit in the SCSR is set when TDR is empty.

A. Baudrate Generator

The 8 MHz clock system clock is first divided by 13 using a counter. This counter output goes to 8 bit binary counter. The output from the flip-flops in this counter relates to divide by 2, 4 and so on upto divide by 256. Out of these outputs, one is selected by a multiplexer. The multiplexer selects inputs that come from the lower 3bits of the sccr. The output corresponds to bclkx8, which is again divided by 8 to give bclk.

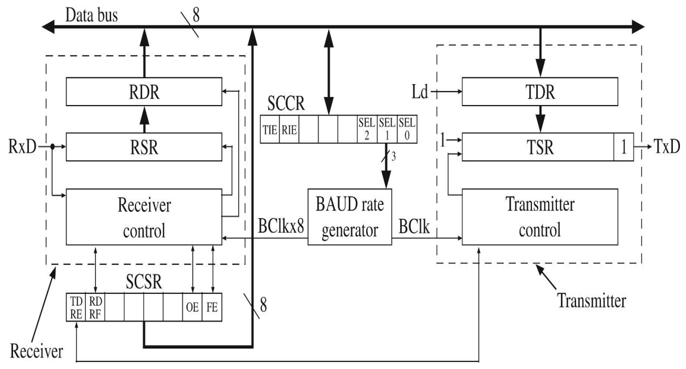


Fig. 2. UART design model block diagram

Initially the process increments the divide by 13 counters on the rising edge of the system clock. The second process increments the divide by 256 counters on the rising edge of clkdiv13. A concurrent statement generates the mux output, bclkx8.The third process increments the divide by 8 counters on the rising edge of bclkx8 to generate bclk.

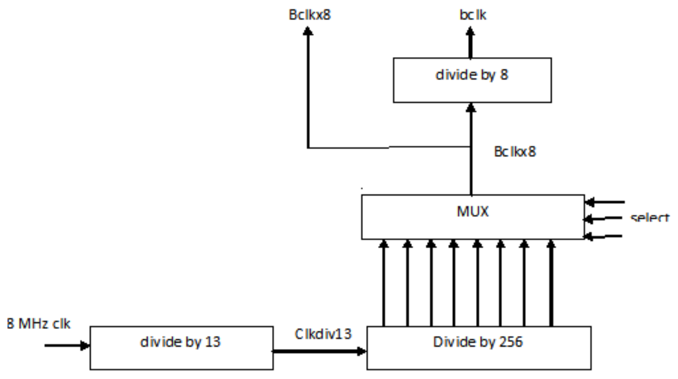


Fig. 3. Baud rate generator block diagram.

B. *UART Transmitter*

When the microcontroller is ready to transmit data, the following occurs.

1. The microcontroller waits until TDRE='1' [5] and then loads a byte of data into TDR clears TDRE.
2. UART transfers data from TDR to TSR and sets TDRE.
3. The UART provides an output as a start bit '0' for one bit time and then shifts TSR right to transmit the eight data bits followed by the stop bit '1'.
4. In this step the UART transfers data from TDR to TSR and sets TDRE.
5. The UART gives a output as a start bit '0' for one bit time and shifts TSR right to transmit the eight data bits followed by the stop bit '1'.

SM chart of transmitter

In the IDLE state, the SM [5] waits until TDR is loaded and consequently TDRE is cleared. In SYNCH state, the SM waits for the rising edge of the bit clock and then clears the low order bit of the TSR to transmit a '0' for one bit time. In the TDATA state, each time bclk↑ is detected, TSR is shifted right to transmit the next data bit and the bit counter (bct) is incremented. When bct=9 eight data bits and a stop bit have been transmitted, bct is then cleared and the SM goes back to idle.

The transmitter contains the TDR and TSR registers [5] and the transmit control. It interfaces with TDRE and data bus (DBUS) [5].The first process represents the combinational network, which generates the next state and control signals. The second process updates the registers on the rising edge of the clock. The signal bclk_rising is '1' for one system clock time following the rising edge of bclk. To generate bclk_rising, bclk is stored in a flip-flop named bclk_dlayed. Then bclk_rising is '1' if the current value of bclk is '1' and the previous value is '0'.

