IOSR Journal of Electronics and Communication Engineering (IOSR-JECE)
e-ISSN: 2278-2834,p- ISSN: 2278-8735.Volume 10, Issue 2, Ver. IV (Mar - Apr.2015), PP 01-10
www.iosrjournals.org

Serial Communication Protocol Conversion and Circular Buffer Implementation in FPGA using Verilog

Madhulika Sharma[1], Puneet Maheshwari[2], Pravin Fatnani[2], Preet Jain[1]
[1](*Department of Electronics and Communication Engineering, Shri Vaishnav Institute of Technology and Science, Indore, India)*
[2](*Accelerator Control Section, Raja Ramanna Centre for Advanced Technology, Indore, India)*

Abstract: *Raja Ramanna Centre for Advanced Technology (RRCAT) has two synchrotron radiation sources, Indus-1 and Indus-2. Microtron is the common injector to both the machines. A new, FPGA based, control system architecture is planned in which it supervises and monitors different subsystems present for the operation of Microtron with the help of Equipment Control Modules (ECMs). At present some subsystems communicate with operator console directly over RS232 interface such as the 'Tesla Meter' and the 'Temperature Scanner'. For modularity, those subsystems must be brought over RS485 interface and with unified pre-defined custom protocol for communication. In this paper monitoring of the Tesla Meter is proposed. An ECM communicates with Tesla Meter over RS32 interface and with an operator console over RS485 interface. Digital logic is designed in Verilog HDL and implemented in FPGA for RS232 and RS485 communication controllers, respectively. A protocol conversion unit between Tesla Meter and ECM is also implemented. For more feature, there is a need to capture data and store it for offline data analysis. Therefore, a circular buffer is also designed and simulated to review the history data from the ECM. The work has been completed using ModelSim XE and Xilinx ISE 8.2i softwares.*

Keywords: *Buffer, FPGA, Protocol Conversion, RS232, RS485, Serial Communication*

I. Introduction

In RRCAT, there is an injector "Microtron" which sends electron pulses to the Booster. In the new control system architecture for the microtron, ECMs are used to supervise and monitor different sub-systems of the Microtron. ECM communicates with the operator console over RS485 interface. The communication is done in a custom predefined protocol. The ECM needs to be interfaced with various subsystems. All the subsystems do not have the same interface and some of them communicate over RS232 serial interface directly with operator console. "Tesla Meter" is one such subsystem (Fig. 1). For modularity, it is necessary to bring all subsystems over RS485 interface therefore it is required to design a logic to communicate over RS232 & RS485 with their respective protocols and sent data from Tesla Meter to operator console over RS485 interface (Fig. 2).

The operator console may send a 'history data' request to ECM for offline analysis and the ECM must respond to this request in correct way. Therefore, a module is designed to have facility to write a data into internal memory and read & send the data as and when required by the operator console in a proper format.

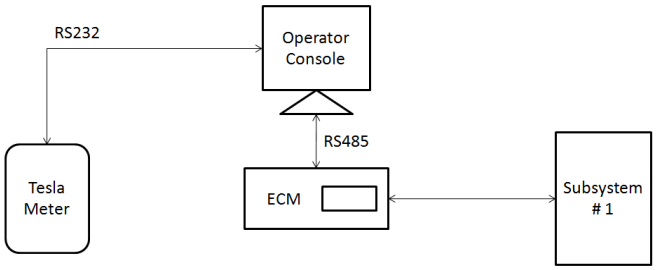


Fig. 1: Communication between Tesla Meter and Operator Console on RS232

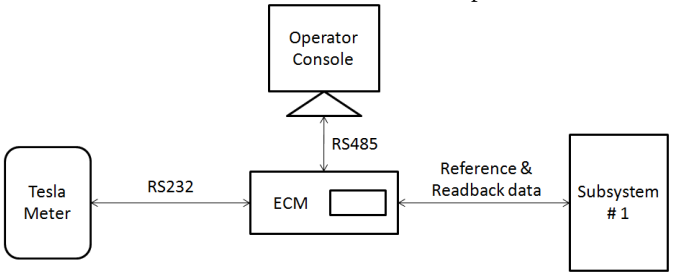


Fig. 2: Communication between Tesla Meter and Operator Console on RS485

II. Communication Protocol

2.1 Protocol for Tesla Meter

Tesla Meter receives a single character 'F' as a command and transmits 'Magnetic field data' as response. The ECM transmits 'F' and receives 'F $D_0D_1D_2D_3D_4D_5D_6D_7T$', Dn represents nth character.

