International Journal of Engineering & Technology
CENTRE OF PROFESSIONAL RESEARCH PUBLICATIONS
ISSN 2049-3444

International Journal of Engineering and Technology Volume 2 No. 10, October, 2012

Design and VLSI Implementation of HDLC Controller

Savita Yadav, Anuradha Yadav, Nishant Tripathi, *Sanjay Singh
Department of Electronics Design & Technology, Gorakhpur UP India
*Department of Electronics & Communication Engg, Gorakhpur UP India

ABSTRACT

The HDLC Controller MEGACELL is a high performance module for the bit oriented, switched, non-switched packet transmission module. The controller fulfills the specifications according to ITU Q.921, X.25 Level 2 recommendation. It supports half duplex and full duplex communication lines, point-to-point and multipoint channels. The Controller is designed to permit synchronous, code transparent data transmission. The control information is always in the same position and specific bit patterns used for control differ dramatically from those representing data, which reduces the chances of errors. The data stream and transmission rate is controlled from the network node. This eliminates additional synchronization and buffering of the data at the network interface. Some common applications include terminal-to-terminal, terminal to CPU, satellite communication, packet switching and other high-speed data links. In system, which require expensive cabling, and interconnection hardware? So this core can be used to simplify interfacing by going serially, thereby reducing interconnects hardware costs. Since it is speed independent, reducing interconnect hardware could become an important hardware.

Keywords: *HDLC, VLSI, CPU*

1. INTRODUCTION

HDLC (High-level Data Link Control) is a group of protocols for transmitting (synchronous) data (Packets) between (Point-to-Point) nodes. In HDLC, data is organized into a frame. HDLC protocol resides with Layer 2 of the OSI model, the data link layer. It is an efficient layer2 protocol standardized by ISO for point-to-point and multipoint data links. HDLC provides minimal overhead to ensure flow control, error control, detection and recovery for serial transmission[1].

The HDLC frame is synchronous and therefore relies on the physical layer to provide method of clocking and synchronizing the transmission and reception of frames. The frames are separated by HDLC flag sequences that are transmitted between each frame and whenever there is no data to be transmitted. To inform the receiving station that a new packet is arriving and synchronizes the receive clock with the transmitted clock a specific bit pattern is added at the front and the back of the packet. The header of the packet contains an HDLC address and an HDLC control field. The specific bit pattern is used to affix with the packet in the case of HDLC Controller is 01111110. The length of the address field is normally 0, 8 or 16 bits in length. In many cases the address field is typically just a single byte, but an Extended Address (EA) bit may be used allowing for multi-byte addresses. A one residing in the LSB bit indicates [the end of the field] that the length of the address field will be 8 bits long. A zero in this bit location [now the first byte of a multi-byte field] indicates the continuation of the field [adding 8 additional bits]. The Control field is 8 or 16 bits and defines the frame type; Control or Data to guarantee that a flag does not appear inadvertently anywhere else in the frame; HDLC uses a process called bit stuffing. Every time the user wants to send a bit sequence having more than 5 consecutive 1s, it inserts (stuffs) one redundant 0after the fifth 1. The trailer is found at the end of the frame, and contains a cyclic Redundancy Check (CRC), which detects any errors that may occur during transmission[3]. A CRC value is generated by a calculation that is performed at the source device. The destination device compares this value to its own calculation to determine whether errors occurred during transmission. First, the source device performs a predetermined set of calculations over the contents of the packet to be sent. Then, the source places the calculated value in the packet and sends the packet to the destination. The destination performs the same predetermined set of calculations over the contents of the packet and then compares its computed value with that contained in the packet. If the values are equal, the packet is considered valid. If the values are unequal, the packet contains errors and is discarded. The receiver can be configured into transparent mode, effectively disabling the HDLC protocol functions. In normal HDLC protocol made, all received frames are presented to the host on the output register. A status register is provided which can be used to monitor the status of the receiver channel, and indicates if the packet currently being received includes any errors[2][4].

