32 Channel HDLC Core V1.2

May 3, 2000 Product Specification

Powered by

Xilinx Inc.
2100 Logic Drive
San Jose, CA 95124
Phone: +1 408-559-7778
Fax: +1 408-559-7114
E-mail: logicore@xilinx.com
URL: www.xilinx.com/ipcenter
Support: www.support.xilinx.com

Features

- Optimized for Virtex™, Virtex™-E, Spartan™-II
- Support for up to 32 full duplex channels, total data rate >40Mb/s
- 16 bit (CRC-16) and 32 bit (CRC-32) frame check sequence - selectable
- 8/16 bit address insertion and detection - selectable
- Transmit frame address insertion disable capability - selectable
- Receive frame address discard capability - selectable
- Selectable broadcast (all stations) address matching
- Selectable “all addresses” mode to capture all received frames
- Flag sharing between back to back frames
- Transparent mode selectable
- Receive frame error flag
- Receive frame abort flag
- Transmit frame abort control and flag
- Channel status indicators
- 8 bit data interface suitable for interfacing to transmit and receive FIFOs
- Serial interface with external clocking for interfacing to the PCM-highway
- Transmission is synchronous to network interface with back pressure mechanism. Buffering at the network interface not required
- Compatible with ITU Q.921
- Compatible with ISO/IEC 3309

Applications

- X.25
- Frame Relay
- B-channel and D-channel

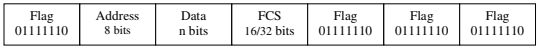
LogiCORE™ Facts

Core Specifics	
Device Family	Virtex™, Virtex™-E, Spartan™-II
Slices Used	523
IOBs Used	151
CLKIOBs Used	1
System Clock fmax	>40MHz
Device Features Used	-
Provided with Core	
Documentation	Datasheet and User Guide
Design File Formats	Targeted EDIF
Constraint Files	HDLC32.ucf
Verification Tool	Modelsim v.5.3 VHDL and Verilog testbenches supplied
Schematic Symbols	HDLC32.xsf
Evaluation Model	Postlayout .vhd, veri, edf
Design Tool Requirements	
Xilinx Core Tools	Design Manager 2.1
Entry/Verification Tool	FPGA Express 3.3 Modelsim 5.3
Support	
Support provided by Xilinx, Inc.	

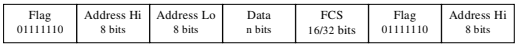
General Description

The HDLC Protocol core is a high performance module for the bit oriented packet transmission mode. It is suitable for Frame-Relay, X.25, ISDN B-Channel (64 KBit/s) and D-Channel (16 KBit/s). The core fulfills the specification according to ITU Q.921, X.25 Layer 2 recommendation. The data stream and transmission rate is controlled from the network node (PCM highway clock) with a back pressure mechanism. This eliminates additional synchronization and buffering of the data at the network interface. The data interface is 8-bit wide synchronous and is suitable for interfacing to transmit and receive FIFOs.

An example of an HDLC frame structure is shown below, for an 8-bit address field with an interframe fill pattern of back-to-back flags. This figure does not include bits inserted for transparency. The fields are transmitted from left to right, least significant bit first.



The following example illustrates a 16-bit address field, with another frame immediately following.



The sections below describe the contents and purpose of each field. Much of this has been drawn from the IETF standards for PPP, such as RFC1662 and STD-51.

Transparency is performed by the core on all bits between and including the address and the FCS. This operation involves inserting a zero after any sequence of 5 consecutive ones in the transmitted data stream (including the last five bits of the FCS). The receiver core detects and removes these inserted zero bits.

At any time the transmission of a frame can be aborted by sending the Abort flag, which is 01111111. An aborted frame will be marked as such by the receiver core. Additionally the channel can be set to an inactive or "Idle" state, where a continuous sequence of one bits is transmitted.