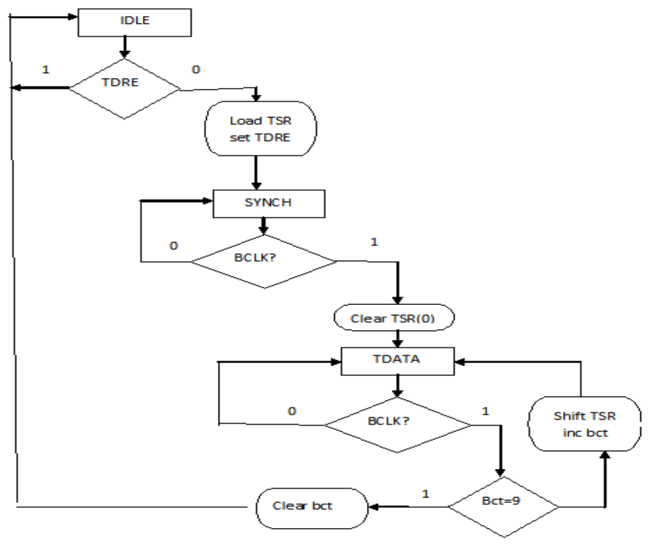


Fig. 4. SM chart of transmitter

C. *UART Receiver*

1. When UART detects a start bit, it reads the left over bits serially and shifts them into RSR.
2. When all the data bits and the stop bit are received, the RSR loads into RDR and the flag of Receive Data Register Full (RDRF) [5] in the SCSR is set.
3. The microcontroller checks for the RDRF flag, and if it is set, the flag is cleared by reading RDR.

RxD is sampled eight times during each bit time. It is sampled on the rising edge of the BClkX8. The bit is read in middle of each bit time for maximum reliability. When RxD first goes low, we will wait for four BClkX8 periods, which should take us closer to the middle of the first data bit. Then reading once per eight BClkX8 clocks is continued until we have read the stop bit.

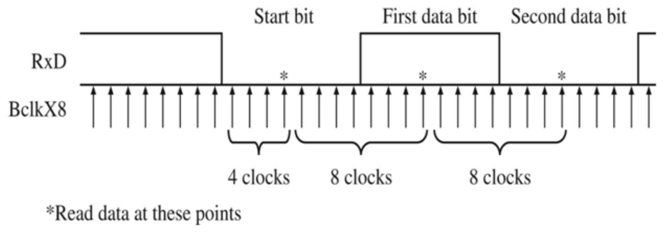


Fig. 5. Sampling RxD with Bclkx8

SM chart of receiver

Two counters are used ct1 counts the number of BClkX8 clocks [5]. ct2 counts the number of bits received after the start bit is encountered. In the IDLE state, the SM waits for the start bit (RxD = ‘0’) and then goes to start detected state. Now the SM waits for the rising edge of BClkX8 and samples RxD once more. Since the start bit should be ‘0’ for eight BClkX8 clocks, a ‘0’ should be read. As ct1 is still 0, it is incremented and SM waits for the rising edge of BClkX8.If RxD = ‘1’, this is an error condition and SM clears ct1 and resets to the IDLE state [5]. Otherwise SM keeps looping till RxD is ‘0’. When RxD is ‘0’ for the fourth time, ct1 = 3, so ct1 is cleared the state goes to receive data. In this state, the SM increments ct1 after every rising edge of BClkX8.After the eighth clock, ct1 = 7 and ct2 is checked. If it is not 8, the current value of RxD is shifted to RSR, ct2 is incremented, and ct1 is cleared. If ct2 = 8, all the 8 bits have been read and we should be at the middle of the stop bit. If RDRF = 1,the microcontroller has not yet read the previously received data byte, and an overrun error has occurred, where the OE flag in the status register is set and the new data is ignored. If RxD = ‘0’, the stop bit has not been detected properly, and the framing error (FE) [5] flag in the status register is set. If no errors have occurred, RDR is loaded from RSR. In all cases, RDRF is set to indicate that receive operation is completed and counter is cleared.