2. LITERATURE SURVEY OF HDLC CONTROLLER

The CDAC HDLC controller operates at the data link layer of the OSI Model. Hence the main focus of the

survey is to understand the data link layer and develop a protocol which can offer its services to the layer above it i.e. is the network layer and the layer below it i.e. the physical layer. The main function of this protocol controller is to perform a number of separate activities like physical addressing, to check for errors, flow control etc.

The layered concept of networking was developed to accommodate changes in technology. Each layer of a specific network model may be responsible for a different function of the network. Each layer will pass information up and down to the next subsequent layer as data is processed[5].

Open Systems Interconnection (OSI) model is a reference model developed by ISO (International Organization for Standardization) in 1984, as a conceptual framework of standards for communication in the network across different equipment and applications by different vendors. It is now considered the primary architectural model for inter-computing and internetworking communications. Most of the network communication protocols used today have a structure based on the OSI model. The OSI model defines the communications process into 7 layers, which divides the tasks involved with moving information between networked computers into seven smaller, more manageable task groups. A task or group of tasks is then assigned to each of the seven OSI layers. Each layer is reasonably self-contained so that the tasks assigned to each layer can be implemented independently. This enables the solutions offered by one layer to be updated without adversely affecting the other layers. [1]

A protocol in the context of networking is essentially a system of rules which define how data is transferred from a source to a destination, at different levels of abstraction from the physical level of electrical pulses carried via cables or wireless, or fiber-optical signals, to the more abstract level of messages sent by an application such as email.

In order for computers with different hardware and operating systems to be able to communicate effectively over a network or an internet, it is clearly important for there to be a uniform set of protocols and standards which the communicating systems and applications will conform to. This in turn suggests a need for organizations with commonly recognized authority that will develop, define and publish standards in different domains.

HDLC [High-level Data Link Control] is a group of protocols for transmitting [synchronous] data [Packets] between [Point-to-Point] nodes. In HDLC, data is organized into a frame. HDLC protocol resides with Layer 2 of the OSI model, the datalink layer. HDLC uses zero insertion/deletion process [bit stuffing] to ensure that the bit pattern of the delimiter flag does not occur in the fields between flags. The HDLC frame is synchronous and therefore relies on the physical layer to provide method of clocking and synchronizing the transmission and reception of frames. [2]

3. DESIGN AND IMPLEMENTATION

The whole design is organized as a collection of two sections that work together to efficiently perform the operation as shown in fig.1.

