Military Technical College
Kobry Elkobbah,
Cairo, Egypt

5th International Conference
on Electrical Engineering
ICEENG 2006

FPGA HARDWARE IMPLIMENTATION FOR EXTERNAL MODEM INTERFACING

Islam Tawfik AbouGindia* Khaled Shehata** M. A. Elkfafi*

Abstract:

Intelligent modems used in military messaging over IP networks require flow control mechanism to prevent overflow of data into its buffers. We contribute for a complete flow control mechanism with built in UART module. All modules are designed using VHDL design entry. All modules are simulated, synthesized then integrated, and downloaded on an FPGA. Timing Simulation results found to be matching with the hardware results obtained from the logic analyzer.

1. Introduction:

Consider Figure 1, which represents the wireless modem used for military messaging over IP networks [1]. The "worst case" characteristics of long-range tactical radio communications over HF link providing IP services is that only low to moderate data rate is supported (< 10 kb/s) since HF propagates via reflecting layers of the ionosphere that support a very limited data rate. Thus the wireless modem will transfer the data to the HF link at a rate much less than that communicating with the terminal PC which is for example 19200 bits per second. Under very favorable conditions, a maximum of 9.6 kb/s user data rate can be achieved in a 3 kHz channel [2]. This situation would cause data flowing into the modem to be lost. To prevent a potential data loss, modem that performs other functions, including error detection and correction, will include data buffers [3]. Because data buffers represent a finite amount of storage, a modem must be able to control the flow of data into its buffers.

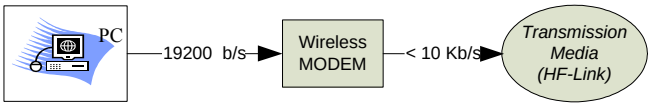

Figure.1. Wireless modem used for military messaging over HF link

We use hardware flow control mechanism. The RTS/CTS method is referred to as hardware flow control [4]. We are designing a base-band external intelligent modem with automatic error detection and correction capabilities. It is used in wireless communication systems for military messaging in IP networks using HF links for tactical communications. The modem is used by highly mobile units not able to utilize a fixed network communications infrastructure. Typical tactical units are naval vessels, aircrafts, land mobiles, and Special Forces carrying

* *Military Technical College, Cairo, Egypt*
** *College of Engineering, Arab Academy for Science and Technology, Cairo, Egypt*

man pack radios [1]. Military Messaging in IP Networks Using HF Link is shown in Figure 2.

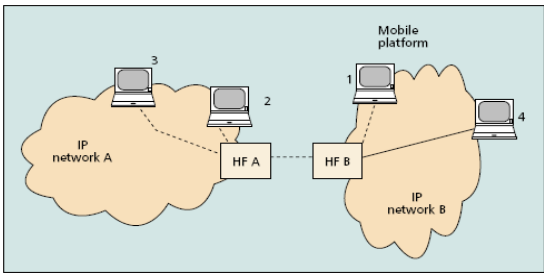

Figure.2. Military Messaging in IP Networks Using HF Link

This paper introduces part of design of a modem that provides the following capabilities:

- Automatic error detection and correction to overcome the disadvantages of data communications over HF link. We use both forward and backward error correction techniques.
- Serial interface with the terminal PC.
- Flow control.
- Protocol management.

In our design we implemented the Y-modem asynchronous file transfer protocol with some modifications to improve the communication performance over the HF link [5].

For our modem, the UART used will be simplified just for the purpose of the modem operation where we don't need the error detection circuit (Parity Generator), and the bit rate will be constant (115200 bps). And due to these simplifications we decreased the system delay, and the implementation area on the FPGA chip.

2. Proposed Modem Design:

The system level for the communication system is shown in Figure 3.