Flag Sequence

Each frame begins and ends with a Flag Sequence, which is the binary sequence 01111110 (hexadecimal 0x7e). All implementations continuously check for this flag, which is used for frame synchronization.

Only one Flag Sequence is required between two frames. Two consecutive Flag Sequences constitute an empty frame, which is silently discarded, and not counted as a FCS error.

Address

The Address field is of programmable size, a single octet or a pair of octets. The field can contain the value programmed into the Transmit Address Register at the time the frame is started. This can be any arbitrary address, or the broadcast or "All-Stations" address, which is all ones.

The receiver core matches the received address field against the value currently held in the Receive Address Register, or, if programmed to, the broadcast address. Frames with unrecognized addresses are silently discarded.

Data

The data field may contain any number of bits. There is no limit within the core to the maximum size of the data field.

Data octets are reassembled by the core and presented to the host interface with a suitable byte strobe. Any extra bits in the last received octet which do not correspond to received data bits are undefined.

The data stream presented to the host can be configured to either include the address field, or have it removed. FCS octets are always removed from the stream to the host.

Higher level protocols such as LAPD, LAPF or PPP will define values for subfields within the data field.

Frame Check Sequence (FCS)

The Frame Check Sequence field is programmable to either 16 bits (two octets), or 32 bits (four octets). The FCS is transmitted least significant octet first, which contains the coefficient of the highest term in the generated check polynomial.

The FCS field is calculated over all bits of the Address, Control, and data fields, not including any bits inserted for transparency. This also does not include the Flag Sequences nor the FCS field itself.

The end of the data field is found by locating the closing Flag Sequence and removing the Frame Check Sequence field.

Frames where the received FCS and calculated FCS indicate that the data has been corrupted during transmission, will be marked by the core at the time that the end of the frame is signalled to the host.

Functional Description

The HDLC Protocol core is divided into modules as shown in Figure 1. The function of the core is described in detail in the following paragraphs.

Transmitter

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 is 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 with a closing flag. Extra zeros are inserted into the bit stream to avoid transmission of the control flag sequences within the frame data.

The transmit data is available on the TxD pin with appropriate setup to be sampled by Clk. If TxEN is deasserted, the transmit pipeline is stalled, and the TxD pin is disabled.