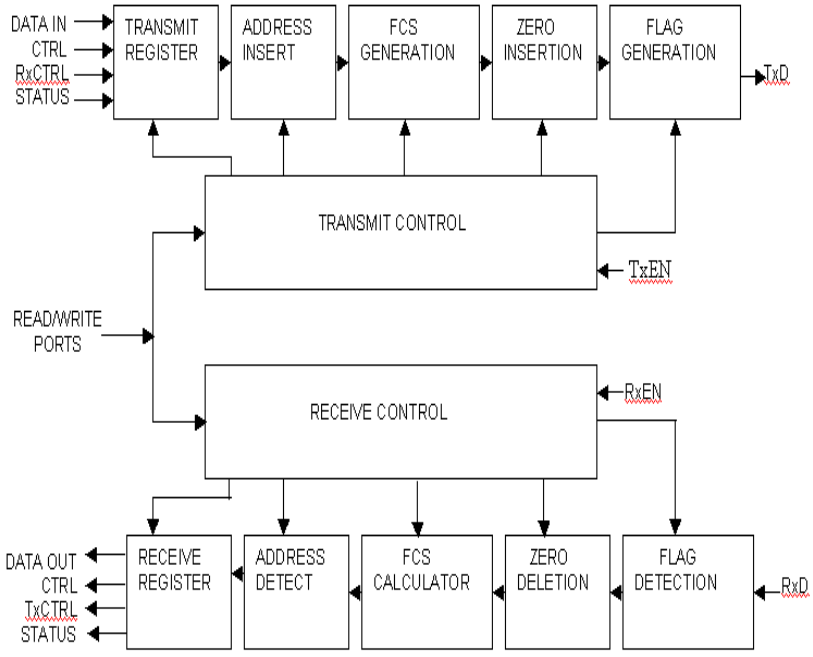


Fig.1 Basic block diagram of HDLC Controller

3.1 Transmitter Section

The Transmit Data Interface provides a byte wide interface between the transmission host and the HDLC Protocol core. Transmit data is loaded into the core on the rising edge of clk when the write strobe input asserted. The start and end bytes of a transmitted HDLC frame are indicated by asserting the appropriate signals with the same timing as the data bytes. The HDLC core will, on receipt of the first byte of a new packet, issue the appropriate flag sequence and transmit the frame data calculating the FCS. When the last byte of the frame is seen, the FCS is transmitted along the closing flag. Extra zeroes are inserted into the Bit stream to avoid transmission of the control flag sequence within the frame data. The transmit data is available on the TxD pin with appropriate setup to be sampled be clk. If TxEN is reasserted, the transmit pipeline is stalled, and the TxD pin is tristated. A transmit control register is provided which can enable or disable the channel, select transparent mode where the HDLC protocol is disabled, and specify the HDLC core action on transmit FIFO under runs. In addition, it is possible to force the transmission of the HDLC Abort sequence. This will cause the currently transmitted frame to be discarded. The transmit core can be configured to automatically restart after an abort, with the next frame, or to remain stalled until the host microprocessor cleared the abort or transmit FIFO under run condition[6][7].

3.2 Receiver Section

Receiver accepts a bit stream on port RxD. The data is latched on the rising edge of clk under the control of the enable input RxEN. The flag detection block stream for the flag sequence in order to determine the frame boundaries. Any stuffed zeroes are detected and removed and the FCS is calculated and checked. Frame data is placed on the receiver data interface and made available to the host. In addition, flag information is passed over indicating the start and end bytes of the HDLC frame as well as showing any error condition which may have been detected during receipt of the frame. The receiver can be configured into transparent mode, effectively disabling the HDLC protocol functions. In normal HDLC protocol made, all received frames are presented to the host on the output register. A status register is provided which can be used to monitor the status of the receiver channel, and indicates if the packet currently being received includes any errors[6][7].