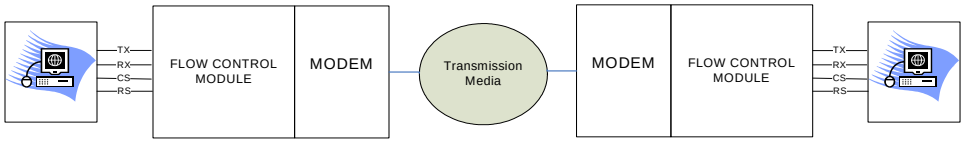

Figure.3. Data Communication system with flow control

The detailed architecture of the base-band modem along with its flow control modules is shown in Figure 4. The proposed modem transmitter architecture consists of the following modules as shown in Figure 4:

1. **Flow control modules:**
   a. UART module: used for serial interface between the external modem and the PC through the RS-232 serial port.
   b. Buffers: used for data storage and flow control purposes to achieve pipelining data transmission.
   c. Control Unit: to control all of the data paths and flow control operation.
2. **Hamming code generator circuitry:** used for the generation of the redundant bits needed to be inserted with the raw data to achieve Forward Error Correction.
3. **CRC generator circuitry:** used for the generation of the 16 redundant bits; this is used for automatic error detection.
4. **Packetizing module:** used for the generation of the packet sequence, the complement of the packet sequence for detecting any errors in the packet sequence, the packet header, and the Address for communication over networks.

