A HARDWARE IMPLEMENTATION OF EIA 709.1 CONTROL NETWORKING STANDARD

Jeon Il Moon[*], Jung Sub Kim[**], Jong Bae Kim[*], Kye Young Lim[***], Byoung Wook Choi[****]

[*]*R&D Center of LG Industrial Systems CO., Ltd, Anyangshi, Korea,*
[**]*School of Electrical Engineering, Penn State University, U.S.A.*
[***]*Korea Polytechnic University, Kyonggido, Korea*
[****]*Dept. of Electrical Engineering, Seoul Nat'l Univ. of Technology, Seoul, Korea*

Abstract: This paper presents a solution for the hardware implementation of EIA-709.1 Control Networking Protocol. It is the basic protocol of LonWorks systems that is widely used for the building system and the sensor system. The EIA 709.1 protocol has been implemented at the hardware level from physical layer to network layer to reduce the computing load on the CPU. VHSIC hardware description language (HDL) has been used for the EIA-709.1 protocol. Other layers have been implemented using C programs on Intel 8051 processor. The EIA-709.1 protocol has been implemented using field programmable gate array (FPGA) technology and the commercial feasibility of the proposed solution has been performed through the communication test using the Neuron Chip of EIA-709.1 protocol. The designed EIA709.1 core is usable as one of the intellectual properties (IPs) and it is applicable to design System-on-a-Chip (SoC) for various industrial controllers.

Keywords: EIA-709.1, fieldbus, networks, protocols, embedded systems

1. INTRODUCTION

Industrial system manufacturers have usually integrated devices by using their own proprietary protocol and physical network. It has been historically difficult to support interoperability between digital control equipment from various manufactures. Therefore, standardization for both open network and protocol in industrial domain is inevitable.

The fieldbus is a digital serial communication network designed to exchange data in real-time between distributed controllers and equipment installed in the automation industry. In building such systems, a growing number of devices such as sensor, loop controller, Programmable Logic Controller (PLC), motor, valve, robot and microprocessor-based control systems are used to implement intelligent and distributed functions; these are connected to a fieldbus network. As the number of devices in a system grows and the functions of a system need to be more intelligent, these devices need to rapidly exchange increasing amounts of data among them. Point-to-point or direct connections are not considered suitable any more for systems composed of many devices because the number of cables is increasing proportional to the square of the number of devices. In order to solve this problem, various serial communication networks have been designed and implemented to provide reliable and efficient communication paths for data exchange among the system components; such networks are called fieldbuses (Schumny, 1998; Thomesse, 1998) A fieldbus is a type of real-time communication system based on a layered structure deduced from the seven layers Open System Interconnection (OSI) model (Zimmermann, 1980). Fieldbuses include Profibus, World FIP, Fieldbus Foundation, Controller Area Network (CAN) and LonWorks (Choi, *et al*., 2000; Lee, *et al*., 2004; Almedia, *et al*., 2002).

Most industrial devices readily incorporate the processor chip without a price burden because the cost of microprocessors has been down for consecutive years. Design engineers began to realize the necessity of open protocols for optimal communication performance in control systems. Control networks have a number of unique requirements that are different from data networks. The followings are some of these unique requirements.

- Frequent, reliable, secure communications between devices
- Short message format for the information being passed

- Peer-to-peer functionality for every device
- Chip price that enable small and low-cost devices

There are needs to address these control specific network requirements. In addition, there is the belief that a market standard for communications would provide interoperability between control devices from various manufacturers and empower the market to increase in both size and efficiency.

The LonWorks protocol was introduced as a solution in the domain of the control specific network (Echelon, 1999). A LonWorks network uses the LonWorks protocol, also known as the ANSI/EIA 709.1 Control Networking Standard, to accomplish these tasks that was invented by Echelon in 1988 (ANSI, 1988). The LonWorks protocol is a layered, packet-based, peer-to-peer communications protocol. To ensure the requirements of control systems, the protocol has the layered structure as recommended by International Standards Organization (ISO). This is a different feature from other fieldbus protocols. By tailoring the protocol of each layer of the Open System Interconnection (OSI) reference model in order to guarantee domain specific control performance, the LonWorks protocol provides a control-specific solution that guarantees reliability, performance, and robust communications required for control applications. The protocol is originally embedded in Neuron Chip which Echelon has published the LonWorks protocol and made it an open standard under the ANSI/EIA 709.1 Control Networking Standard. The protocol is, therefore, freely available to anyone.