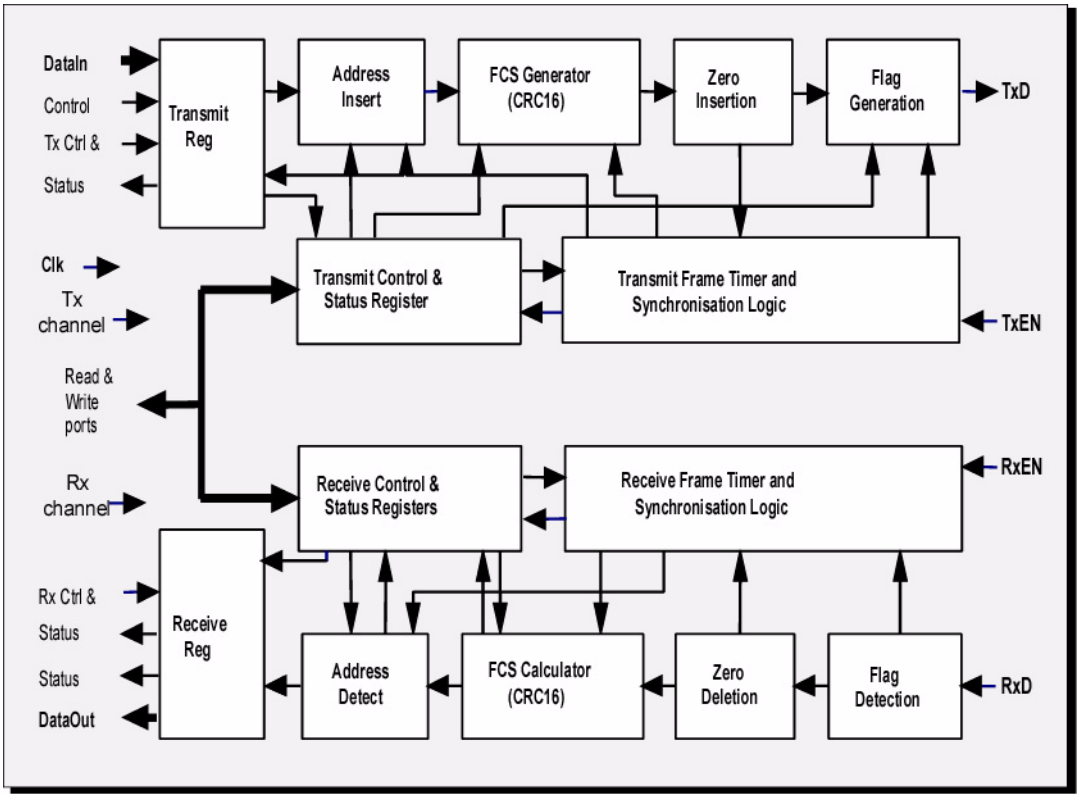


Figure 1: HDLC Controller Block Diagram

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 underruns. 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 clears the abort and any transmit FIFO underrun condition.

Up to 32 separate transmit channels can be processed, with any channel active on a tick by tick basis. No time is required to change the core from one channel to another. A five bit input port controls which channel is currently active. For any one tick, the active channel will be monitoring all input ports and driving all output ports. All inactive channels will be stalled, with their state preserved until reactivation. Only the currently active channel can be configured with the register access ports and strobes.

Receiver

The HDLC Protocol core 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 searches the bit stream for the flag

sequence in order to determine the frame boundaries. Any stuffed zeros are detected and removed and the FCS is calculated and checked. Frame data is placed on the receive data interface and made avail-able 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 conditions 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.

Up to 32 separate receive channels can be processed, with any channel active on a tick by tick basis. No time is required to change the core from one channel to another. A five bit input port controls which channel is currently active. For any one tick, the active channel will be monitoring all input ports and driving all output ports. All inactive channels will be stalled, with their state preserved until reactivation.

Only the currently active channel can be configured with the register access ports and strobes.

Initializing the Core

To correctly initialize the core for correct operation it is necessary to perform the following sequence during reset.

While notReset is asserted (low) the TxChannel and RxChannel inputs must be cycled through all values. For example this can be achieved using a minimum reset period of 33 Clk ticks and incrementing TxChannel and RxChannel from 0 to 31, with one extra Clk tick before notReset is deasserted to allow for pipelining.

Pinout

A representative symbol, with the signal names, is shown in Figure 2.

Unless otherwise stated all signals are active high and bit(0) is the least significant bit.

The following table summarizes the signal functions.