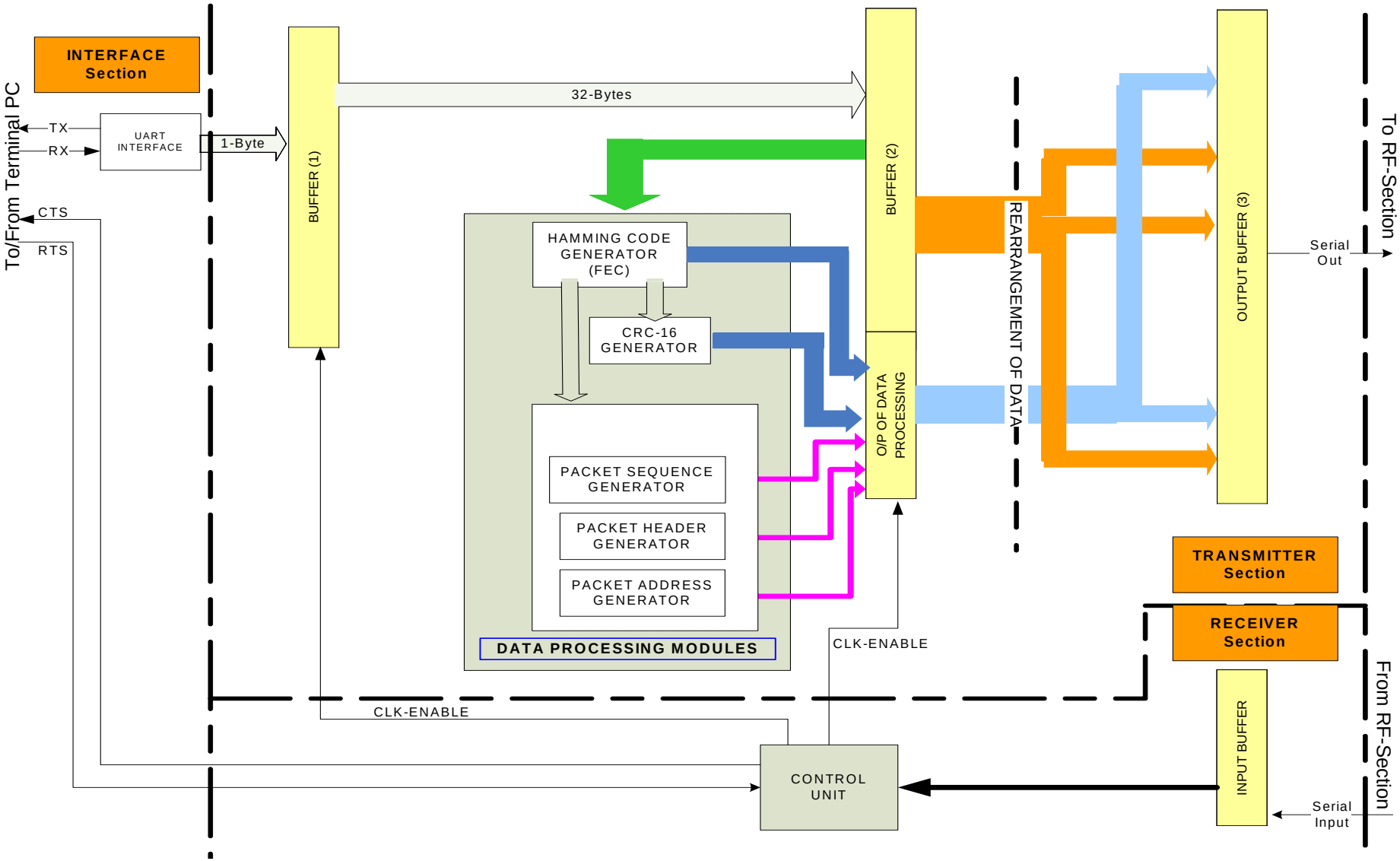

Figure .4. Detailed Architecture of the Designed Modem

3. UART Module Design

The UART module composes of the following sub modules:

1. UART Transmitter.
2. UART Receiver.
3. Fixed Baud rate generator.
4. Clock Divider.

The starting point for the design is the top-level block diagram. The Modem serial device has the ability to reset the UART and to write to and read from the registers in the UART.
The UART interface is designed for use with an External Modem serial device and consists of the following signals:

- **Clk_brd:** The FPGA board system clock. 50% duty cycle. Typically 30 MHz clock frequency.
- **Reset:** Active high Asynchronous reset.
- **Tx_data [7:0]:** This is the 8-bit Parallel input data bus which carries the data in which the Modem writes to the UART transmit register.
- **Tx_register_empty:** this signal to inform the Modem receiver buffer that the UART transmit register is empty (Transmitter empty flag) this happens after the last bit of data (MSB) is sent.
- **Shift_Load:** This signal is the synchronous load, active low pulse to the UART from the Modem where the data have to be loaded once from the Modem Receiver buffer to the transmit register then shift this data after the framing process.
- **Data_Out [7:0]:** This is the 8-bit Parallel output data bus which carries the data received by the UART receiver to the Modem Transmitter buffer.
- **Data_Ready:** This signal to inform the Modem transmitter buffer that a new unframed data byte is received at the receiver register and ready to be read.

The top design for the UART used in our modem (TOP_DESIGN) is shown in Figure 5.