The variety of services provided by the LonWorks protocol allows for enhanced reliability, security, and optimization of network resources.

In this paper we develop EIA-709.1 protocol by using VHDL and the C programming language. Of 7-layers of EIA-709.1, the physical layer, Medium Access Control (MAC) layer, and the data link layer are implemented at the hardware level. And then the functional verification is performed in Field Programmable Gate Array (FPGA). The software implemented in an 8032 microprocessor handles the protocol implemented in FPGA, and the experiment is performed with Neuron chip to verify the functionality and interoperability.

2. OUTLINE OF EIA-709.1 PROTOCOL

The seven layers of the ISO/OSI model, along with the corresponding services, are provided by the LonWorks protocol. It is not a requirement that any given protocol implement every layer of this model. A truly complete and fully scalable protocol – such as the LonWorks protocol – provides all the services described in this model.

The physical layer defines the raw bits over a communication channel. The LonWorks protocol is media-independent so that multiple physical layer protocols are supported according to the communication medium.

The link layer defines media access method and data encoding scheme to ensure efficient use of a single communications channel. The raw bits of the physical layer are broken up into data frames. The LonWorks protocol uses a unique media access control algorithm, called predictive p-persistent Carrier Sense Multiple Access (CSMA) protocol that has excellent performance characteristics even during the period of network overload, allowing a channel to operate with full capacity with minimum collision. The CSMA protocol is a listen-before-transmit scheme in which a device with a message to transmit first listens to the network. If no message traffic is detected, then the device will transmit its message after a calculated number of packet time slots.

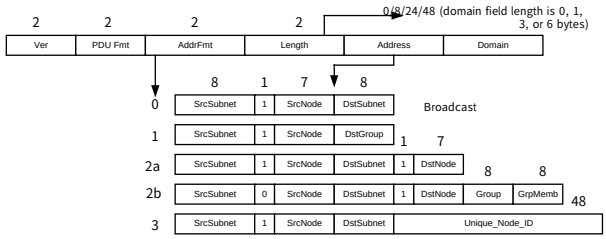

Fig. 1. EIA-709.1 Address Format

The link layer supports simple connection-less service. It is limited to Frame transferring, frame encoding, and error search function. However, the layer does not support the error restoration function. The address format of this protocol supports a total 5 modes, and the details are shown in Fig. 1. The format performs 2 broadcast modes, 2 subnets, node communication, and Unique Node ID communication.

The network layer deals with transmission of packets within a single domain, but does not support communication between domains. The network layer provides connection-less and unacknowledged service. On the contrary, it does not support message division and the reassembling function.

The transaction control sub-layer that is common in both transport layer and session layer makes transactions in order and senses duplicated message. The transport layer provides connection-less reliable transaction with either one or several nodes. Authentication to identify message sender is provided optionally. Transaction control sub-layer is designed only to perform that function. Therefore, the message from the transport layer and session layer can be authorized using all the addressing modes except broadcast. The session layer provides simple Request-Response mechanism approachable to remote server.

There are a total of 5 forms of communication services used here. First of all, Unacknowledged service does not require an acknowledge frame, and a

simple unacknowledged message that is one of this service does not need to sense the duplicated message. Just as acknowledged service requires an acknowledge frame, repeated service also requires an acknowledge frame, but instead, it completes the service after sending messages repeatedly as many as specified count. Request/Response service requires a response frame instead of an acknowledged frame. Finally, Authentication service is a service form to be used to avoid uncertainty in security.

Both Presentation layer and practical application layer provide all the general services to transmit and receive messages including Network Variables. In addition, the service related to maintenance such as management and security of network is performed at this layer.

3. PROBLEMS OF CONVENTIONAL METHOD

EIA-709.1 protocol was developed by Echelon. Implementation method widely-used till now is, as mentioned previously, as follows: to design hardware board using Neuron Chip of either Toshiba or Cypress, to compile basic form, firmware and application program using LonBuilder Developer's Workbench and Neuron C of Echelon, and then to load the Neuron Chip with the compiled result. Although EIA-709.1 is open protocol, due to dependency of both hardware and software on a particular company, several problems could happen.