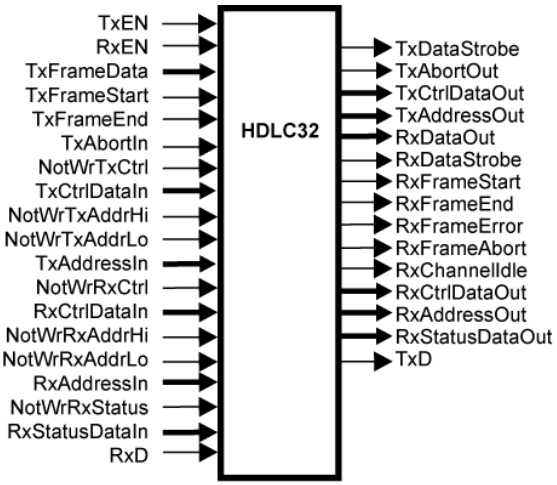


Figure 2: Core Schematic

Table 1: Signal Pinout

Signal	Signal Direction	Description
Clk	Input	**Core Clock**, bit rate clock input signal at up to core rated maximum clock speed for the chosen target technology.
NotReset	Input	**Core reset (active low)**. Asynchronous reset that initializes the HDLC into an inactive state, transmitting the idle pattern.
TxChannel	Input	**Transmit Channel Context.** This input selects which transmitter channel is active on the next Clk tick (i.e. a pipeline depth of one).
RxChannel	Input	**Receive Channel Context**. This input selects which receiver channel is active on the next Clk tick (i.e. a pipeline depth of one).
TxD	Output	**Transmission data** is the serial output data line, which is permanently driven and can be directly connected to the PCM highway. Data bits are clocked out by the rising edge of Clk and in least significant bit first order.
TxEN	Input	**Transmission Enable** input signal enables the transmission data and the TxD signal. May be generated by the channel multiplexer. Data is clocked out on the Clk rising edge when TxEN is asserted, and transmitter core state is frozen while TxEN is deasserted.
RxD	Input	**Receive Data** is the serial input data line, sampled on the rising edge of Clk when RxEN is asserted in least significant bit first order.
RxEN	Input	**Receive Enable** input signal enables the received data. It may be generated by a channel multiplexer. Receiver core state is frozen while RxEN is deasserted
TxFrameData	Input	**Transmit Data**, a bytewide data bus into the transmit section of the core - sampled and held every 8 Clk cycles, so should be held valid for 8 Clk cycles.
TxDataStrobe	Output	**Transmit Data Write Strobe**, asserted to acknowledge that a byte has been latched by the core. It is asserted during the last cycle of an 8 clock cycle sequence - see timing diagrams.
TxFrameStart	Input	**Transmit: Start new frame**. When this signal is asserted at the same time as a byte is written that byte will be the first in a new HDLC frame.
TxFrameEnd	Input	**Transmit: End of frame**. When this signal is asserted at the same time as a byte is written that byte will be the last in the current HDLC frame.
TxAbortIn	Input	**Transmit Frame Abort Input**. When asserted, transmission is stopped by Frame Abort. It can be a single Clk cycle or longer.