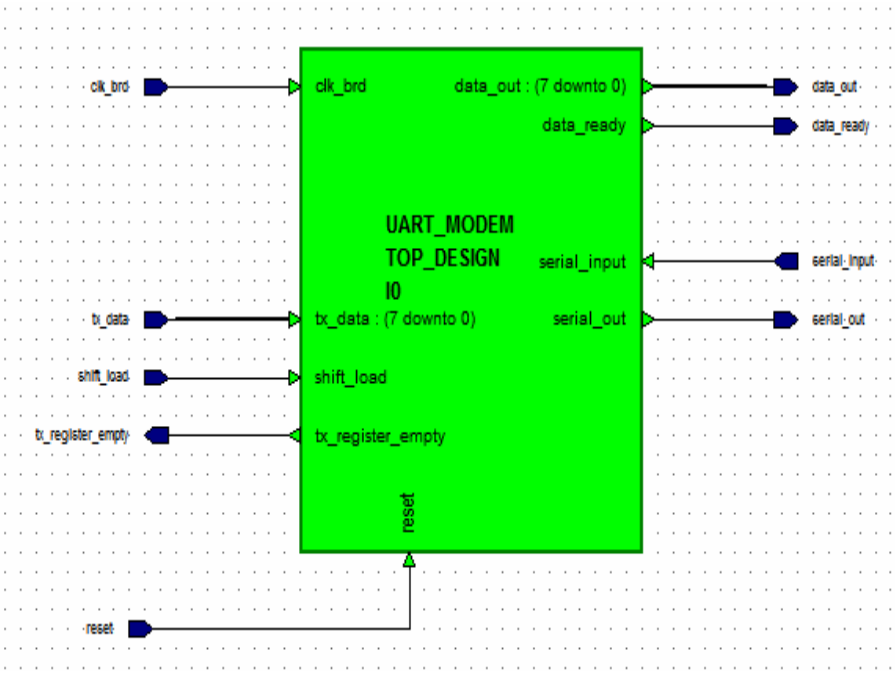

Figure.5. UART Top Design Symbol

The UART main sub-modules are shown in Figure 6.

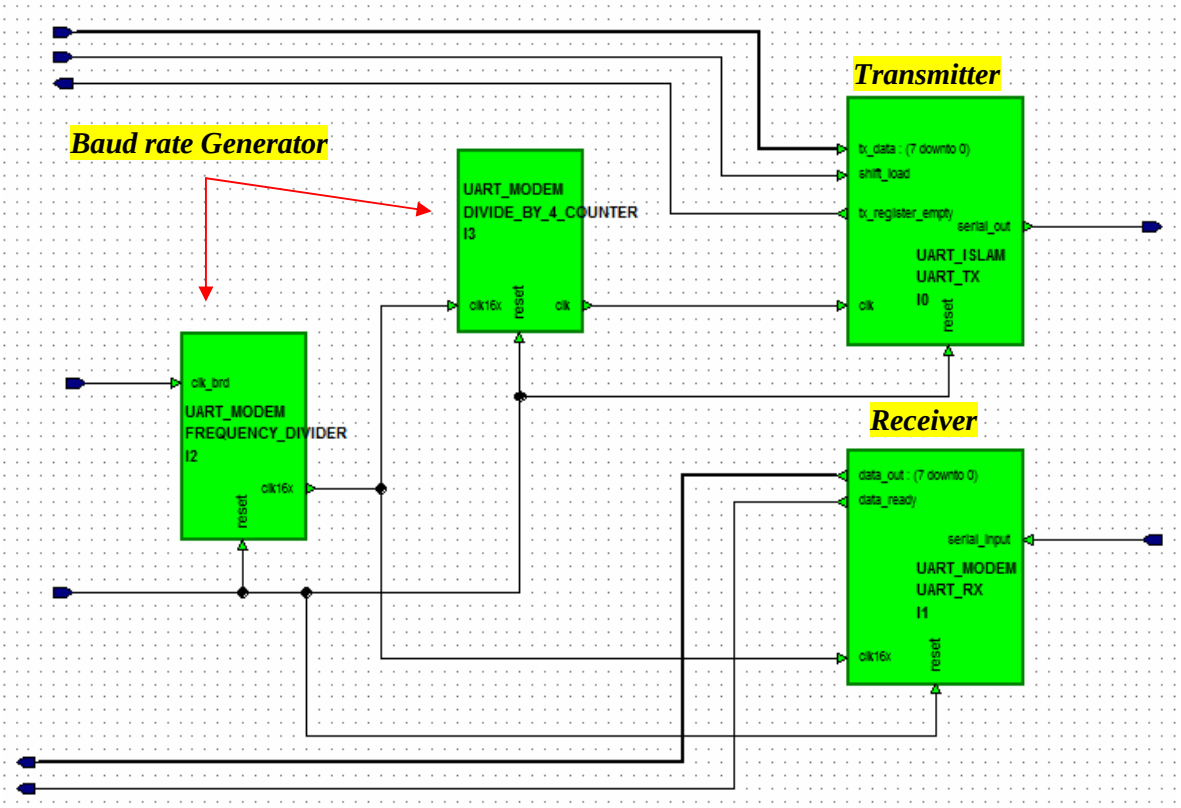

Figure.6. UART main sub-modules

3.1 UART Transmitter: (UART_TX)

The UART transmitter top design symbol (UART_TX) is shown in Figure 7.