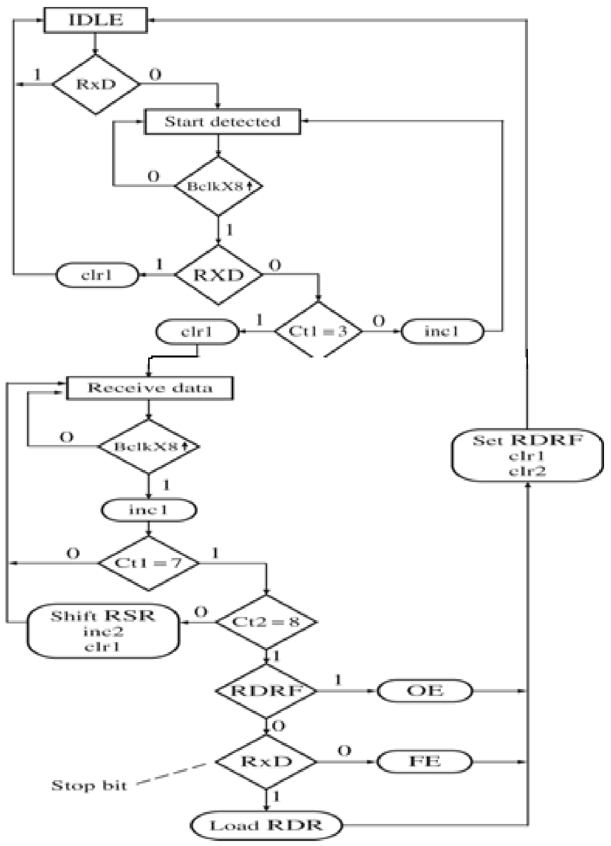


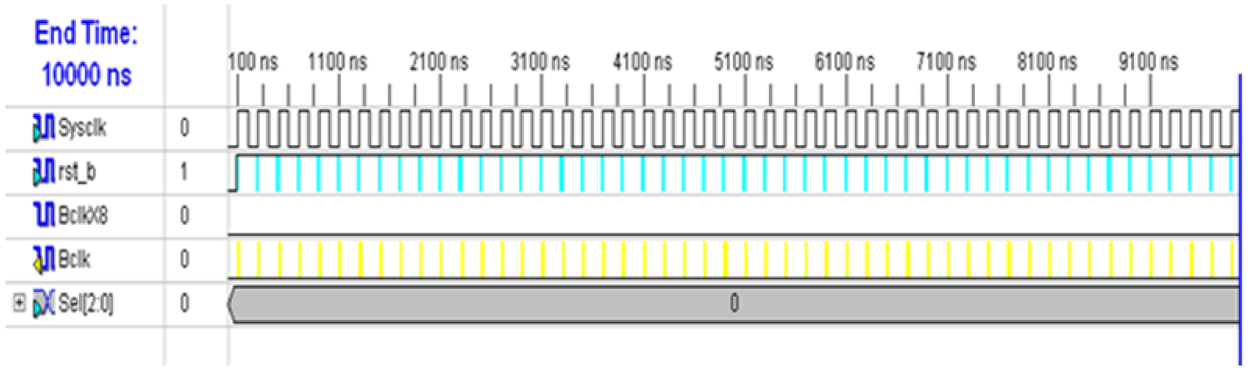
Fig. 6. SM chart of receiver

The receiver contains the RDR and RSR registers and receives control. The control interfaces with SCSR [5], and RDR can drive data on to the data bus. The first process represents a combinational network, which generates the next state and control signals. The second process updates the registers on the rising edge of the clock. The signal BclkX8_rising is generated the same manner as Bclk_rising.

