XAPP345 (v1.0) August 8, 2001

IrDA and UART Design in a CoolRunner CPLD

Summary

This application note illustrates the implementation of an IrDA and UART system using a CoolRunner™ XPLA3 CPLD. The fundamental building blocks required to create a half-duplex IrDA and full-duplex UART interface design is described. The source code for this design is available and can be found in the section "HDL Code" on page 10.

Introduction

IrDA devices provide a walk-up, point-to-point method of data transfer that is adaptable to a broad range of computing and communicating devices. The first version of the IrDA specification (version 1.0) provides communication at data rates up to 115.2 Kbps. Later versions (version 1.1) extended the data rate to 4 Mbps, while maintaining backward compatibility with version 1.0 interfaces. The protocol described in this application note is only for 115.2 Kbps. The 4 Mbps interface uses a pulse position modulation scheme which sends two bits per light pulse.

The IrDA standard contains three specifications. These relate to the Physical Layer, the Link Access Protocol, and the Link Management Protocol. This document provides information on the Physical Layer and does not provide a detailed explanation of the requirements for full IrDA conformity. For more information on IrDA see "References" on page 11.

IrDA System

Figure 1 illustrates the basic hardware building blocks for IrDA communication. The selection of UART interface, RS232, and microcontroller or microprocessor, depends upon the communication speed required. Data rates above 115.2 Kbps require a direct interface to the address and data lines of the microprocessor or microcontroller. Data rates below 115.2 Kbps can be implemented over a UART or RS232 port

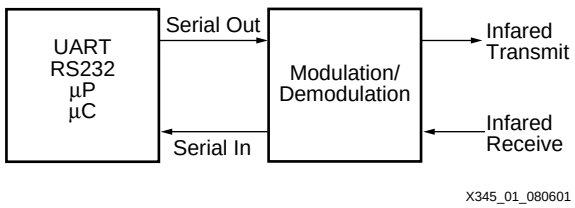

Figure 1: **IrDA Block Diagram**

A UART interface is implemented in this design for data rates up to 115.2 Kbps. The IrDA specification is intended for use with a serial communications controller such as a conventional UART. The data is first encoded before being transmitted as IR pulses. As shown in Figure 2, the serial encoding of the UART is NRZ (non return to zero) encoding. NRZ encoded outputs do not transition during the bit period, and may remain High or Low for consecutive bit periods. This is not an efficient method for IR data transmission with LEDs. To limit the power consumption of the LED, IrDA requires pulsing the LED in a RZI (return to zero, inverted) modulation scheme so that the peak power to average power ratio can be increased. IrDA

requires the maximum pulse width to be 3/16th of the bit period. A 16x clock is required, and counting three clock cycles can easily be done to encode the transmitted data.

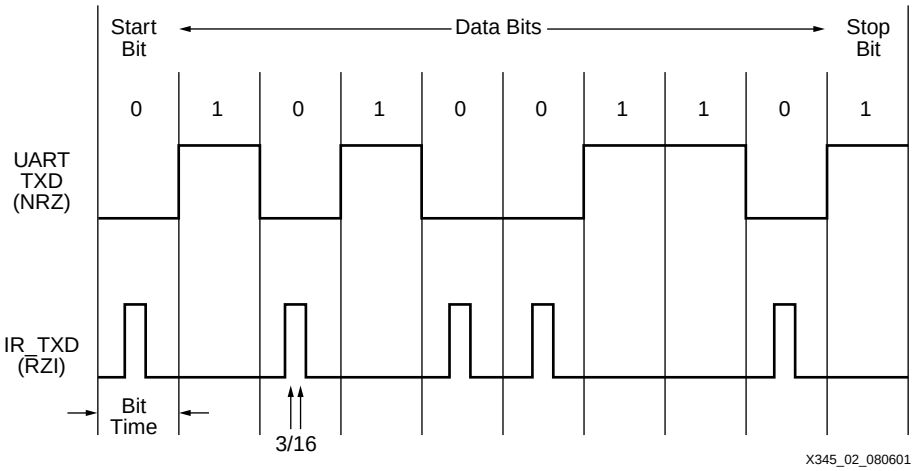

Figure 2: **IrDA 3/16 Data Modulation**

Half Duplex and Latency

The IrDA link cannot send and receive data at the same time. The IrDA link is a half-duplex interface and a time delay must be allowed from when a link stops transmitting until it can receive data again. A time period with a duration of 10 ms must be allowed between transmitting and receiving data. The UART interface design is full-duplex, supporting simultaneous read and write operations from the microprocessor or microcontroller interface.