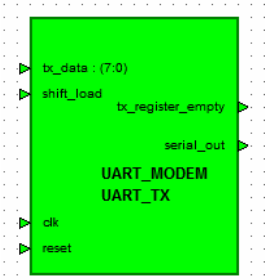

Figure.7. UART Transmitter

The UART frames the data with adding to it a START bit '0' at the beginning and a STOP bit '1' at the end of the packet so that the packet will be as shown in Figure 8.

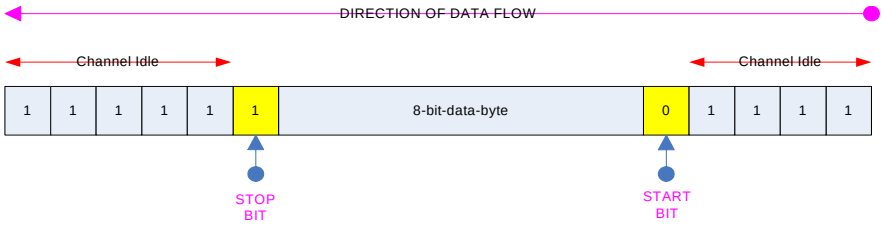

Figure.8. UART output Framed Data

Figure 9 shows an example simulated using Mentor Graphics Modelsim. The Modem loads a value of "10101010" to the UART Transmitter. As soon as the data is loaded, the UART transmitter asserts on the Serial_Out signal the START bit, followed by the 8-bit data from LSB to MSB, followed by a STOP bit. In this simulation the data is "10101010". Thus, the serial data sequence from LSB to MSB is "0-01010101-1" when the START and STOP bits are considered. When the data is sent, UART Transmitter deasserts a Transmitter Empty (tx_register_empty) signal to flag the Modem that it cannot receive another byte. When all bits are sent, it asserts the (tx_register_empty) signal so that at the next clock cycle a new data can be loaded.