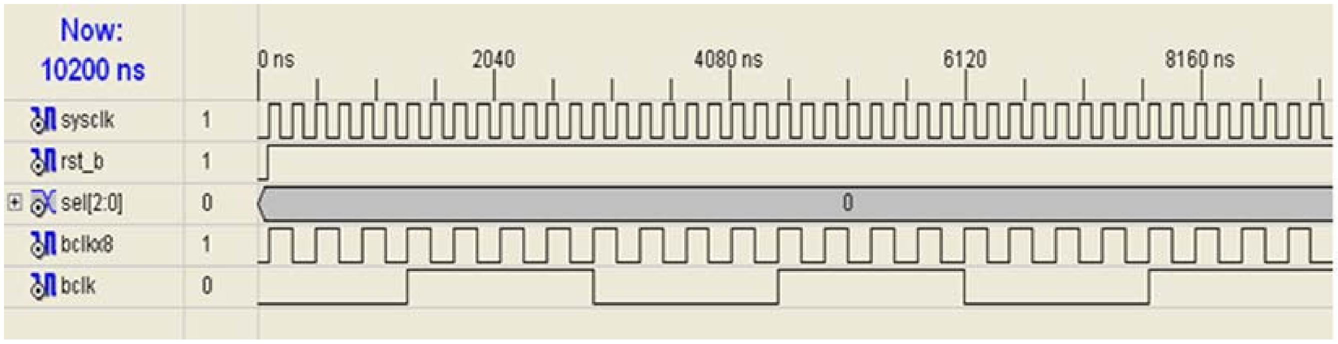
III. VHDL SIMULATION

A. BAUDRATE GENERATOR

1) Test bench input

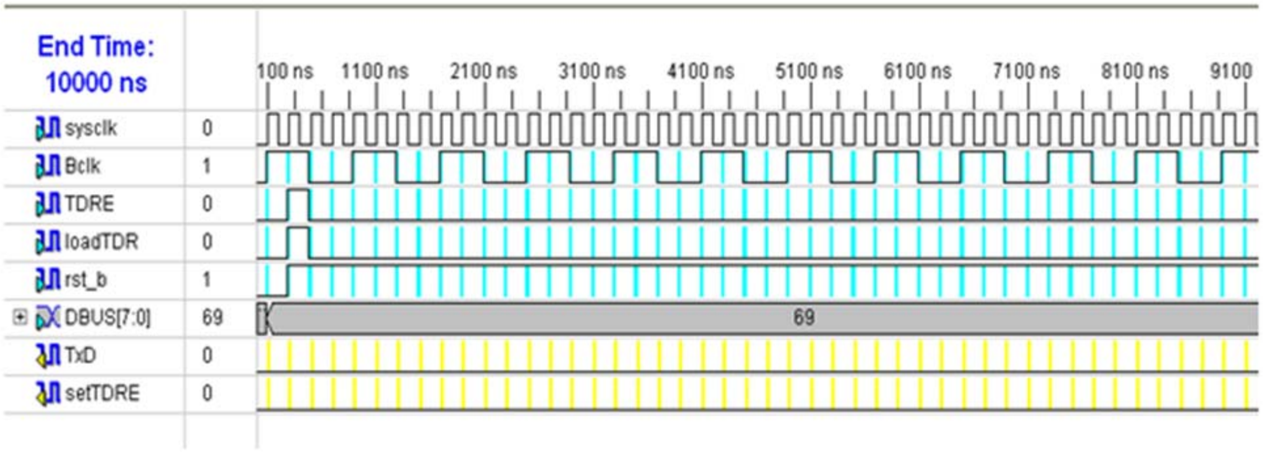


2) Test bench output

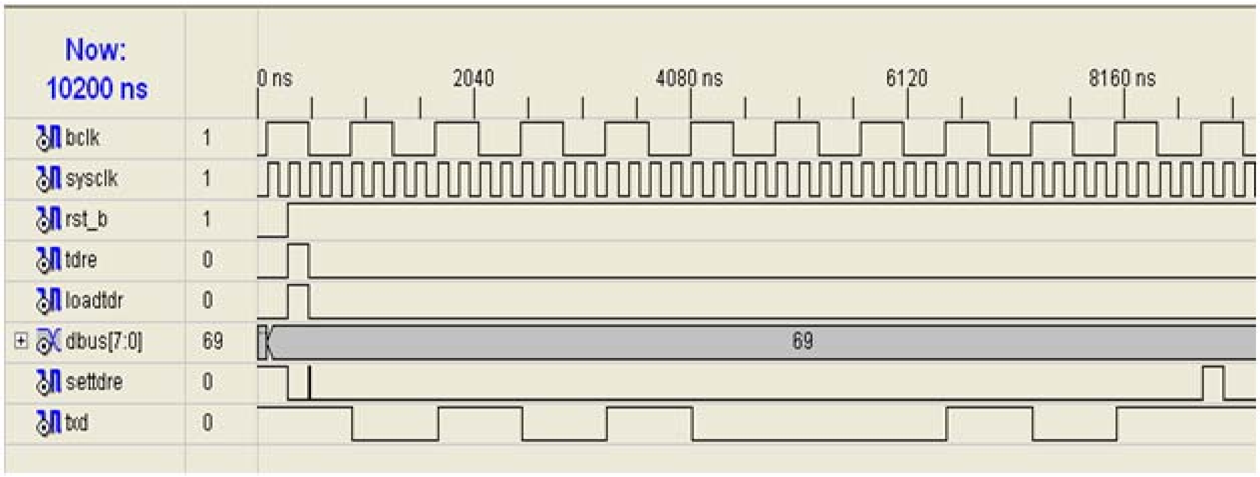


B. TRANSMITTER

1) Test bench input

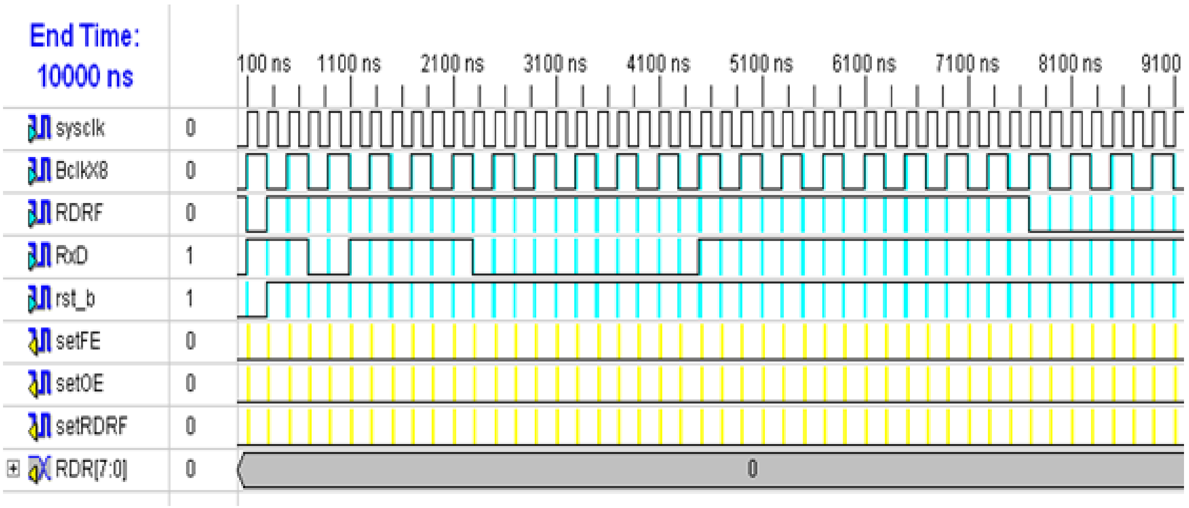


2) Test bench output

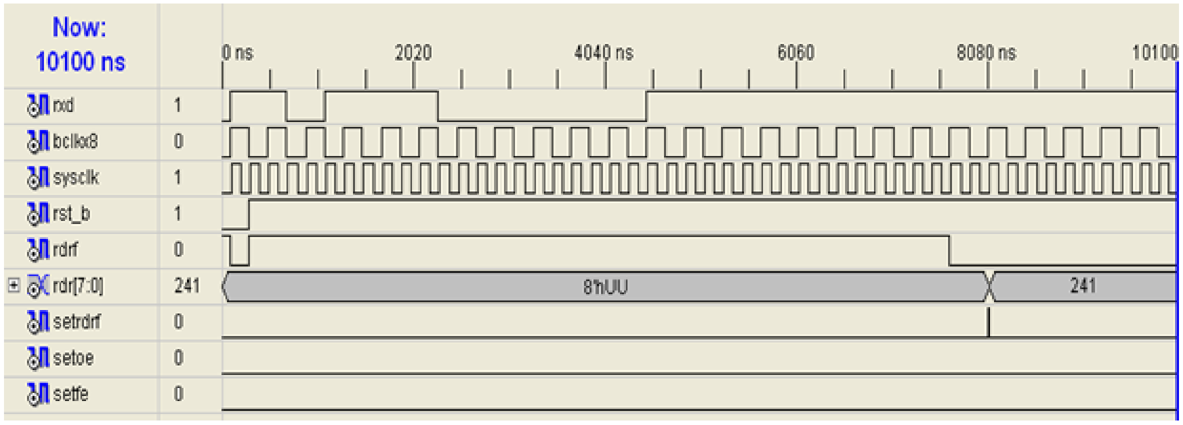


C. RECEIVER

1) Test bench input



2) Test bench output



IV. CONCLUSION

The architecture is successfully implemented in FPGA. Fast time to market, low cost for small production volume and reprogram ability make FPGA devices an ideal solution for military and university research. The high speed capability and register rich architecture of FPGA an ideal implementation of UART architecture. The design implementation entailed the employment of Xilinx ISE8.2 software tool. Implementing the design on a virtex-II and hardware testing and verification of the UART could be done. Finally simulation and synthesis with Xilinx xst tool and RTL schematic of filter chip was obtained.

REFERENCES

[1] Douglas L. Perry, "*VHDL Programming by examples*", TMH Publication.

[2] J.Bhaskar , "*A VHDL Primer*", Pearson Education.

[3] www.wikipedia.org/uart

[4] Stefan sjoholm, Lennart Lindth *"VHDL for Designers"* Prentice Hall ist edition, 1997.

[5] Walter A. Triebel & Avtar Singh, "The 8088 and 8086 Microprocessors"