First of all, the network configuration is possible only with the hardware referred to as Neuron Chip; therefore, other network configurations that Neuron Chip does not provide are not feasible. As other communication speed except the speed provided by Neuron Chip cannot be used, the efficiency of a network could not be improved. When some problem happens in network, it cannot be handled because debugging is impossible at chip level. Also, in the case of the communication using Neuron Chip and LonBuilder, due to the problem dealing with firmware, the duration between sending data packet and sending next data packet is relatively long compared with logical duration that can be implemented by EIA-709.1. This is not such a big problem in the normal state, but it could cause serious problems in the case that either the amount of packet gets increased or some error occurs on the network.

Another problem is to write an application program only by following the guideline that LonBuilder supports. In other words, Neuron C programming language, which is transformed from ANSI-C, must be used instead of the general C. In this case, software programming becomes tightly dependent on the LonBuilder development environment. When implementing various functions, we cannot use the functions that the development environment does not support. Therefore, application program has little flexibility and compatibility, and there is a limit in implementing program as well.

4. CONFIGURATION OF THE DEVELOPMENT SYSTEM

This system implemented EIA-709.1 protocol in hardware using VHDL, which is composed of a physical layer, MAC layer and link layer, and network layer. FPGA fitting was made, and design of firmware and hardware board was performed in order to verify function specification. The hardware board is not only for implementing the protocol, but also for designing the hall station board for an elevator system that adopts the protocol. That is why some other functions were added.

4.1 Implementation of FPGA

FPGA was designed by Logic design using VHDL (VHSIC Hardware Description Language; VHSIC stands for Very High Speed Integrated Circuit), synthesized using Design Compiler of Synopsis, and performed Logic simulation using Verilog-XL simulator of Cadence. Finally, FPGA P&R (Place & Route) and FPGA fitting were performed using Maxplus II Tool of Altera (Motorola, 1997; Altera, 1999; Douglas, 199).

This FPGA is composed of major 6 blocks such as MAC block, transmitting block, receiving block, timer block, interrupt block, and register block. The overall block diagram is shown in Fig. 3.

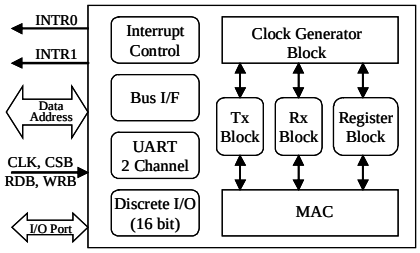

Fig. 3. Overall Block Diagram of EIA-709.1 protocol

4.2 MAC (Media Access Control) Block

This block is divided into several sub-blocks such as the blocks for variable communication boardrate setup, Differential Manchester Encoder/Decoder, digital filter for communication port, and Cyclic Redundancy Check (CRC) generation and detection. Predictive p-persistent CSMA algorithm is implemented in the MAC block as well (Fred, 1988).

In the case of communication through Twisted Pair cable, all of the communication data are not the form of NRZ, and the communication is always performed using Differential Manchester coding algorithm. Differential Manchester coding is a method to cause a transition only once at the starting point of 1 bit, and to keep the current value for 1 bit duration in

case that the value equals to 1, and to cause a transition once more at the time position of 1/2 bit in case of 0. The design algorithm of Differential Manchester Encoder/Decoder is shown in Fig. 4.

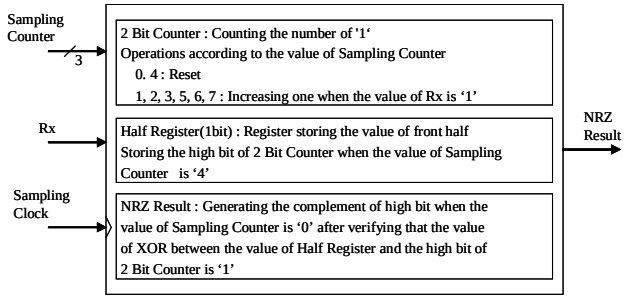

Fig. 4. Manchester Encoder and Decoder Algorithm

4.3 Transmitting Block

The Transmitting block is generated as state transition block according to generation of frame to transmit, generation of control signal, and generation signal of each frame. The diagram of the transmitting block is described in Fig. 5.