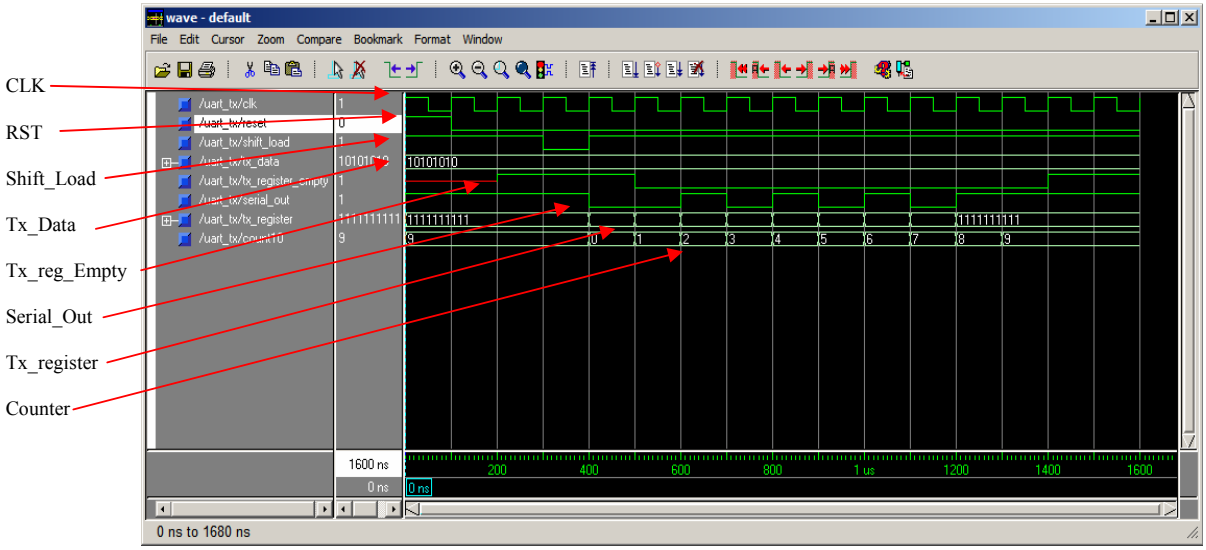

Figure.9. UART Transmitter Simulation results

3.2 UART RECEIVER: (UART_RX)

The UART receiver top design symbol (UART_RX) is shown in Figure 10.

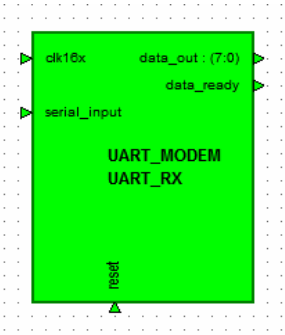

Figure.10. UART receiver symbol

- The UART receiver consists of receive input register (serial_input_register) to store the incoming serial data from the PC and synchronize the data with the system clock. The serial_input_register reclocks the serial input data received from the computer (serial_input) with a 16 times clock, to sample each received bit 16 times [6]. A state machine remembers if the receive register is empty (rx_register_empty) and waits for a new message.

Figure 11 shows a simulation using Mentor Graphics Modelsim. The UART receiver receives the following data "00110011" consecutively, for each data bit received it is sampled by a 16

clock. Each data bit received is loaded into the serial_input_register, it is supposed that before the detection of the START bit the channel is idle (i.e. stream of 1's), once a START bit is detected at the serial_input_register and receive_register is empty, then the 16-counter is reset to zero, thus synchronizing with the start of a new bit. At this moment, the receive_register will flag that it is not empty, since it is now busy receiving a new message. When the 16 counter reaches the mid clock count (i.e. count = 7) the received data is sampled (read) and written into the 10-bit receive register (i.e. data in the serial_input_register will be shifted to the left into the receive_register). This process is repeated for 10 consecutive data bits received which is the size of the framed message, thus the receive_register now contains 10 data bits such that the tenth bit will be '1' which is the STOP bit while the first bit will be '0' which is the START bit. At this time, the receive_register will flag that it is now empty and can receive any other message, and a Data_Ready signal will flag the Modem buffer (1) to read the 8-bit parallel data at the DATA_OUT bus.