2.2 Protocol for Operator Console

Protocol for the communication between ECM and Operator Console is designed in the form of Command –Acknowledge-Response frames. The ECM receives Command frame and transmits Acknowledgement and Response frames in a predefined format as shown in Fig. 3, 4 and 5. The acknowledgement can be positive or negative depending upon the checksum in the Command frame.

Start Delimiter	Destination Address	Source Address	Command Type	No. of parameters	Data Field	Checksum	End Delimiter	\r\n

Fig. 3: Command Frame

Start Delimiter	Destination Address	Source Addres	Frame Type	Last frame received	Acknowledgement	End Delimiter	\r\n

Fig. 4: Acknowledgement Frame

Start Delimiter	Destination Address	Source Addres	Readback	Device Value	Data Field	End Delimiter	\r\n

Fig. 5: Response Frame

III. Controller and Conversion Unit

For the two serial communication interfaces the design is divided into three modules:

1. Controller for communication over RS232.
2. Controller for communication over RS485.
3. Inter protocol conversion unit

Other than the protocol conversion unit both the controllers have one transmitter and one receiver control logic. The modules communicate at the baud rate of 19.2 kbps.

3.1 RS232 Transmitter

RS232 Transmitter transmits 'F' every tens of milliseconds to the Tesla Meter. A counter is implemented to define delays between two 'F' characters. An internal register is used to store 'F' and shift registers to transmit 'F'.

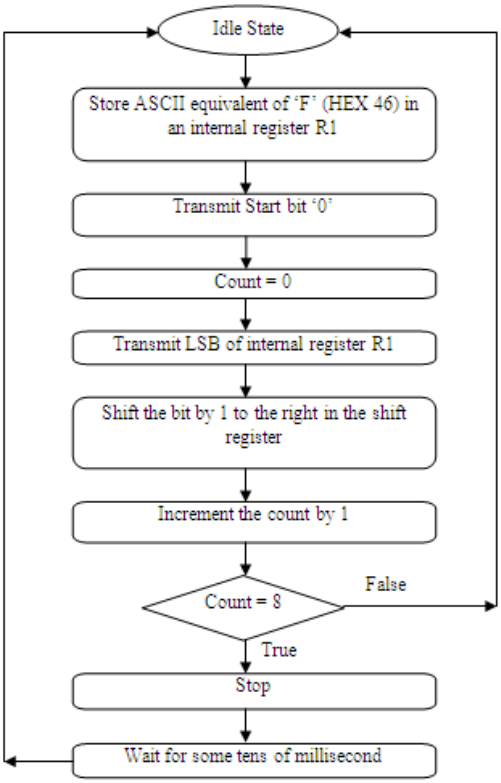
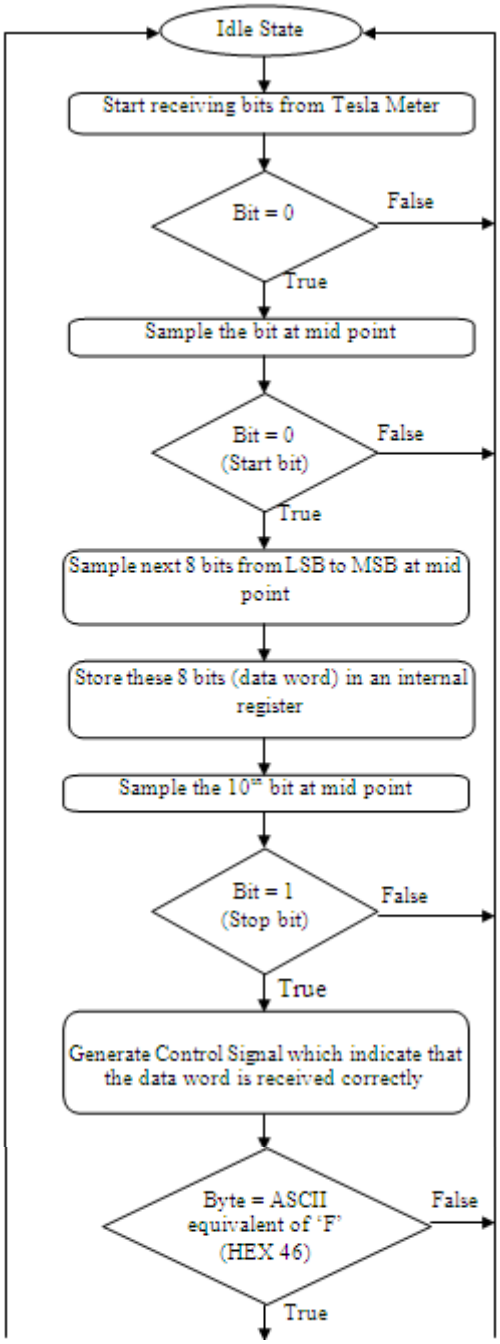