Table 1: Signal Pinout

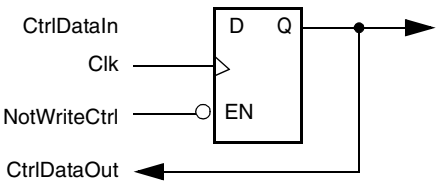
Signal	Signal Direction	Description
TxAbortOut	Output	**Transmit Frame Abort Flag**. Asserted to indicate that a frame abort has occurred either via the input pin or the control register bit. It is held asserted until the abort condition is cleared.
NotWrTxCtrl	Input	**Transmit Control Register Write Strobe (active low)**. Asserted to write data into the transmit control register on the rising edge of Clk.
TxCtrlDataIn	Input	**Transmit Control Register In**. This port carries the new value for the transmit control register, written on the rising edge of Clk when NotWrTxCtrl is asserted. Bits within this register are defined in a section below.
TxCtrlDataOut	Output	**Transmit Control Register Out**. This port reflects the current values of the transmit control register. Bits within this register are defined in a section below.
NotWrTxAddrHi	Input	**HDLC Frame Tx Address Strobe MSB (active low)**. The MSB of the HDLC frame address is latched on the rising edge of Clk when this strobe is asserted
NotWrTxAddrlo	Input	**HDLC Frame Tx Address Strobe LSB (active low)**. The LSB of the HDLC frame address is latched on the rising edge of Clk when this strobe is asserted
TxAddressIn	Input	**HDLC Frame Tx Address input**, is the address inserted into the HDLC frame. The register has separate enables for LSB and MSB to ease interfacing to an 8 bit bus (NotWrTxAddrHi and NotWrTxAddrLo). The address is latched on the rising edge of Clk when the relevant strobe is asserted.
TxAddressOut	Output	**HDLC Frame Tx Address output**, shows the current HDLC Tx frame address.
RxDataOut	Output	**Receive Data**, a bytewide data bus from the receive section. Carries a valid byte for reading when RxDataStrobe is asserted.
RxDataStrobe	Output	**Receive Data Output Strobe**. Asserted to indicate that a valid byte of data is available on the RxDataOut bus. It is asserted for only 1 Clk cycle and assumes that the data is latched when this occurs - see timing diagrams.
RxFrameStart	Output	**Receive Start of new frame**. When this signal is asserted at the same time as a byte is read from the core, that byte was the first in a new HDLC frame.
RxFrameEnd	Output	**Receive: End of current frame**. When this signal is asserted at the same time as a byte is read from the core, that byte was the last in the current HDLC frame. This signal is also used to qualify RxFrameError.
RxFrameError	Output	**Receive Frame contained error**. This signal is asserted at the same time as the last byte in a HDLC frame is read from the core, to indicate the presence of an FCS check error in that frame. This signal is not asserted until the end of the frame and it is only driven for the last byte in the frame (i.e. RxFrameEnd will also be asserted).
RxFrameAbort	Output	**Receive Frame aborted**. When this signal is asserted at the same time as a byte is read from the core, that byte and all previously read bytes since the start of the frame should be discarded. This indicates that the Abort signal sequence has been seen by the receiver. The next valid byte from the receiver core will be the start byte of the next frame.
RxChannelIdle	Output	**Receive Channel sees Idle**. This signal is asserted while the receiver is seeing continuous IDLE pattern driven onto the serial bus and all received data bits have been flushed from the receiver core.
NotWrRxCtrtl	Input	**Receive Control Register Write Strobe (active low)**. Asserted to write data into the receive control register on the rising edge of Clk.
RxCrtldataIn	Input	**Receive Control Register Data Input**. This port carries the new value for the receive control register, written on the rising edge of Clk when NotWrRxCtrl is asserted. Bits within this register are defined in a section below.
RxCtrlDataOut	Output	**Receive Control Register Data Output**. This port reflects the current values of the receive control register. Bits within this register are defined in a section below.
NotWrRxAddrHi	Input	**Receive HDLC Frame Address Strobe MSB (active low)**. The MSB of the required HDLC frame address is latched on the rising edge of Clk when this strobe is asserted (active low).

Table 1: Signal Pinout

Signal	Signal Direction	Description
NotWrRxAddrLo	Input	**Receive HDLC Frame Address Strobe LSB (active low)**. The MSB of the required HDLC frame address is latched on the rising edge of Clk when this strobe is asserted (active low).
RxAddressIn	Input	**Receive HDLC Frame Address input**. The receive HDLC frame address is latched on the rising edge of Clk when the relevant strobes (NotWrRxAddrHi and NotWrRxAddrLo) are asserted (active low).
RxAddressOut	Output	**Receive HDLC Frame Address Strobe output**. This port reflects the current values of the HDLC frame address register.
NotWrRxStatus	Input	**Receiver Status Register Write Strobe (active low)**. Asserted to write data into the HDLC receiver status register on the rising edge of Clk.
RxStatusDataIn	Input	**Receiver Status Register Data Input**. This port carries the new value for the HDLC core receive status register, written on the rising edge of Clk when NotWrRxStatus is asserted. Bits within this register are defined in a section below.
RxStatusDataOut	Output	**Receiver Status Register Data Out**. This port reflects the current values of the HDLC core receive status register. Bits within this register are defined in a section below.

Registers

The read/write control registers within the core have a common interface method. A data in signal feeds a register, which is clocked by the core Clk signal. An active low write enable signal controls the external loading of the register. The current value of the register is also available as an output signal. This flexible interface allows a number of different host interface methods to be easily implemented. An example for a single bit control register is illustrated below. Registers for the currently active channel can be accessed in this way. Inactive channels are inaccessible.



