

Design of HDLC Controller with CRC Generation Using VHDL

Mahfooz Ahmad¹, Saifur Rahman²

^{1,2}(Department of ECE, Integral University, Lucknow, India)

Abstract: In this paper we design the High-level Data Link Control to permit synchronous, code transparent data transmission. The control information is always in the same position and specific bit patterns which used for control differ dramatically from those representing data that reduces the errors chances. The transmission rate and data stream are controlled by 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 cost. The HDLC Controller MEGACELL is a high performance module for the bit oriented, switched, non-switched packet transmission module. It supports half duplex and full duplex communication lines, point-to-point and multipoint channels.

Keywords: HDLC Controller, VHDL, CRC, OSI Model and Bit stuffing.

I. 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.

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 0 after 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. 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 [2]. 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.

II. RELATED WORK

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. [1]

2.1 OSI 7 LAYERS REFERENCE MODEL FOR NETWORK COMMUNICATION

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]

The following list details the seven layers of the Open System Interconnection (OSI) reference model:

1. Layer 7—Application
2. Layer 6—Presentation
3. Layer 5—Session
4. Layer 4—Transport
5. Layer 3—Network
6. Layer 2—Data link
7. Layer 1—Physical

2.2 NETWORK PROTOCOL AND STANDARD

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 [3].

2.3 DATA LINK LAYER SPECIFICATIONS

The physical layer takes care of getting data on the wire and off of it again. At the data link layer, we must take this incoming stream of data from higher or lower layers and create frames from it. Handling the data requires a solid protocol that can perform better error checking and more efficient throughputs. The first to really address these needs was the Synchronous Data Link Control (SDLC) protocol from IBM. Developed for their Systems Network Architecture (SNA) systems, IBM created what is known as a bit-oriented protocol. This meant that specific bits themselves had meaning. Information wasn't formed just on the byte level. [4]

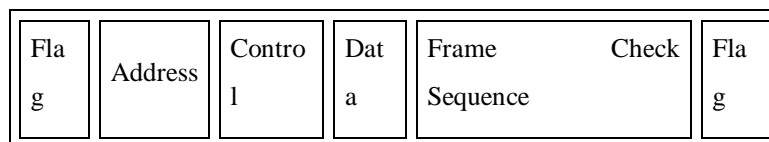


Fig. 2: SDLC Data Frame

2.4 ETHERNET SYSTEM

Ethernet was originally conceived of in the early 70s by Xerox designers. Its successful use in the Xerox Alto PC led two a consortium of three companies who wanted to be able to interlink various minicomputers. The companies were Digital Equipment Company (DEC), Intel Corporation and Xerox Corporation. Intel took on the task of providing the chips for NICs. Xerox wrote the software to operate it and DEC stepped in to make use of the technology for its minicomputers. The result was a high-speed connection that provided an alternative to IBM's networking architectures. In 1980 these companies released a

specification for Ethernet Version 1. This version was followed by a second version in 1982. These early versions comprise the standard we should refer to as "Ethernet" today. [5]

2.5 STANDARD ORGANIZATION

A wide variety of organizations contribute to internetworking standards by providing forums for discussion, turning informal discussion into formal specifications, and proliferating specifications after they are standardized. Most standards organizations create formal standards by using specific processes: organizing ideas, discussing the approach, developing draft standards, voting on all or certain aspects of the standards, and then formally releasing the completed standard to the public.

Some of the best-known standards organizations that contribute to internetworking standards include these [1]:

1. International Organization for Standardization (ISO)—ISO is an international standards organization responsible for a wide range of standards, including many that are relevant to networking. Its best-known contribution is the development of the OSI reference model and the OSI protocol suite.
2. American National Standards Institute (ANSI)—ANSI, which is also a member of
3. the ISO, is the coordinating body for voluntary standards groups within the United States. ANSI developed the Fiber Distributed Data Interface (FDDI) and other communications standards.
4. Electronic Industries Association (EIA)—EIA specifies electrical transmission standards, including those used in networking. The EIA developed the widely used EIA/TIA-232 standard (formerly known as RS-232).
5. Institute of Electrical and Electronic Engineers (IEEE)—IEEE is a professional organization that defines networking and other standards. The IEEE developed the widely used LAN standards IEEE 802.3 and IEEE 802.5.
6. International Telecommunication Union Telecommunication Standardization Sector (ITU-T)—formerly called the Committee for International Telegraph and Telephone (CCITT), ITU-T is now an international organization that develops communication standards. The ITU-T developed X.25 and other communications standards.
7. Internet Activities Board (IAB)—IAB is a group of internetwork researchers who discuss issues pertinent to the Internet and set Internet policies through decisions and task forces. The IAB designates some Request For Comments (RFC) documents as Internet standards, including Transmission Control Protocol/Internet Protocol (TCP/IP) and the Simple Network Management Protocol (SNMP).