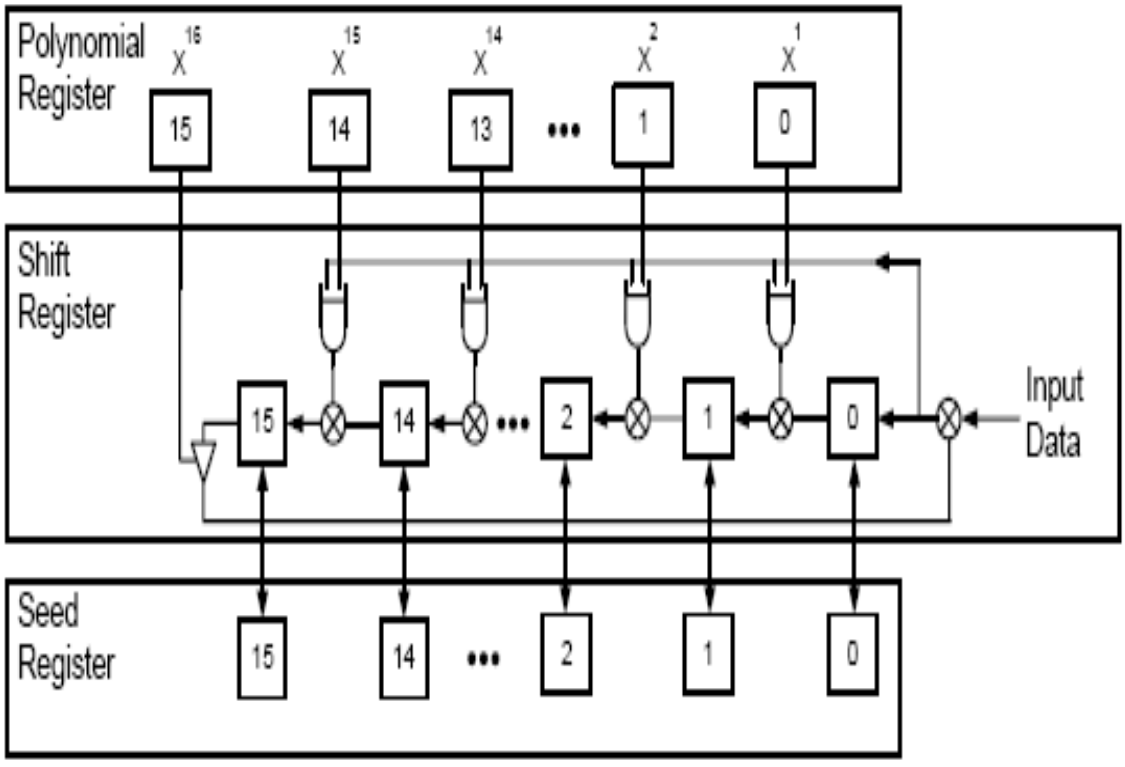
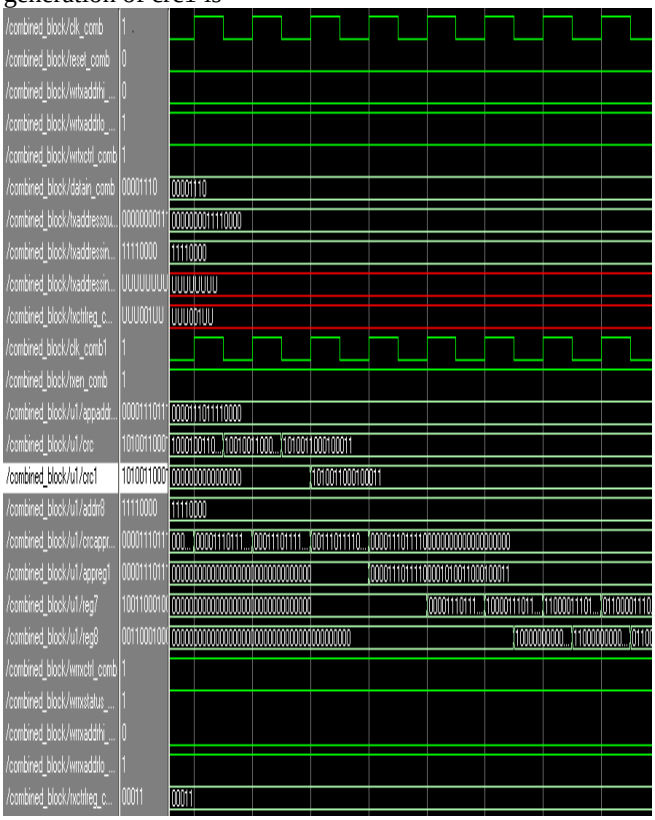


Fig.2 Basic block diagram of CRC16

4. SIMULATION OF HDLC CONTROLLER

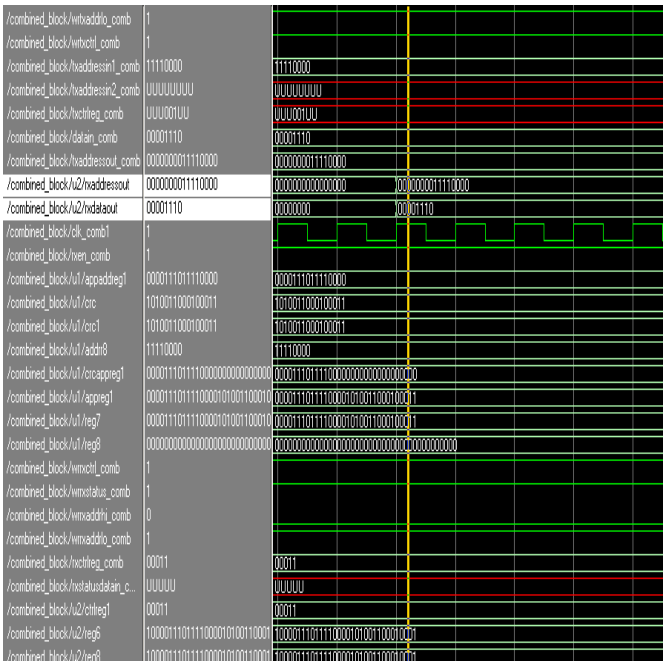
4.1 Simulation result for 8-bit data, 8 bit address and crc-16