Fig. 6: Flow chart of RS232 Transmitter

3.2 RS232 Receiver

RS232 Receiver receives the serial data from the Tesla Meter. The data consists of 11 characters which are received one by one. A character reception is initiated by the bit transition from logic '1' to '0' on idle line (logic '1'). This logic '0' is a start bit which is sampled in the mid of the bit duration to validate its logic states. On validation the next 8 bits from LSB to MSB are stored in an internal register followed by a stop bit at logic '1'. All the data bits are sampled at the mid of the bit duration. On receiving the correct stop bit, a 'Character Valid' signal is generated otherwise the character is discarded. The procedure is repeated for complete data from the Tesla Meter and whole data is stored in an internal register. On storing the data, a 'data valid' signal is generated which is used by the protocol conversion unit to move from one unit to another.



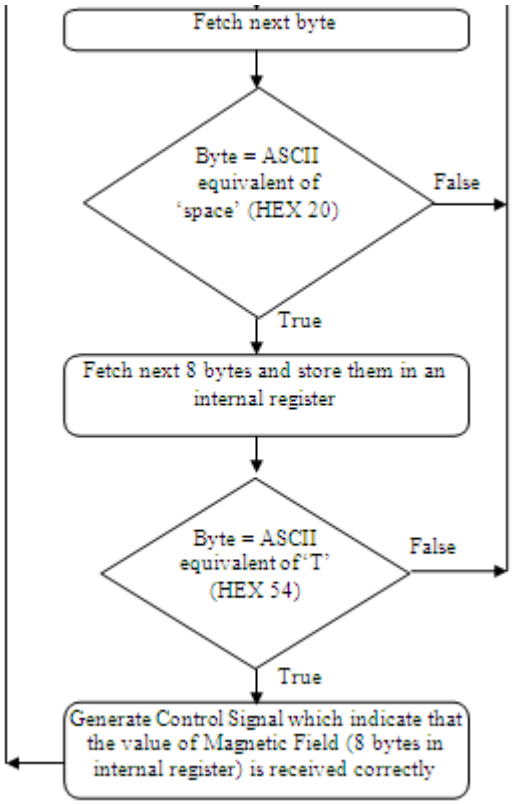
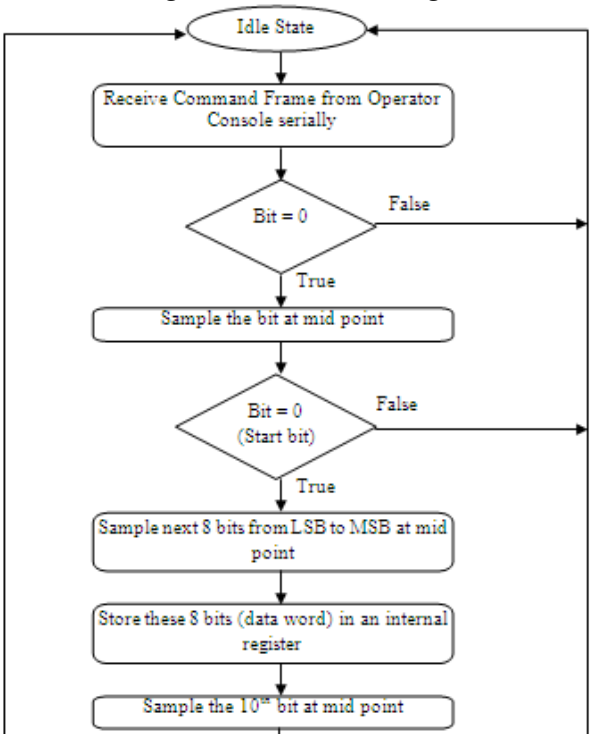
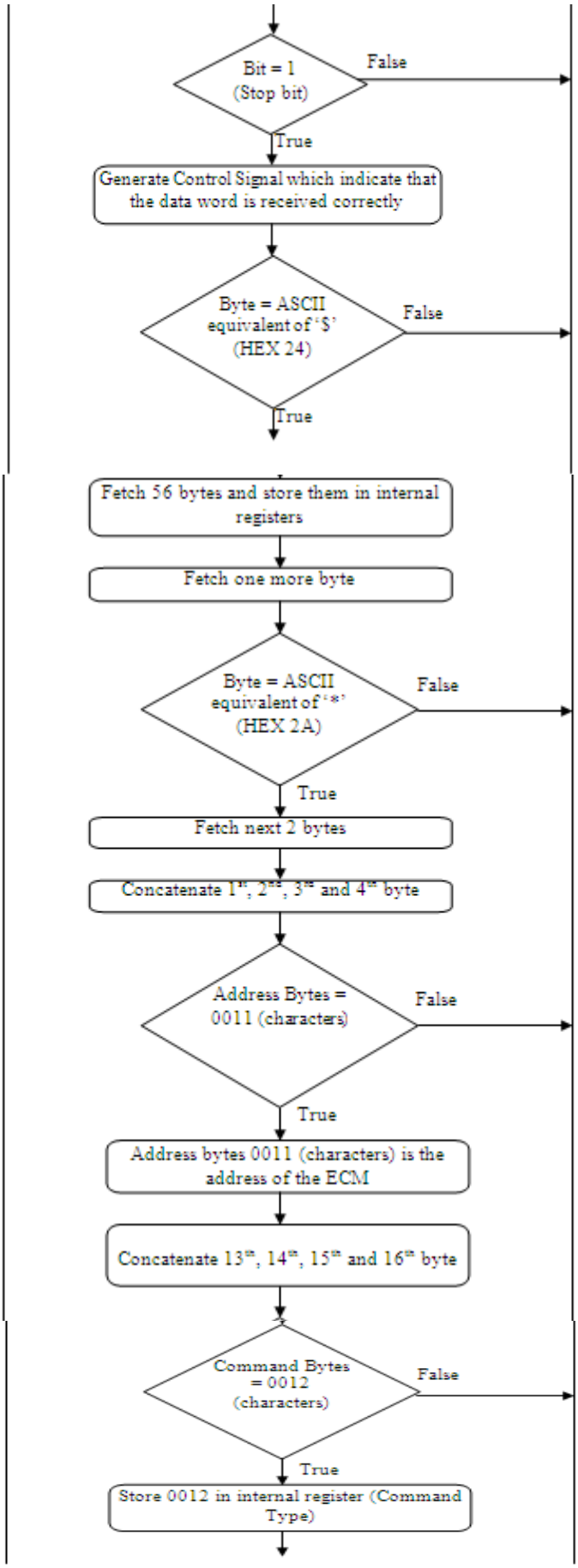