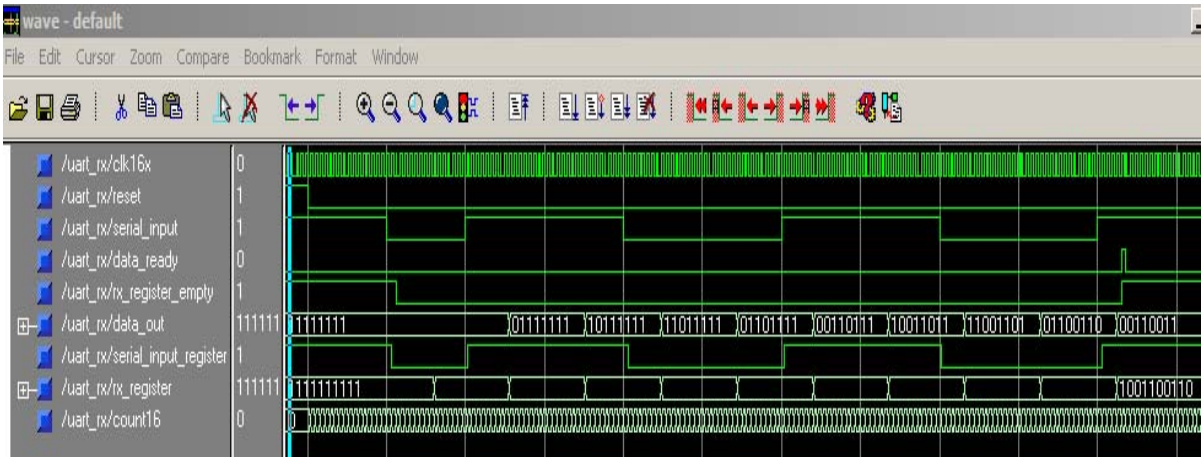

Figure.11. UART Receiver simulation results

For detailed Design description, algorithms and simulation results. Refer to our paper [7].

4. System Setup for testing:

For obtaining the hardware results we used the following devices:

1. Two PC's. (One for Sending data while the other for receiving data)
2. FPGA Kit*. (Where we downloaded our UART transceiver)
3. Power Supply*. (For obtaining a 5V DC required for the FPGA kit)
4. MAX 232 circuit*. (For Voltage level conversion)
5. Logic Analyzer*. (For tracing the output signals)

* Parts are shown in Figure (12).

Complete setup for testing is shown in Figure (12).

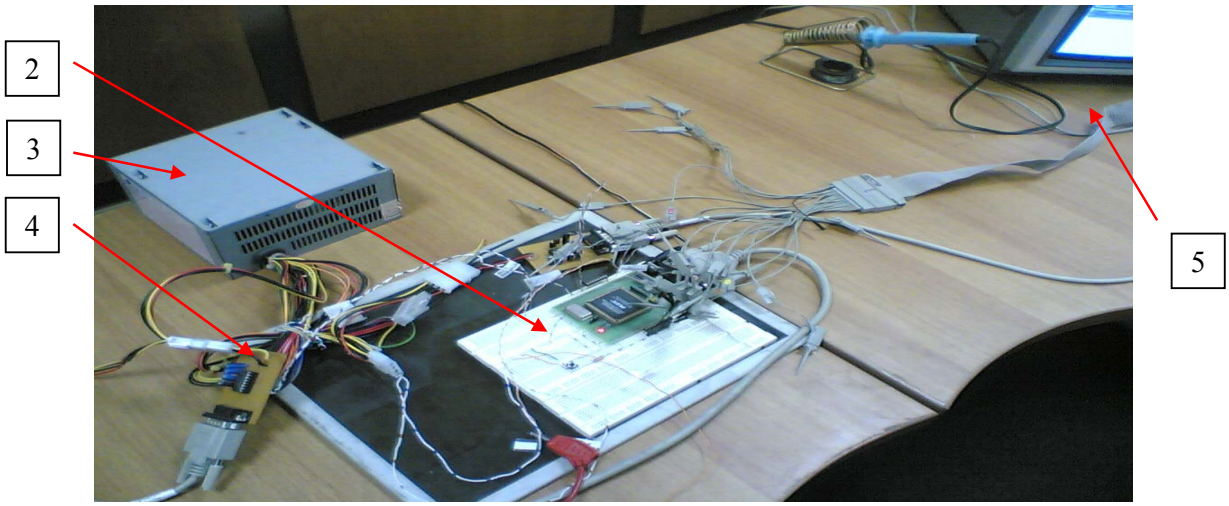

Figure.12. complete setup for testing

The test procedure is that we send a file from PC#1 through COM1, which in turns outputs the file as a continuous serial data stream into the RS-232. This serial data is then received by the UART receiver byte by byte. Each byte received with a data ready flag. We have designed a Feeder module for test purposes only, where not all of the modem modules have been finished yet. This Feeder Transfers the received bytes into the UART Transmitter. When the Feeder detects a data ready flag, it generates a load signal for one cycle, and a shift signal for the rest of the cycles.

Thus the UART Transmitter will transmit the received bytes to PC#2 through COM1 which will receive these data bytes. The received data at PC#2 was compared with the transmitted data from PC#1 and found to be matching. The Hardware results obtained from the logic analyzer at a baud rate of 115200 bits per second are shown in Fig (13), which typically matches the simulation results.