UART and IrDA Design

Figure 3 illustrates the system architecture for implementing a UART serial port interface with an IrDA module in a CoolRunner CPLD. The UART or a discrete device must provide a 16x clock for the IrDA 3/16 modulation scheme.

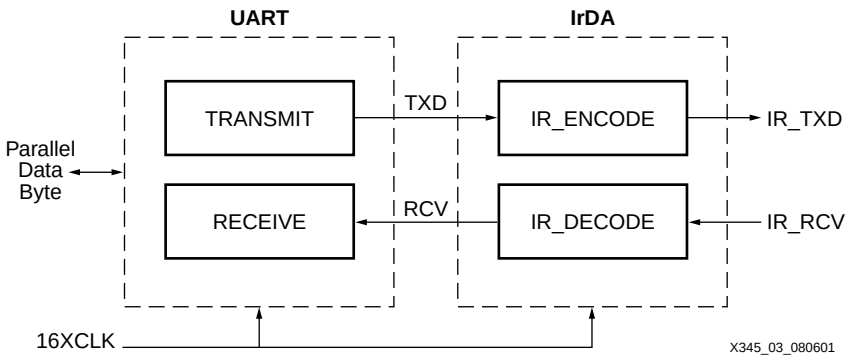

Figure 3: **UART and IrDA Block Diagram**

The Verilog code provided in this design for the UART interface consists of two HDL modules, TRANSMIT and RECEIVE. Data is written to the transmitter and data is read from the receiver through an 8-bit parallel data bus.

The Verilog code provided in this design for the IrDA emulates the operation of the Agilent Technologies HSDL-7000. The IrDA HSDL-7000 consists of logic for both encoding and decoding the transmit and receive data. Each encode and decode operation is driven by the clock, derived from the UART, or supplied from a discrete source. This clock must be initially

configured to cope with the IrDA specified startup data rate of 9.6 Kbps, then adjusted to 16 times the desired baud rate.

UART Interface

Figure 4 illustrates the functionality of the UART interface. The data bus interface to the UART module is 8-bits. Even or odd parity can be selected on the serial data out, SOUT.

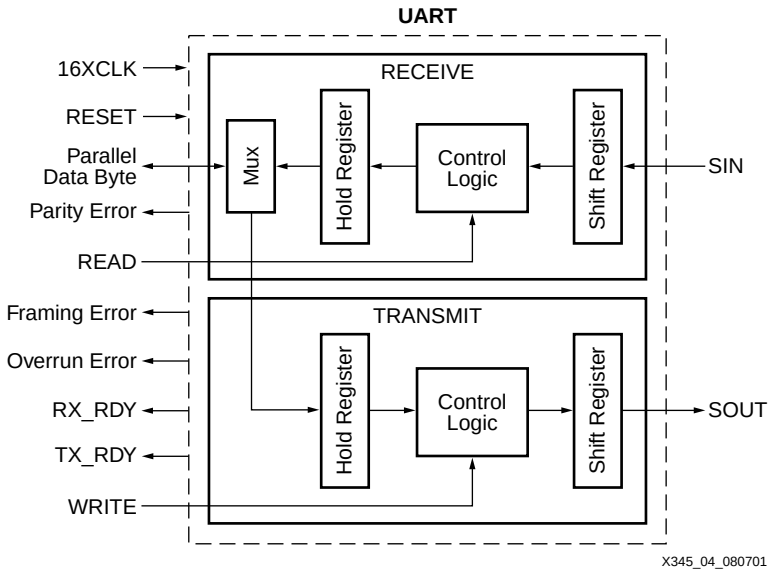

Figure 4: **UART Main Interface Logic**

The serial data out, SOUT follows the format shown in Figure 5.