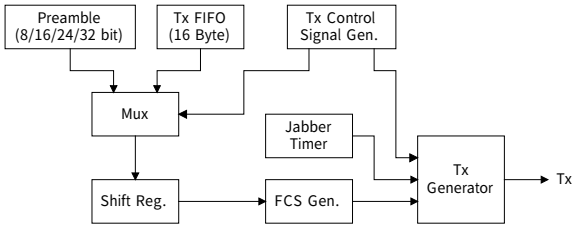

Fig. 5. Block Diagram of the Transmitting Block

The transmitting order starts from the Line Idle state. While in the Line Idle state, Beta-1 timer operates automatically and checks the Beta-1 time. When Beta-1 timer is operating, expired or ready to operate, if the Transmit Start Enable signal get activated from outside, first the transmitting block waits for the Beta-1 timer to expire. Afterwards, while waiting for the Beta-2 time of Priority Slot and Random Slot, if the line state keeps being inactivated, the block starts to transmit a packet.

Preamble is loaded on a shift register as many times as there are Preamble Bytes. Preamble consists of BitSync and ByteSync; BitSync has the value of '1' and ByteSync has the value of '0'. First, the value of '1' is transmitted to all the bits of every shift register, and then the final Preamble would be loaded with the value of '0' on the last bit.

After completing Preamble transmission, a packet is transmitted in order from FIFO to shift register one by one. While transmitted from FIFO one by one, the number of transmission data length written before transmitting is decreased. If the number is '0' after all packets are transmitted, the normal operation is performed, and otherwise Data Length Mismatch Error takes place. After transmitting all values of FIFO, it sends out 16 bits CRC by performing the operation of CRC generator. It also checks by Jabber Timer if a transmission line is occupied abnormally. After completing CRC transmission, it notices the end of a packet by remaining in the state of Code Violation for interval time of 2.5 bits.

4.4 Receiving Block

The Receiving block consists of receiving control block, FIFO of 32 bytes, shift register, address recognition block, and etc. The block diagram is described in Fig. 6.

As the Differential Manchester decoder recognizes the new packet, it transfers serial data to the shift register. When 16 bit data is stored in the shift register, upper 8 bits are loaded into FIFO. At the same time, it recognizes the address field in the shift register and decides if the data is to be received. In case that address does not match, data is not loaded in FIFO, and CRC in the shift register is checked to determine whether CRC error occurs. Finally, it is used to control generation of random numbers.

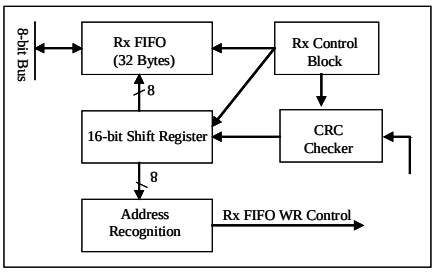

Fig. 6. Block Diagram of the Receiving Block

The Receiving control block is used to control FIFO of the receiving block, shift register, and address recognition block.

4.5 Interrupt Control Block

Interrupt control block supports the interrupt handling by Interrupt Pin, and selectively deals with the interrupt based on the priorities of eight interrupt sources. Interrupt source supports Edge Trigger Mode only. The overall block diagram is described in Fig. 7.

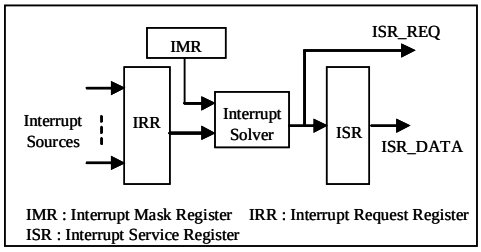

Fig. 7. Interrupt Control Block Diagram

4.6 Other Blocks

As for other implementation, Loop-back mode is supported for the convenience of debugging, and 16 input ports and 16 output ports are provided for hall station control of an elevator system. The implementation provides software reset function and has an output port to show current communication status.

4.7 Testing Board

The test board was designed with 8032 for CPU, ROM (32K bytes) and RAM (32K bytes). The board was also implemented with RS-232 channel for debugging, and FPGA was implemented using EPF10K100GC503-4 chip of Altera. The board performs communication using the designed communication transceiver, and consists of other blocks for elevator hall station. The overall block diagram is shown in Fig. 8.