Fig. 7: Flow Chart of RS232 Receiver

3.3 RS485 Receiver

On receiving the characters in the Command frame between Start Delimiter ($) and End Delimiter (*), the ASCII equivalent of HEX character is converted into the corresponding HEX value. We validate the start of frame with Start Delimiter '$' and terminate the reception on getting End Delimiter (*). The storage is done on the verification of the destination address of the received frame. Frame integrity is done by checksum calculation which is last 4 nibbles of sum of all the data values between '$' and '*' in hex format. The 'Command type' value is also extracted from the command frame for the RS485 Transmitter. This all is done with the synchronization on baud rate clock generated inside the design.




Bit = 1
(Stop bit)
False
True
Generate Control Signal which indicate that the data word is received correctly
Byte = ASCII equivalent of '$'
(HEX 24)
False
True
Fetch 56 bytes and store them in internal registers
Fetch one more byte
Byte = ASCII equivalent of '*'
(HEX 2A)
False
True
Fetch next 2 bytes
Concatenate 1st, 2nd, 3rd and 4th byte
Address Bytes = 0011 (characters)
False
True
Address bytes 0011 (characters) is the address of the ECM
Concatenate 13th, 14th, 15th and 16th byte
Command Bytes = 0012 (characters)
False
True
Store 0012 in internal register (Command Type)

