Single-Channel XF-HDLC Controller

February 14, 2000 Product Specification

7810 South Hardy Drive, Suite 104
Tempe, Arizona 85284 USA
Phone: +1 888-845-5585 (USA)
+1 480-753-5585
Fax: +1 480-753-5899
E-mail: info@memecdesign.com
URL: www.memecdesign.com

Features

- Supports 4000X, Spartan, Virtex™, Virtex™-E, and Spartan™-II devices.
- Conforms to International Standard ISO/IEC 3309 Specification
- Starting point for a custom design
- 16-bit/32-bit CCITT-CRC generation and checking
- Flag & Zero insertion and detection
- Full Duplex Operation allowed
- DC to 53 Mbps (STS-1) data rate
- Full synchronous operation
- Interface can be customized for user FIFO and DMA requirements

AllianceCORE™ Facts	
Core Specifics	
See Table 1	
Provided with Core	
Documentation	User's guide
Design File Formats	Verilog source RTL[1]
Constraint Files	hdlc.ucf
Verification	Verilog Testbench, test vectors
Instatiation Templates	VHDL, Verilog
Reference Designs and Application Notes	Application Note
Additional Items	None
Simulation Tool Used	
Silos III	
Support	
Support provided by Memec Design Services	

Notes:
1. Synplify 5.08A used for synthesis

Applications

- Frame Relay, ISDN and X.25 protocols
- Logic consolidation

Table 1: Core Implementation Data

Supported Family	Device Tested	CLBs[2]		Clock IOBs	IOBs[1]		Performance[3] (MHz)	Xilinx Tools	Special Features
		Core	Core+ Ext logic		Core	Core+ Ext logic			
Spartan-II	2S15-5	183[2]	183[2]	2	32	32	77	M2.1i	None
Virtex-E	V50E-6	183[2]	183[2]	2	32	32	90	M2.1i	None
Virtex	V50-4	183[2]	183[2]	2	32	32	79	M2.1i	None
Spartan	S10-4	127	127	2	32	32	53	M1.5i	None
4000X	4005XL-2	134	134	2	32	32	57	M1.5i	None

Notes:
1. Assuming all core I/Os are routed off-chip.
2. Utilization numbers for Virtex, Virtex-E, and Spartan-II, are in CLB slices.
3. Minimum guaranteed speed