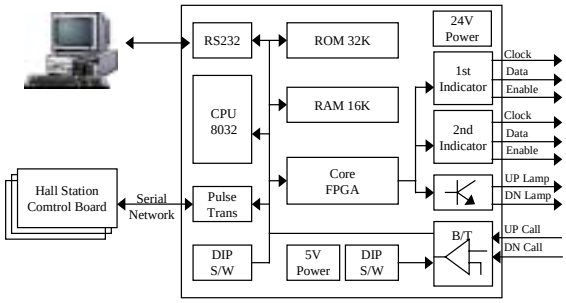

Fig. 8. The Board with FPGA implementing EIA 709.1

4.8 Firmware

The firmware program for i8051 using C language was designed. It is compiled using IAR 8051 C compiler of IAR Systems Co., Ltd. The test program is performed to display the transmitting/receiving data on the display unit using RS-232 port. We also debugged the firmware with a commercial emulator and a ROM-based method.

5. EXPERIMENT OF EIA 709.1

5.1 Test System Configuration

The test environment, as shown in Fig. 9, consists of one hall board equipped with FPGA that is implemented with EIA-709.1 protocol, one original hall board, and controller board that manages overall data. Besides them, the test system consists of other test components such as button or lantern so that the transmission and receiving of packets were verified. Fig. 9 shows the detailed picture of the hall board equipped with FPGA implemented with EIA-709.1 protocol. The clock for CPU and FPGA is 14.7456MHz and 20MHz, respectively. The maximum input frequency shown in the specification sheet of the Neuron Chip is 10 MHz. Therefore, the frequency of 20 MHz for FPGA is suitable for EIA-709.1 protocol to be implemented. The communication speed is used with 78.125Kbps fixed. An original hardware with the Neuron Chip implemented is used to test and verify interoperability between the original hardware and a new hardware implemented with the newly designed FPGA.

Fig. 9a. Test System Configuration

Fig. 9b. The Board with FPGA implemented with EIA-709.1 Protocol

5.2 Verification Results

First of all, the result of simulation was verified using SimWaves and Verilog-XL of Cadence. Fig. 10 shows a communication packet that the transmitting block generates in case of Unrepeated service communication of the group address method. When the total of 10 data is loaded and then the transmission starts, Fig. 10 shows that the packets for each state are generated and sent out.

Fig. 11 shows the simulation waveforms of the receiving block. These are the waveforms at the time of receiving the second packet. The total of 9 data points, from the 11th received FIFO through the 19th received FIFO, are received and stored, and we can also see that the control signals corresponding to each state are generated.

The board test for the transmitting block was performed as the following. As key input is activated, packets are transmitted and then the original Neuron board receives the packets and changes the value of lantern of a hall board. On the contrary, the test for the receiving block was performed such that the received packets are displayed on the terminal using RS-232 port. Fig. 12 shows the oscilloscope waveforms measured from FPGA board when transmitting. Channel number 1 represents the signal

that packet is transmitting, and channel number 2 shows the data of the transmitting packet and represents that a normal operation has been performed.