Transmit Control Register

Table 2: Transmit Control Register

Bit	Usage	Reset	Description
0	R/W	0	**Channel Enable**. 1 = channel enabled 0 = channel will transmit the Idle pattern (all ones)
1	R/W	0	**HDLC Protocol Disable.** 1 = transparent mode selected 0 = normal HDLC operation selected
2	R/W	0	**Address Insertion.** 1 = address insertion enabled 0 = address insertion disabled
3	R/W	0	**Address size**. 1= 16 bit address 0= 8 bit address
4	R/W	0	**FCS Size**. 1 = CRC-32 used for FCS 0 = CRC-16 used for FCS
5	R/W	0	**Frame Abort.** 1 = indicates that a frame abort is in progress. It can also be written to 1 to initiate a frame abort. 0 = normal operation
6	R/W	0	**Stop Tx on frame abort**. 1 = the transmit core will stop, awaiting further servicing from the host microprocessor, i.e. the host processor must clear the Frame Abort bit before normal operation can resume. 0 = the HDLC core will attempt to recover from frame aborts by continuing operation with the next frame. The Frame Abort bit is automatically cleared in this case, and the TxAbortOut pin is unused.
7	R/W	0	**Tx channel active**. This status bit is cleared at the start of a frame, and set when the transmit channel is enabled. It can be used to check that the transmitted data is reaching the serial bus. 1 = transmit channel enabled (TxEN asserted) 0 = transmit frame started (TxFrame Start asserted)

Transmit Address Register (LSB and MSB)

Table 3: Transmit Address Register

Bit	Usage	Reset	Description
7:0	R/W	00..00	**HDLC transmit frame address** - least significant byte
15:8	R/W	00..00	**HDLC transmit frame address** - most significant byte

Receive Control Register

Table 4: Receive Control Register

Bit	Usage	Reset	Description
0	R/W	0	**HDLC Protocol Disable**. 1 = transparent mode selected 0 = normal HDLC operation
1	R/W	0	**Address Recognition Disable**. 1 = all addresses are recognized and passed to the host. Setting this bit implies that the address field is passed through to the host, regardless of the state of the Address Field Discard bit. 0 = normal address matching operation.
2	R/W	0	**Address Field Discard**. 1 = address field in received frames is discarded 0 = address field in received frames is passed through Note that when Address Recognition is disabled, the address field is passed through to the host processor regardless of the setting of this bit.
3	R/W	0	**Address size**. 1 = 16 bits 0 = 8 bits
4	R/W	0	**Broadcast address match Disable** - when set the HDLC receiver does not match on the broadcast address (all 1's) 1 = broadcast match disable 0 = broadcast match enable
5	R/W	0	**FCS Size**. 1 = CRC-32 used for FCS 0 = CRC-16 used for FCS

Receive Status Register

Table 5: Status Register

Bit	Usage	Reset	Description
0	R/W	0	**Channel Enabled**. This bit is cleared when RxEN is asserted. It can be set by writing a 1, and is used to check that there is activity on the receive serial channel. 1 = channel inactive 0 = channel active
1	R/W	0	**Receiving frame data**. This bit is set on receipt of the first byte of a frame, and cleared on receipt of frame abort or the closing flag. 1 = currently receiving a frame 0 = currently not receiving a frame
2	R/W	0	**Channel active**. This bit is set whenever the receiver is not receiving Idle pattern (all ones). It is used to indicate that there are frames present on the serial channel. 1 = data on channel 0 = idle pattern on channel
3	R/W	0	**Frame error**. Last frame contained an error. This status bit is only updated on the closing byte of complete frames, and thus shows the status of the previous frame, rather than the current one. It is recommended that the RxFrameError signal be used instead. 1 = frame error 0 = no frame error
4	R/W	0	**Frame aborted**. Last frame was aborted. This status bit is cleared on the closing byte of complete frames, and set by a frame abort sequence. It thus shows the abort status of the previous frame, rather than the current one. It is recommended that the RxFrameAbort signal be used instead. 1 = frame aborted 0 = frame not aborted