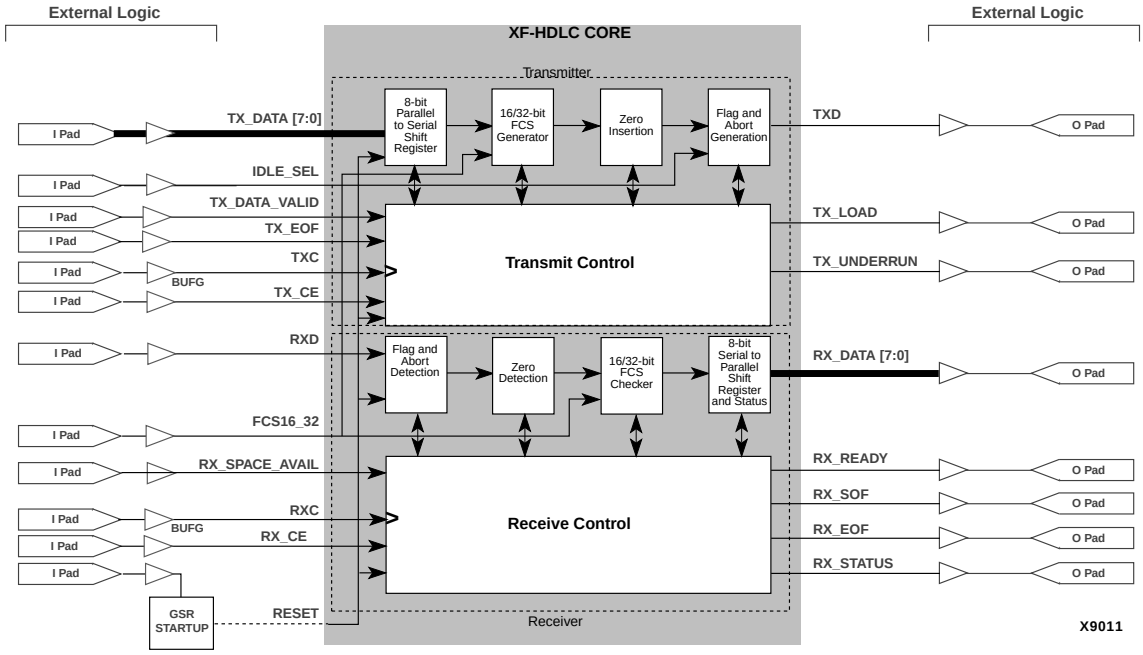


Figure 1: HDLC Controller Block Diagram

General Description

The XF-HDLC performs the most common functions of an HDLC controller. Data bytes are clocked into the device based on a divided version of the transmit clock. This data is then serialized and framed according to the rules of HDLC and sent out the serial transmit data pin. Receive frames are clocked into the receive data pin synchronous to the receive clock. The framing overhead is then stripped off and the data bytes are converted from serial to parallel and passed on through the parallel receive bus. Figure 1 shows the block diagram.

MDS cores are designed with the philosophy that no global elements should be embedded within the core itself. Global elements include any of the following components: STARTUP, STARTBUF, BSCAN, READBACK, Global Buffers, Fast Output Primitives, IOB Elements, Clock Delay Components, and any of the Oscillator Macros. MDS cores contain resources present in only the CLB array. This is done to allow flexibility in using the cores with other logic. For instance, if a global clock buffer is embedded within the core, but some external logic also requires that same clock, then an additional global buffer would have to be used.

In any instance, where one of our cores generates a clock, that signal is brought out of the core, run through a global buffer, and then brought back into the core. This philosophy allows external logic to use that clock without using another global buffer.

A result of this philosophy is that the cores are not self-contained. External logic must be connected to the core in order to complete it. MDS cores include tested sample designs that add the external logic required to complete the functionality. This datasheet describes both the core and the supplied external logic.

Functional Description

Transmitter

The transmitter portion of the HDLC core will begin to transmit when the user's external logic asserts the TX_DATA_VALID signal. The transmitter will respond by asserting the TX_LOAD signal to load the first byte of the packet. The timing diagram assumes that IDLE_SEL is tied to a '1' and the transmitter is generating continuous '1' bits between frames. If IDLE_SEL is set to a '0', the number of clocks from the assertion of TX_DATA_VALID to TX_LOAD will vary from 5 to 12. Before the transmitter can begin to send data serially, it must send an opening flag (7E). Immediately after the flag is sent, the first byte is clocked out of the input shift register. Once a transmit frame has begun, the user is required to make sure that data is available for each subsequent requested byte. The transmitter will continue to request data by asserting TX_LOAD until the user

supplies a TX_EOF signal. This informs the transmitter that the last byte is on the data bus. The transmitter then appends a 16- or 32-bit Frame Checking Sequence (FCS) to the transmitted data. After the FCS is sent, a closing flag (7E) byte is appended to mark the end of the frame. HDLC Transmitter consists of the following blocks as shown in Figure 1.

8-bit Parallel-to-Serial Shift Register

This block is responsible for capturing the user's transmit data on the rising edge to TXC when the TX_LOAD signal is asserted. Data is sent to the TXD pin and the FCS Generator at the same time.

16/32-bit FCS Generator

The Frame Checking Sequence (FCS) Generator is used to calculate a CRC across the transmitted message. Two different polynomials can be selected by statically controlling the FCS16_32 pin. The 16-bit FCS uses the polynomial $x^{16} + x^{12} + x^5 + 1$ and is selected when the FCS16_32 pin is a logic LOW. The 32-bit FCS uses the polynomial $x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$ and is selected when the FCS16_32 pin is a logic HIGH. Either type of FCS is complemented before being transmitted.