III. METHODOLOGY

The whole design is organized as a collection of 2 sections that work together to efficiently perform the operation as shown in fig. 3. These units are:

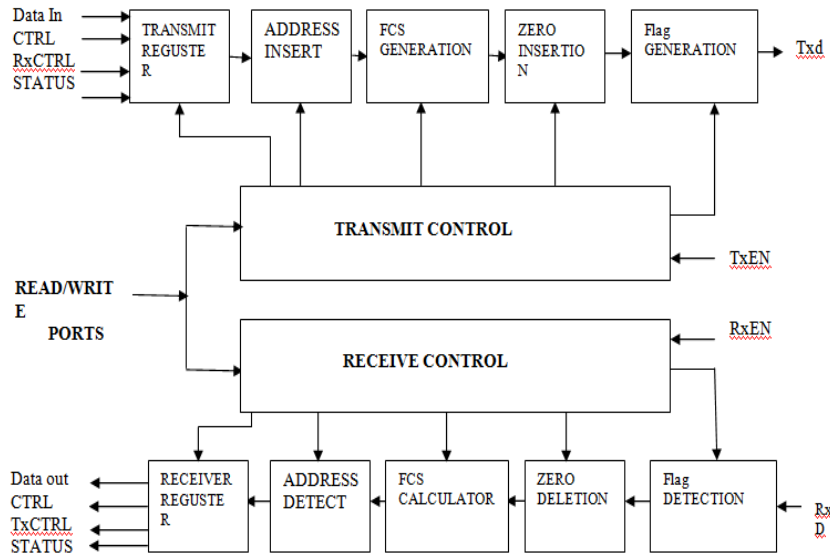
1. Transmitter section

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 by 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.

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 mode, 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.



errors.

Fig 3: Basic block diagram of HDLC Controller

V. SIMULATION RESULTS

After the design and implementation of the HDLC Controller, we found simulation result for 8-bit data, 8 bit address and 16 bit CRC Check for the data <=00001110 and 8 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 highlighted below-

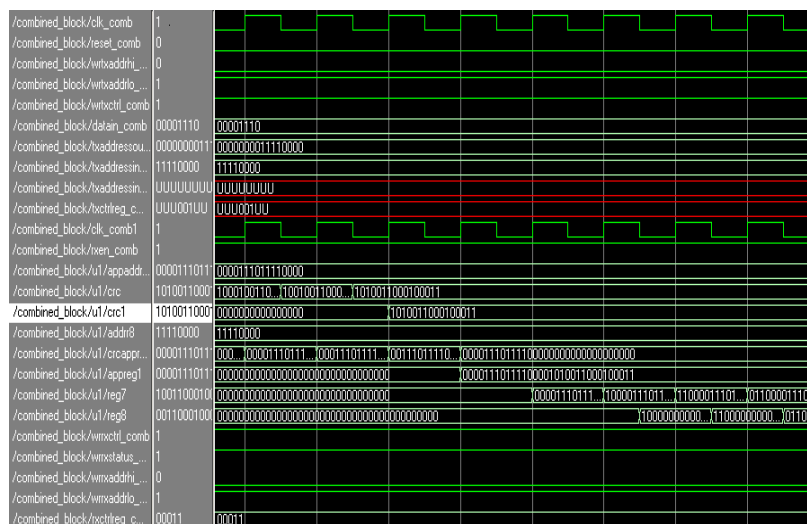


Fig. 4: Simulation result for 8-bit data, 8 bit address and 16 bit CRC Check

And at the receiver the data input, address and the appended crc 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

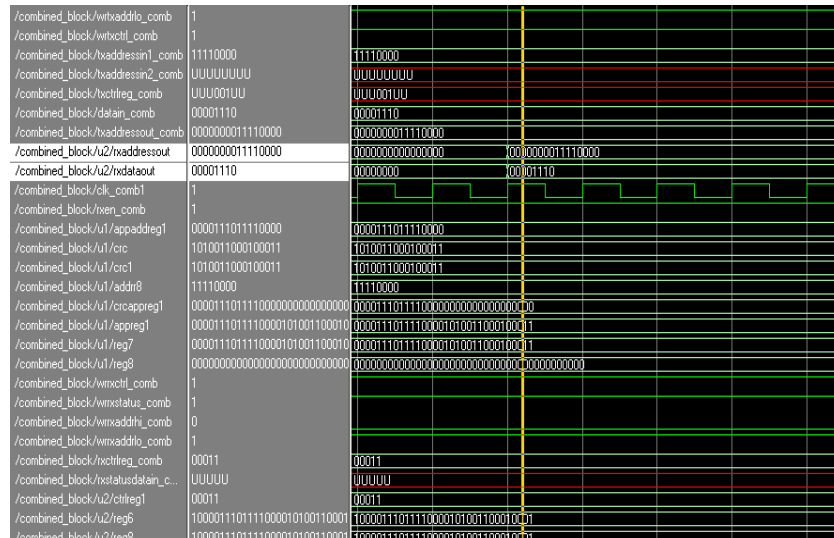


Fig. 5: Simulation result for O/P at the receiver for 16 bit address and 8 bit data

VI. CONCLUSION

Finally we conclude that HDLC Controller has been developed has the capability to operate in full duplex and half duplex mode. Controller 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). It has the capability to work in normal mode and asynchronous response mode. HDLC uses a process called bit stuffing where it stuffs a zero whenever it finds 5 consecutive 1's. VHDL model for the HDLC Controller has also been developed.

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] Arshak, K. Jafer, E. McDonagh, D. Ibalá, 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
- [3] 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 4th International Conference on Wireless Communications, Networking and Mobile Computing, WiCOM '08.
- [4] 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- counterfeiting, Security, and Identification in Communication, 2009. ASID 2009. 3rd International Conference, 20-22 Aug. 2009
- [5] 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.