Receive Address Register (LSB and MSB)

Table 6: Receive Address Register

Bit	Usage	Reset	Description
7:0	R/W	00..00	**HDLC receive frame address** - least significant byte
15:8	R/W	00..00	**HDLC receive frame address** - most significant byte

HDL Entity Definition

The HDL entity/module declaration for the core is included with the deliverables. This shows the generic parameters that may be specified for the function, the default values if none are specified, and the I/O port names and allowable values. The module may be included in a user design by instancing the module as a component. A VHDL component declaration for the core is provided in the HDLC32.vho file and a Verilog module declaration HDLC32.veo in the /net/ directory located at $xilinx/coregen/ip/xilinx/<modulename_vir>/com/xilinx/ip/<modulename_vir>/<product_family>/net.

Timing Diagrams

The following timing diagrams provide only the functional timing for the HDLC32 core as the actual timings are dependent on the target implementation technology.

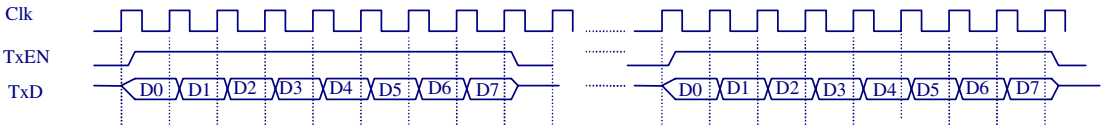


Figure 3: Serial Network Interface Transmit Timing

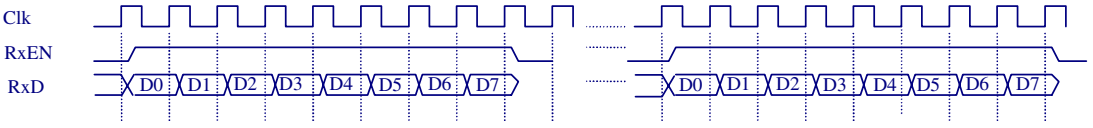


Figure 4: Serial Network Interface ReceiveTiming

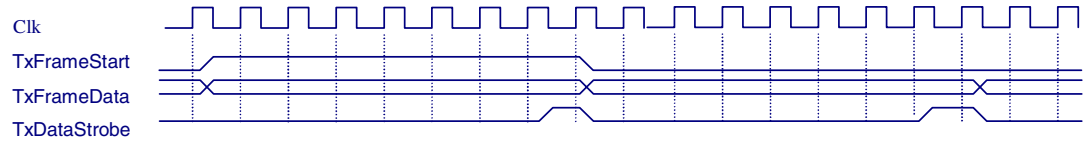


Figure 5: System Interface Transmit Timing 1

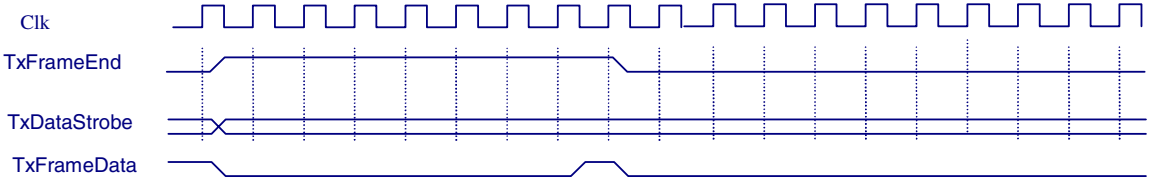


Figure 6: System Interface Transmit Timing 2

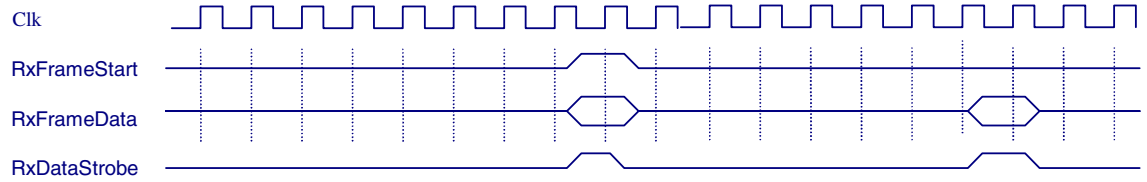


Figure 7: System Interface Receive Timing 1

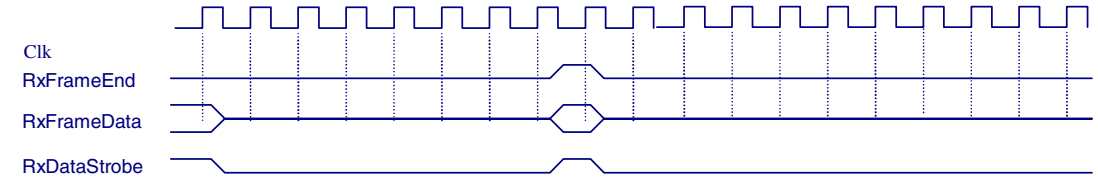


Figure 8: System Interface Receive Timing 2

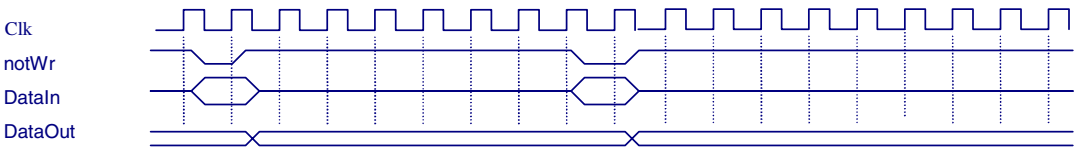


Figure 9: Register Interface Timing

Recommended Design Experience

It is assumed that the user is familiar with HDL based design flows, including VHDL/Verilog language and syntax, component instantiation, synthesis based around the use of scripts, and HDL simulation using testbenches.