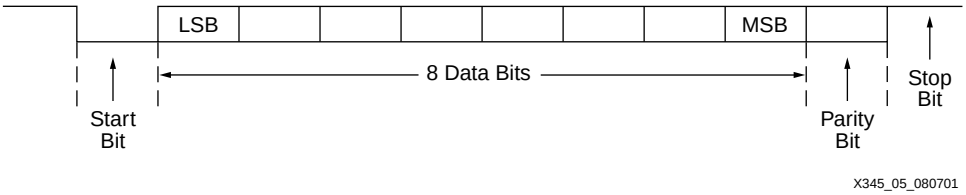

Figure 5: **UART Data Out Format**

UART Transmit Logic

Data transfer in this design is controlled by the system microprocessor or microcontroller. The UART design must interface with the parallel processor bus and necessary control lines. The UART transmit logic consists of interpreting processor write commands, generating the transmit clock, TXCLK, at the desired baud rate, and shifting out data on SOUT. The UART logic must interpret the active Low write signal from the processor and read in data from the data bus. The data is read into the transmit hold register. Once the write signal is de-asserted,

a flag is asserted to start shifting data out on SOUT. Figure 6 illustrates the logic of interpreting the write signal.

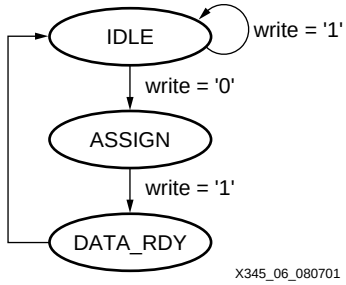

Figure 6: **Assigning Transmit Data**

The second part of control logic for the UART transmitter is dividing the 16x clock to transmit data at the desired baud rate. The transmit clock, TXCLK, is generated using a 3-bit counter that increments on the rising edge of the 16x input clock. TXCLK controls when data changes on the serial data output of the UART. Figure 7 illustrates this logic, TXCLK changes value when the 3-bit counter is equal to zero.

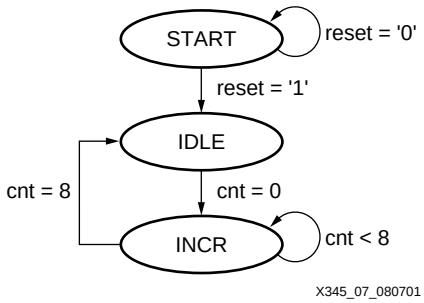

Figure 7: **TXCLK Generate Logic**

The last main portion of the UART transmit logic is shifting out data on SOUT. Figure 8 illustrates the control logic to send data out according to the data format shown in Figure 5. The START TRANSMIT logic sends the start signal out on SOUT. The SHIFT OUT logic shifts the transmit shift register and sends data out on SOUT. When the paritycycle signal is asserted, the

parity bit is transmitted. Once the data and parity has been transmitted, the done bit is sent by the STOP OUT logic.