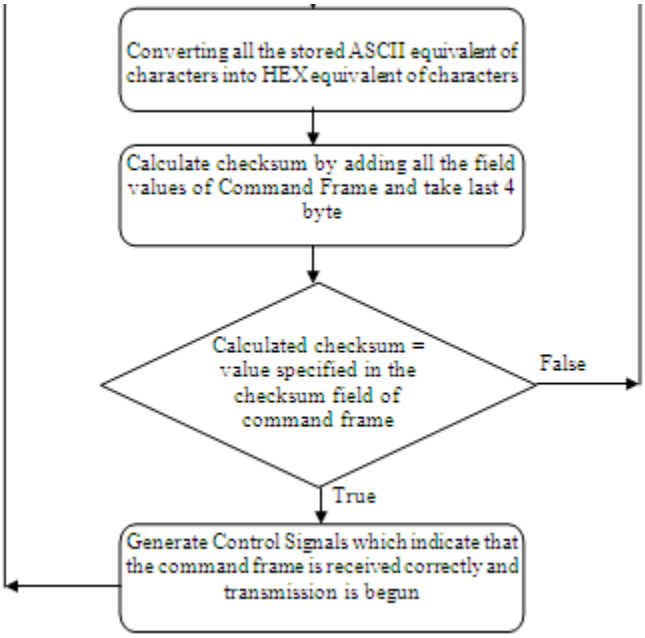


Fig. 8: Flow Chart of RS485 Receiver

3.4 RS485 Transmitter

RS485 Transmitter has two important signals, the 'data enable' and 'Serial data out' signals. The RS485 is a multi-drop interface in which many slaves can communicate with each other over a single bus. All slaves can receive data simultaneously but only one slave can transmit data and all other must drive their transmitter output in high impedance state. A 'data enable' is generated within the module which is used to enable different driver ICs whenever ECM is required to send the data over RS485 interface. An Acknowledgement frame is sent when Command frame received is either correct or incorrect and a Response frame is sent when Command frame received is correct. The Acknowledgement and Response frames are stored in an internal register array. The Acknowledgement frame consists of various fields in which the address of the source and destination, type of command and type of acknowledgement are given. The Response frame contains address of the source and destination along with data from Tesla Meter and subsystem's status information.

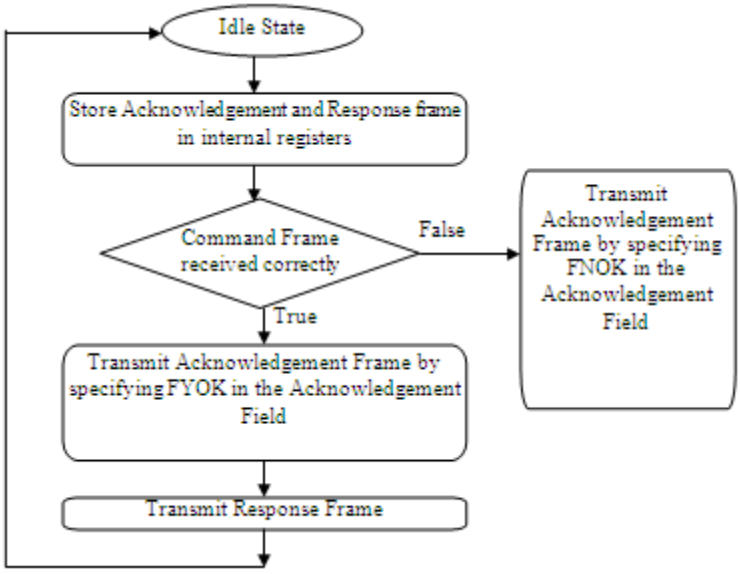


Fig. 9: Flow Chart of RS485 Transmitter

4. RS232 to RS485 Protocol Converter

The Protocol Converter is fed by the RS232 Controller which inputs the Magnetic Field value. The value is written into the temporary buffer of RS485 transmitter continuously till transmission from the RS485 Transmitter is not active.

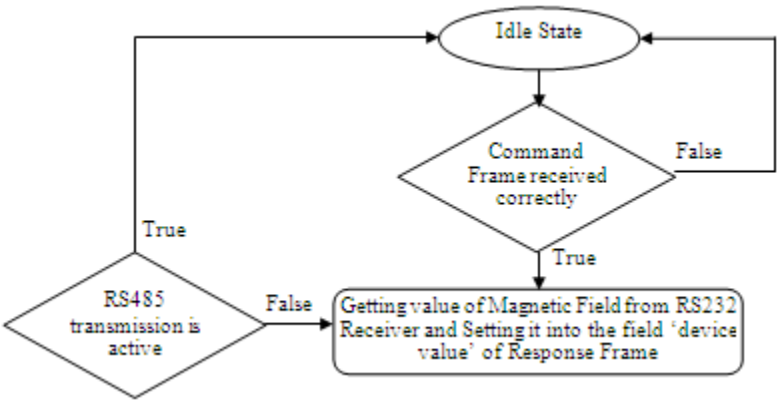


Fig. 10: Flow Chart of RS232 to RS485 Protocol Converter

5. Circular Buffer

A circular buffer is defined as a data structure which has a fixed-size, single buffer connected end-to-end. If the buffer is filled then the new data is written from the beginning and the old value has been overwritten.

A circular buffer of a LIFO type is designed in which a clock of few tens of milliseconds is used to write the data. The clock time period and number of locations decide the duration of data storage. On the positive edge of the clock the data is written and on the negative edge of the clock the data is read. The control of Read or Write operation is done by the generation of the read and write signal internally. Whenever Operator Console sends a read data command, a read signal is generated which reads the whole buffer starting from the current data location to last. Once 'read' is complete, the buffer goes into write state as shown in the simulation waveform (Fig. 16).