Zero Insertion

The transmitter is responsible for examining the frame content between the opening and closing flags and checking for 5 consecutive '1' bits, including the FCS bits. If 5 consecutive '1' bits are detected, a '0' bit is inserted into the serial transmission. This will allow the receiver to distinguish between an opening or closing flag and actual data.

Flag and Abort Generation

An opening flag is sent when the user asserts the TX_DATA_VALID signal. As soon as the last byte of the FCS has been transmitted, a closing flag is sent. If a transmission has been started and the TX_DATA_VALID signal is deasserted while the transmitter is requesting another byte, an underrun condition will occur. This condition will be reported with the TX_UNDERRUN pin, but will also result in the transmitter sending 8 consecutive '1' bits. This is defined as an "abort" condition.

The transmitter will inter-frame fill by driving out a contiguous stream of '1' bits or a repeating flag. If back-to-back frames are available to send (the user continues to assert TX_DATA_VALID), the transmitter will share the closing flag of the first frame with the opening flag of the second frame.

Serial Output

All data exits the transmitter on the TXD pin and transitions on the rising edge of TXC.

Transmit Control

The control state machines and interface timing for the transmitter are driven by the rising edge of TXC.

Receiver

The receiver clocks serial HDLC frames in continuously through the RXD pin. When an opening flag is recognized, the receiver locks to all subsequent octet bytes. The user informs the receiver of the ability to store the frame by asserting the RX_SPACE_AVAILABLE input. The receiver informs the user that a data byte is available by asserting the RX_READY signal. The receiver indicates the beginning of the frame by asserting the RX_SOF signal. Bytes will continue being passed to the user until the receiver recognizes the closing flag. At this point, the last byte of the FCS sequence will be passed to the user coincident with the RX_EOF signal. It must be stressed that the core does not contain the additional pipeline registers to "swallow" the 2 or 4 bytes of FCS, and these will therefore be passed on to the user. If this is undesirable, the corresponding pipeline should be added externally to keep these bytes from passing on as part of the received frame. After the reception of the frame has completed, the receiver will pass a byte of status information to the user by placing the status on the receive data bus and asserting the RX_STATUS signal. The Receiver consists of the following blocks as shown in Figure 1.

Flag and Abort Detection

The receiver begins operation by hunting for an opening flag character. Once the flag has been recognized. the receiver begins to receive the incoming frame, but continues to monitor for a closing flag. Once the closing flag has been detected, the frame is complete. Once the receiver has detected an opening flag, it will monitor the serial data stream to see if 8 consecutive '1' bits are detected. This condition is defined as a receive abort and is reported to the user through a receive status bit. The receiver is capable of handling back-to-back frames where the closing flag of the first frame also acts as the opening flag of the second frame. The receiver will idle on either contiguous '1' bits or repeating flag characters.

Zero Detection

The receiver checks the incoming data frame to see if 5 consecutive '1' bits are received. If this condition is detected, the following zero is deleted from the incoming frame.

16/32-bit FCS Check

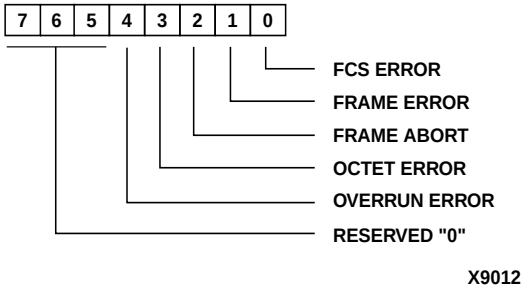
The Frame Checking Sequence (FCS) Checker performs the same generator polynomial division as the transmitter across the entire transmitted message including the FCS field. The result of this polynomial division will be a constant remainder indicating the packet integrity. The receiver sup-

ports the same 16-bit and 32-bit FCS as the transmitter. The version is statically selected using the FCS16_32 pin, the 16-bit version is selected by a logic LOW and the 32-bit version is selected with a logic HIGH.