For the data <=00001110and8 bit address <=11110000, we make clock = 1, reset = 0, wrtaddresshi = 0 and wrtaddresslo=1. After the address and the data are attached together, we divide them with a constant polynomial and append the remainder of the division along the data and address. The simulation result for the generation of crc1 is



4.2 Simulation result of the final O/P at the receiver end for 8 bit data, 8bit address and 16-bit crc

At the receiver the data input, address and the appendedcrc is again divided with the same constant polynomial and if the crc2 comes out be zero, it shows an error free reception of the packet. The receiver O/P i.e. rxdataout <= 00001110 and rxaddressout <=0000000011110000 which is same as that of transmitter.



5. CONCLUSION

The controller has the capability to operate in full duplex and half duplex mode. It can automatically check frame sequence generation using cyclic redundancy check i.e. CRC-16 and CRC-32 to ensure error free transmission. It is compatible with all the protocols present at the physical layer i.e.X.25 protocol and network layer i.e. Internet protocol (IP protocol).

REFERENCES

[1] Guozheng Li Nanlin Tan State Key Lab. of Rail Traffic Control & Safety, Beijing Jiaotong Univ., Beijing, China "Design and Implementation of HDLC Protocol and Manchester Encoding Based on FPGA in Train Communication Network", Information and Computing (ICIC), 2010 Third International Conference.

[2] Lu, Y., Z. Wang, L. Qiao and B. Huanq, 2002. "Design and implementation of multi-channel high speed HDLC data processor," IEEE International Conference on Communications, Circuits and Systems, and West Sino Expositions, 2: 1471-1475

[3] Amendola, A.M. Benso, A. Corno, F. Impagliazzo, L. Marmo, P. Prinetto, P. Rebaudengo, M. Sonza Reorda, M. CRIS, Napoli "Fault behavior observation of a microprocessor system through a VHDL simulation-based fault injection experiment Design Automation Conference, 1996, with EURO-VHDL '96 and Exhibition, Proceedings EURO-DAC '96, European ,16-20 Sep 1996

[4] Arshak, K. Jafer, E. McDonagh, D. Ibala, C.S. Univ. of Limerick, Limerick, "Modelling and simulation of wireless sensor system for health monitoring using HDL and Simulink mixed environment" Computers & Digital Techniques, IET Sept. 2007

[5] Gheorghiu, V., S. Kameda, T. Takagi, K. Tsubouchi and F. Adachi, 2008. "Implementation of frequency domain equalizer for single carrier transmission," In Proceedings of the 4thInternational Conference on Wireless Communications, Networking and Mobile Computing, WiCOM '08.

[6] Jun Wang; Wenhao Zhang; Yuxi Zhang; Wei Wu; Weiguang Chang; Sch. of Electron. & Inf. Eng., Beihang Univ. (BUAA), Beijing, China "Design and implementation of HDLC procedures based on FPGA" , Anti- ounterfeiting, Security, and Identification in Communication, 2009. ASID 2009. 3rd International Conference, 20-22 Aug. 2009

[7] Meng, X. and V. Chaudary, 2009. "Boosting data throughput for sequence database similarity searches on FPGAs using an adaptive buffering scheme," Parallel Computing, 35(1): 1-11.