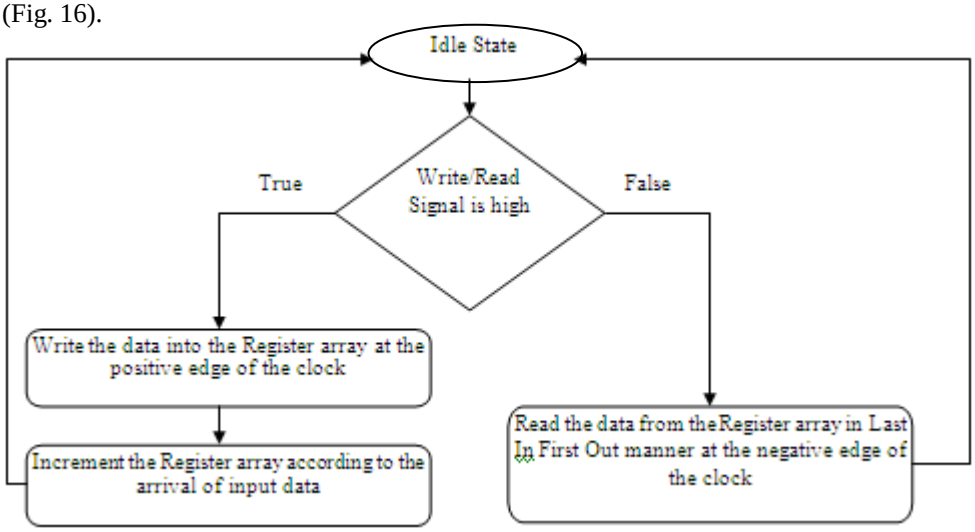


Fig. 11: Flow Chart of Circular Buffer

IV. Results

Table 1: Theoretical, Software Simulation and Practical Results

	Theoretical Value (duration)	Post Route Simulation Value (duration)	Practical Value (duration)
Appearance of ASCII 'F'	26.70ms	26.70ms	26.40ms
Acknowledgement Frame of 240 bits	12.50ms	12.57ms	12.56ms
Response Frame of 280 bits	14.58ms	14.76ms	14.64ms
Acknowledgement &Response Frame of 520 bits	27.08ms	28.53ms	29.12ms

The Post route simulation waveforms are given in Fig. 12 to 16:

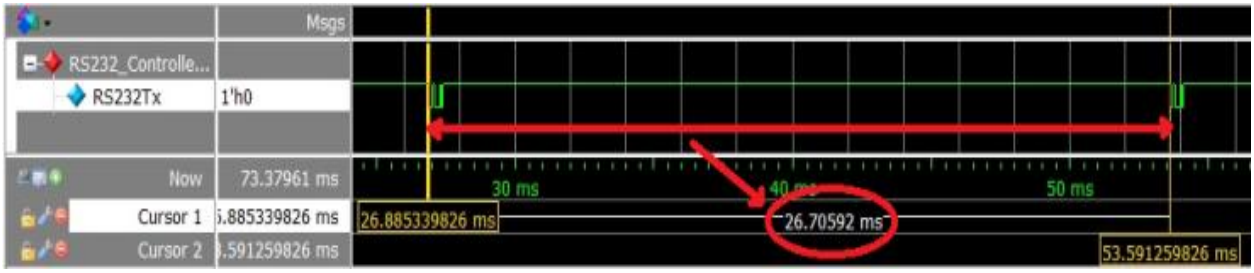


Fig. 12: Appearance of ASCII 'F' in every 26.70ms

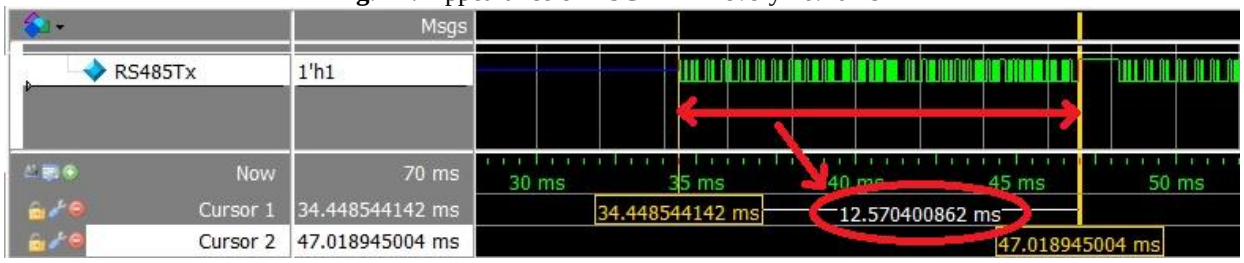


Fig. 13: Acknowledgement Frame transmits in 12.57ms

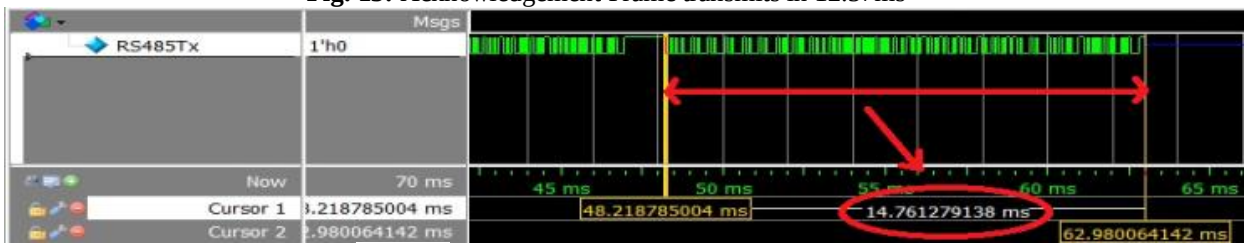


Fig. 14: Response Frame transmits in 14.76ms

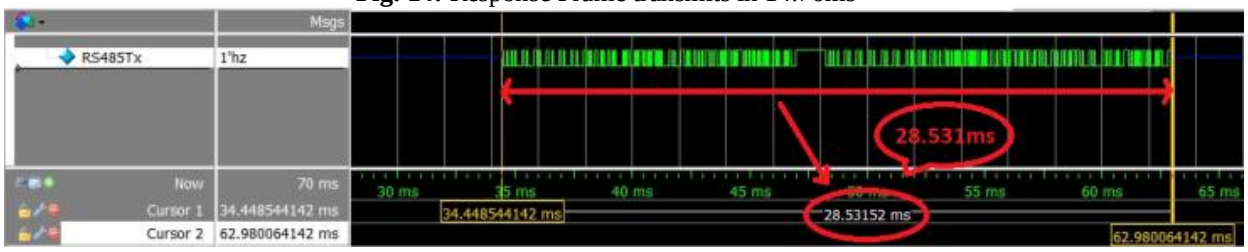


Fig. 15: Acknowledgement Frame followed by Response Frame in 28.53ms.

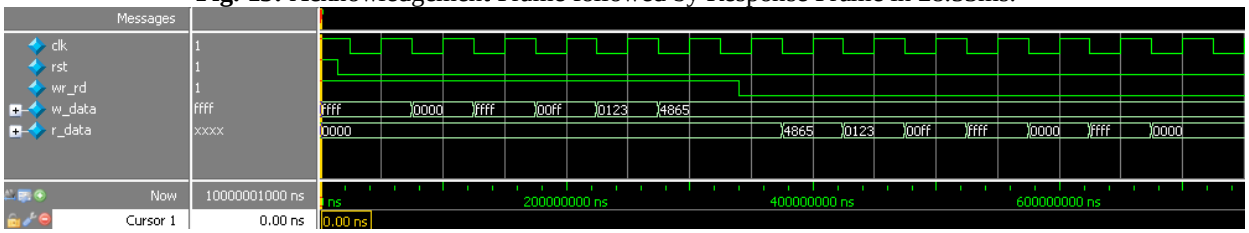


Fig. 16: Functional simulation of Circular Buffer

The Oscilloscope waveforms are given in Fig. 17 to 20.