8-bit Serial Shift Register

As serial data is clocked into the receiver, it is assembled back into bytes through a serial-to-parallel shift register. The receiver informs the user of a valid byte by asserting the RX_READY signal. RX_READY can be further qualified with additional signals to help the user track the progress of an incoming frame. RX_SOF is asserted coincident with RX_READY to indicate reception of the first octet of a frame. RX_EOF is asserted coincident with RX_READY to indicate the last byte of the receive FCS. The RX_STATUS signal is asserted coincident with RX_READY to indicate to the user that the receive data contains a valid byte of status information.

Status Byte



The status byte will be presented to the user at the end of the frame or after a receive error is detected. The receiver will inform the user of valid status on the RX_DATA bus by the coincident assertion of RX_READY and RX_STATUS.

The FCS ERROR will be set at the end of the frame if the remainder after polynomial division does not match the proper 16-bit or 32-bit constant.

The FRAME ERROR status bit will be set if a frame is received that is shorter than 32 bits when using the 16-bit FCS, and shorter than 48 bits when using the 32-bit CRC. There is no test to check for frame lengths that exceed a certain length. This bit will also be set when the OCTET ERROR is set.

The FRAME ABORT status bit will be set if the receiver has detected 8 consecutive ‘1’ bits in a row after frame reception has begun.

The OCTET ERROR status bit is set whenever the closing flag is received on an odd bit boundary. The receiver tests to make sure all frames are an integral number of octets.

All remaining status bits are reserved and will be presented as ‘0’.

Receive Control

The control state machines and interface timing for the receiver is driven by the rising edge of RXC.

Core Modifications

Customizing is available through Memec Design Services.

Pinout

The pinout of the Single-Channel XF-HDLC Controller core has not been fixed to specific FPGA I/O, allowing flexibility with a user’s application. Signal names are provided in Table 2.

Verification Methods

Complete functional and timing simulation has been performed on the HDLC using SILO III. Simulation vectors are provided with the core. The HDLC core has been hardware tested with TTC Fireberd 6000A Frame Relay Option. This core has also been used successfully in customer designs.

Ordering Information

The Single Channel XF-HDLC Controller core is provided under license from Memec Design Services for use in Xilinx programmable logic devices and Xilinx HardWire gate arrays. Please contact Memec for pricing and more information.

Information furnished by Memec Design Services is believed to be accurate and reliable. Memec Design Services reserves the right to change specifications detailed in this data sheet at any time without notice, in order to improve reliability, function or design, and assumes no responsibility for any errors within this document. Memec Design Services does not make any commitment to update this information.

Memec Design Services assumes no obligation to correct any errors contained herein or to advise any user of this text of any correction, if such be made, nor does the Company assume responsibility for the functioning of undescribed features or parameters. Memec Design Services will not assume any liability for the accuracy or correctness of any support or assistance provided to a user.

Memec Design Services does not represent that products described herein are free from patent infringement or from any other third-party right. No license is granted by implication or otherwise under any patent or patent rights of Memec Design Services.

Memec Design Services products are not intended for use in life support appliances, devices, or systems. Use of a Memec Design Services product in such application without the written consent of the appropriate Memec Design Services officer is prohibited. All trademarks, registered trademarks, or service marks are property of their respective owners.