It is also assumed that the user is familiar with HDLC protocol, specifically as described in ITU Q.921, X.25 Layer 2 recommendation. Further reading and reference material may be obtained by contacting:

International Telecommunications Union
Place des Nations
CH-1211Geneve 20
Switzerland

Phone: +41 22 730 51 11
Fax: +41 22 733 72 56

ITU Recommendation Q.921.

Related Information

Installation Guidelines

In order to install the delivered files into the users CoreGenerator tool integrated into Xilinx design flows, the user should follow installation instructions contained in the readme text file provided.

Libraries

The included modelsim.do file constructs the libraries required for simulation. It should be executed from within the /verif/<HDL> directory. It also requires that the simulation files remain organized in the delivered directory structure.

Component Instantiation

To use the HDLC core the 'HDLC1' component must be instantiated into the design file. This component definition may be found in the "net/HDLC32.vho" and "net/HDLC32.veo" files located at $xilinx/coregen/ip/xilinx/<modulename_vir>/com/xilinx/ip/<modulename_vir>/<product_family>/net.

Simulation Guidelines

A HDL testbench is provided, together with a set of test utilities, or tools, are provided as part of the core deliverables. A modelsim script, (/verif/<HDL>/modelsim.do) is also provided at $xilinx/coregen/ip/xilinx/<modulename_vir>/com/xilinx/ip/<modulename_vir>/<product_family>/verif.

Testbench and Test Data

The HDL testbench does the following:

- instantiates the core
- generates the input test vectors and applies to the module
- generates or utilizes reset and clock
- generates or utilizes expected results
- automatically checks the actual result from the module against the expected values
- flags any mismatch between the actual outputs and the expected results during simulation

Ordering Information

Xilinx LogiCORE modules are provided under Xilinx LogiCORE standard license agreement. For price and availability information, please contact your local Xilinx Sales Representative.