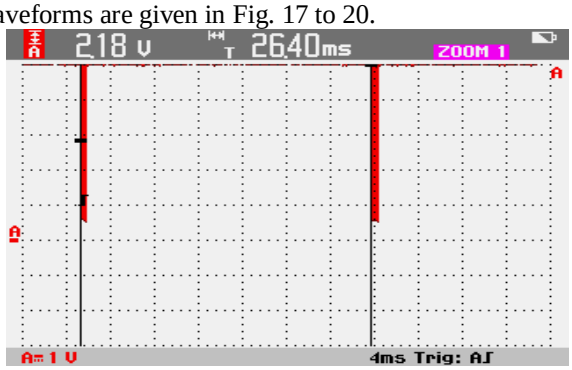


Fig. 17: Appearance of ASCII 'F' in every 26.40ms

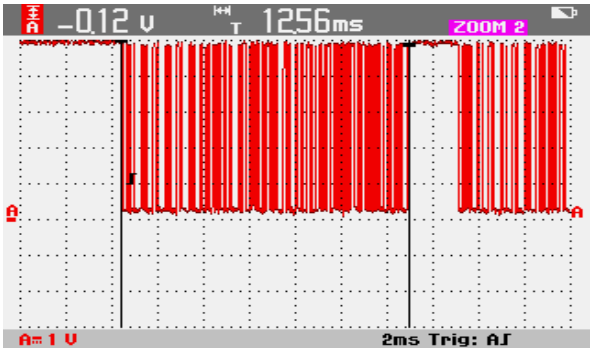


Fig. 18: Acknowledgement Frame transmits in 12.56ms

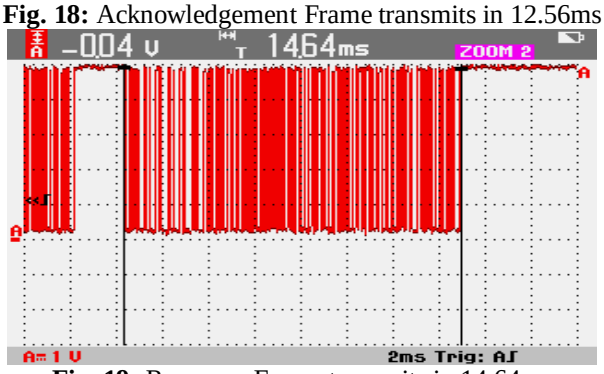


Fig. 19: Response Frame transmits in 14.64ms

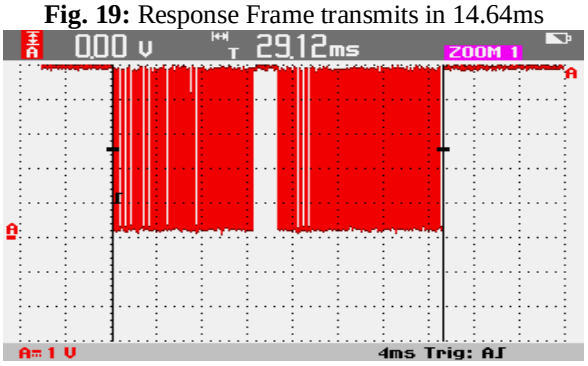


Fig. 20: Acknowledgement Frame followed by Response Frame in 29.12ms

Analysis of the results is done by the comparison of theoretical, simulated and actual values. The Table 1 shows the various numbers with some differences. The reason for the variation between theoretical and simulated values is due to difference in the baud rate clock timings. The theoretical value of the Baud Clock is 19.20 kHz and simulation value is 19.17 kHz so that the duration of 1 bit of data would be 52.083µs and 52.160µs respectively. Other reasons can be propagation delays and measurement error. The actual results are measured with time base in milliseconds. The measurement of microseconds are done with time base of few milliseconds which itself can cause some deviation of results from simulated values. Although these deviations are observed, the behaviour and functionality of the communication is not affected. The designed system is successful in all the aspects of implementation

V. Conclusion

RS232 & RS485 controllers are designed, simulated and implemented in FPGA with the help of Verilog HDL. The digital design for controllers is synthesized in terms of logic resource of the Spartan FPGA. The modules are designed to work at a baud rate of 19.2 kbps. This is verified through implementation and controllers are working successfully with Tesla Meter and Operator Console. It is noted that for complete communication at RS485 interface, it takes around 60 ms from reception of command to the transmission of response. This implies that data from 15 subsystems can be updated at this baud rate in 1s. Thus it may be concluded that this protocol conversion is useful for Microtron like systems where data update rate is of the order of a second and number of sub-systems are less than 15. The circular buffer is also designed and simulated using ModelSim XE & Xilinx ISE 8.2i. The implementation part will be completed in the near future.

Reference

[1]. Puneet Maheshwari, Yogendra Sheth, Pravin Fatnani, C.P. Navathe. RRCAT, FPGA based Control System Hardware for Microtron, DAE-BRNS Indian Particle Accelerator Conference, ©2013, Kolkata, West Bengal, India.
[2]. Jan Axelson, Serial port Complete COM ports, USB Virtual COM Ports, and Ports for Embedded Systems (Lakeview Research LLC, 5310 Chinook Ln., Madison WI 53704, 2nd edition, ©2007, Stockton, California).
[3]. Clive "Max" Maxfield, FPGAs World Class Designs (Newnes, 1st edition, ©2009, Burlington, United States of America).
[4]. Samir Palnitkar, Verilog HDL A guide to Digital Design and Synthesis (SunSoft Press, 2nd edition, ©2003, New Delhi, India).
[5]. Pong P. Chu, FPGA Pototyping by Verilog Examples Xilinx Spartan™-3 Version (Wiley A John Wiley & Sons, Inc., Publication, 1st edition, ©2008, Hoboken, New Jersey, United States of America).
[6]. M. Morris Mano, Michael D. Ciletti, Digital Design with an Introduction to the Verilog HDL (Pearson, 5th edition, ©2013, New Jersey, United States of America).
[7]. http://www.asciitable.com/