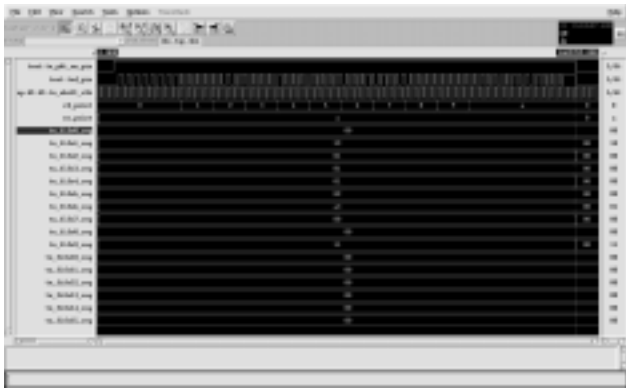

Fig. 10. Simulation Waveforms of Transmitting Block

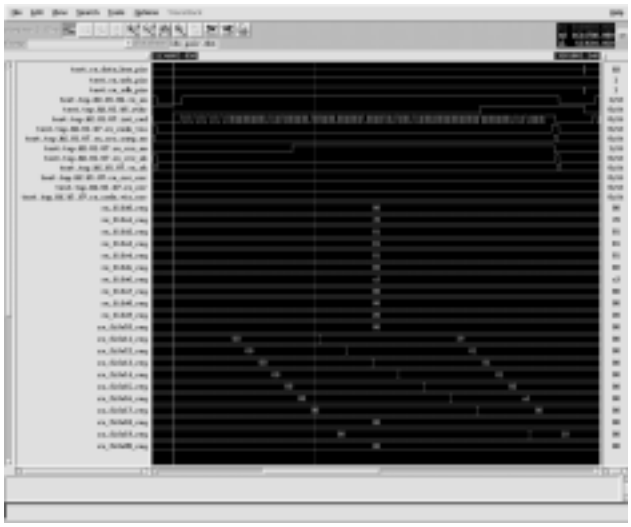

Fig. 11. Simulation Waveforms of Receiving Block

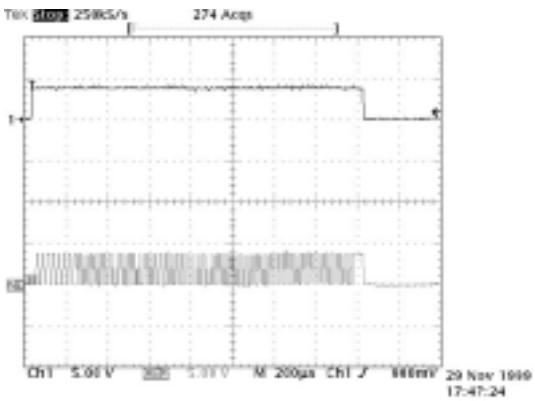

Fig. 12. Oscilloscope waveforms of Receiving Block

6. CONCLUSIONS

In this paper, EIA-709.1 Control Networking Protocol was implemented. We resolved several problems raised as the conventional methods based on original Neuron chip are implemented. First, in order to increase flexibility, the protocol from physical layer to network layer was implemented using VHDL in hardware level, and verified with FPGA. Other upper layers were handled in software. By doing this, it is possible to control lower layers directly so that we can effectively deal with communication errors such as network failure, packet error, and etc. Secondly, CPU can be widely selected to control communication speed or process freely compared to conventional method, and high effectiveness could be accomplished as well. Therefore, if the result of this study is adopted to implement the EIA-709.1 protocol, the flexibility can be increased. In addition, because any kind of commercial CPU can be used with the method proposed in this paper, it can be used as a more useful and convenient approach than conventional methods. As a hardware part is designed using VHDL and implemented as IP, it is possible to apply this result in designing any application product using EIA 709.1 protocol such as SoC chip design of a distributed control system. The further work is to design optimal hardware and advanced firmware program. In addition, in case of using general processor, it is essential to build the integrated development environment, and especially, the study on application layer and network maintenance should be performed.

7. REFERENCES

Almedia, L., E. Tovar, J.A.G. Fonseca and F. Vasques (2002). Schedulability Analysis of real-time traffic in WorldFIP networks: an integrated approach, *IEEE Transactions on Industrial Electronics*, **49**, pp. 1165-1174.

Altera (1999), *Altera Device Data Book.*

Choi, B. W. and J. S. Kim, C. H. Lee, J. B. Kim, and K. Y. Lim (2000). Implementation of the field bus system which used EIA-709.1 Control Network Protocol, *Journal of ICASE*, **7**, pp. 594-601.

Douglas L. Perry (1993). *VHDL* Second Edition, R. R. Donnelley & Sons Company

Echelon (1999). *Introduction to the LonWorks System*

Fred Halsall (1988). *Data Communications, Computer Networks and OSI*. 2nd edition. Addison Wesley, pp. 88 ~91.

Lee, K. C., S. Lee and H. H. Lee (2004). Implementation and PID tunning of network-based control systems via Profibus polling network, *Computer Standards & Interfaces***, 26**, pp. 229-240.

Motorola (1997). *LonWorks Technology Device Data*

Schumny, H. (1998). Fieldbuses in measurement and Control, *Computer Standards & Interfaces*, **19,** pp. 295-304.

Thomesse, J. P. (1998). A Review of the Fieldbuses, *Annual Reviews in Control*, **22,** pp. 35-45.

Zimmermann, H. (1980). OS1 reference model. The IS0 model of architecture for open system interconnection, *IEEE Trans. COM* **28**, pp. 425-432.