Table 2: Core Signal Pinout

Signal	Signal Direction	Description
TX_DATA[7:0]	Input	Transmitter Parallel Data Bus: 8-bit transmit data bus loaded synchronously based on TX_LOAD signal and TXC clock.This bus is driven by the user's transmit FIFO or RAM buffer.
IDLE_SEL	Input	Idle Select: Selects inter-frame idle fill type. When tied low, device sends continuous flags between frames; when tied high, device sends continuous ones between frames.
TX_DATA_VALID	Input	Transmit Data Valid: active high user input, synchronous to TXC, to inform transmitter that an external packet is ready to send.
TX_EOF	Input	Transmit End-Of-Frame: an active high user input pulse, synchronous with TXC, to inform transmitter that current data byte is the last byte of a sending packet. This input should coincide with TX_LOAD.
TXC	Input	Serial Transmit Clock: a user-provided clock for all transmit activity that takes place on the low to high transition of TXC.
TX_CE	Input	Transmit Clock Enable: an active high user input, synchronous with TXC.
RXD	Input	Serial Receive Data: an input for serial receive data, sampled on the rising edge of RXC.
FCS16_32	Input	FCS Select: selects 16-bit FCS, $x^{16} + x^{12} + x^5 + 1$, when tied low. Selects 32-bit FCS, $x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$, when tied high.
RX_SPACE_AVAIL	Input	Receive Space Available: an active high user input, synchronous to RXC, to inform the receiver that external receive FIFO or buffer RAM can accept more data.
RXC	Input	Serial Receive Clock: a user provided clock for all receive activity that takes place on low to high transition of RXC.
RX_CE	Input	Receive Clock Enable: an active high user input, synchronous with RXC.
RESET	Input	Global Reset: asynchronously resets all internal registers.
TXD	Output	Serial Transmit Data: provides serial transmit data and transitions on the rising edge of TXC clock.
TX_LOAD	Output	Transmit Load: an output pulse from transmitter, synchronous to TXC, that acts as a clock enable signal to external transmit buffer to request an input byte.
TX_UNDERRUN	Output	Transmit Underrun: an active high output pulse, synchronous to TXC, from the transmitter indicating an underrun error. This occurs upon start of frame transmission, if TX_DATA_VALID is deasserted when TX_LOAD is asserted.
RX_DATA[7:0]	Output	Receive Parallel Data Bus: 8-bit receive data bus providing user output data synchronous to RX_READY and RXC clock. This bus is tied to user's receive FIFO or RAM buffer and also reports frame status at the end of a receive.
RX_READY	Output	Receive Ready: an active high pulse from receiver, synchronous to RXC, that acts as a clock enable signal to external receive buffer to output a received byte. The STATUS pin distinguishes receive data from frame status.
RX_SOF	Output	Receive Start-Of-Frame: an active high pulse, synchronous to RXC, to inform user that current receive data byte is first byte of a frame. This pulse coincides with RX_READY pulse.
RX_EOF	Output	Receive End-Of-Frame: an active high pulse, synchronous to RXC, to inform the user that the current receive byte is the last byte (either 2 or 4) of the frame checking sequence. This pulse is coincident with the RX_READY pulse.
RX_STATUS	Output	Receive Status: an active high pulse, synchronous to RXC, to inform the user that receive frame status is being output on the RX_DATA bus. RX_STATUS is coincident with the RX_READY signal.

Recommended Design Experience

For the source version, users should be familiar with Verilog HDL entry, synthesis, simulation, and Xilinx design flows.

Related Information

ISO/IEC 3309 *High-Level Data Link Control Procedures-Frame Structure*

ISO/IEC 4335 *High-Level Data Link Control Procedures-Elements of Procedures*

ISO/IEC 8885 *High-Level Data Link Control Procedures – General Purpose XID Frame Information Field Content and Format*

Xilinx Programmable Logic

For information on Xilinx programmable logic or development system software, contact your local Xilinx sales office, or:

Xilinx, Inc.
2100 Logic Drive
San Jose, CA 95124
Phone: +1 408-559-7778
Fax: +1 408-559-7114
URL: www.xilinx.com

For general Xilinx literature, contact:

Phone: +1 800-231-3386 (inside the US)
+1 408-879-5017 (outside the US)
E-mail: literature@xilinx.com

For AllianceCORE™ specific information, contact:

Phone: +1 408-879-5381
E-mail: alliancecore@xilinx.com