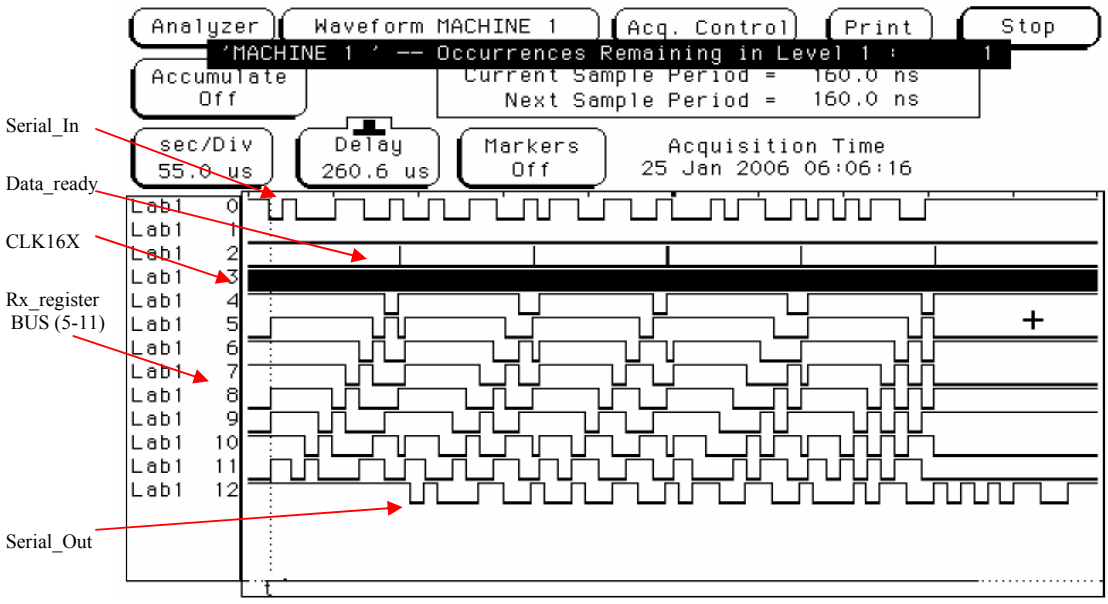

Figure.13. Hardware results

5. Conclusion

In this paper a complete flow control mechanism along with a UART modules are constructed. These modules contribute in establishing a base-band intelligent modem used in military messaging over IP networks. The simulation of these modules using Mentor Graphics is successfully performed. The design is synthesized on ALTERA chip Apex 600KE, and occupies (2%) of the FPGA. The timing simulation is also performed and showed that the design contribute on a speed much higher than the required baud rate. The Hardware testing tested at maximum baud rate of 115200 bits per second. It functioned correctly and typically matches the simulation results.

6. References

[1] Vivianne Jodalen, Anders Eggen, Bjørn Solberg, and Ove Grønnerud. Military Messaging in IP Networks Using HF Links. IEEE Communications Magazine, pages 98-104, November 2004.
[2] NATO STANAG 4406, "Military Message Handling System (MMHS)," ed. 1, 1999.
[3] Gilbert Held. The Complete Modem Reference, John Wiley & Sons, Inc., 1991.
[4] William Stallings. Data & Computer Communications, Prentice Hall, 2000.
[5] Protocols for File Transfer. http://en.wikipedia.org/wiki/Protocols_For_File_Transfer.
[6] Mark Balch. Complete Digital Design, McGRAW-HILL, 2003.
[7] Islam Tawfik AbouGindia, Khaled Shehata, and Magdi ElKafafi. Design and Implementation of a flow control module for External Modem on an FPGA. First Workshop on VLSI and EDA Tools by Mentor Graphics, pages 24-36, December 2005.