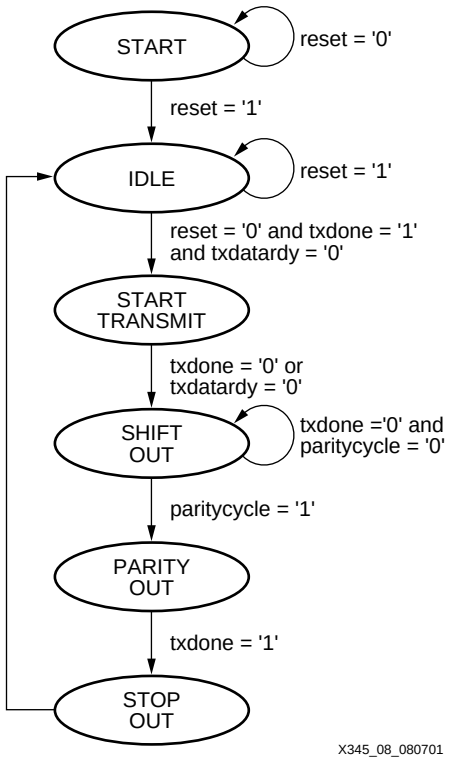

Figure 8: **SOUT Control Logic**

UART Receive Logic

The UART receive logic must interpret the incoming data from the IrDA module on SIN, as well as present a parallel byte of data to the microprocessor or microcontroller in the system. To interpret the incoming SIN data, the receive logic must search for the start bit in the data

stream. The start bit is indicated by an active Low signal for eight clock cycles after a falling edge on SIN.

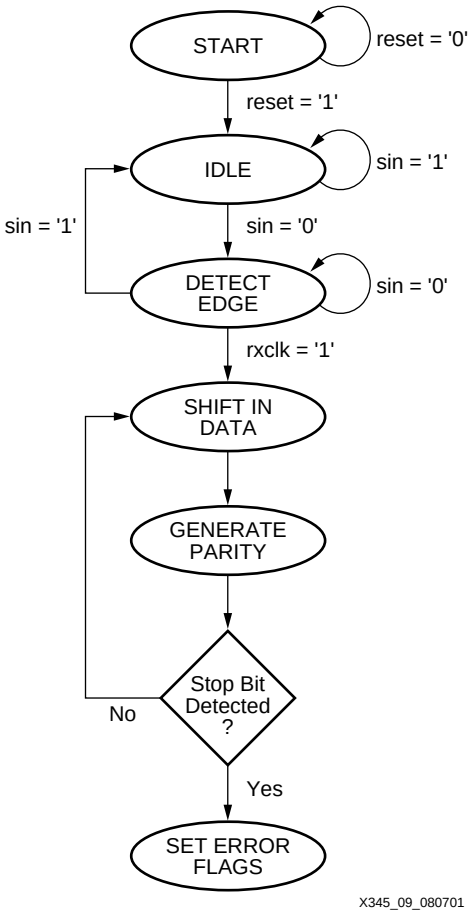

Figure 9: **UART Receive Logic**

A falling edge on SIN is read by the DETECT EDGE logic as shown in Figure 9. To receive data, the receive clock must be centered on the low leading start bit. The receive clock, RXCLK is generated by dividing the 16x clock using a 4-bit counter.

Once a valid start bit is detected, the data is sampled on SIN at each RXCLK rising edge. The receive shift register is shifted with the incoming SIN data. Running parity is generated with each incoming data bit. When a stop bit is detected, any error flags are set. This includes parity, overrun, and framing error flags.

A main function of the UART receive logic is interfacing with the processor. The CPLD detects a valid edge on the READ signal asserted by the processor. The CPLD then places the received parallel data on the system data bus.

IrDA Interface

Figure 10 illustrates the input and output requirements of the IrDA module in this design. RXD and TXD are the serial connections to the UART SIN and SOUT data lines respectively. IRRXD and TXRXD are the IrDA 3/16th pulse signals that are fed into the LED driver/receiver circuitry as shown in Figure 10.

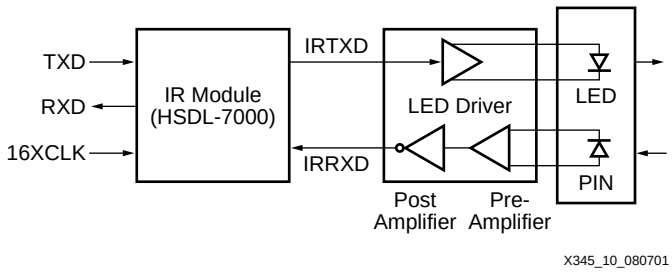

Figure 10: **IR HDL Block Diagram**

The encoding scheme shown in Figure 2 sends a pulse for every space or "0" that is sent on the TXD line. On a High-to-Low transition of the TXD line, the generation of the pulse is delayed for seven clock cycles of the 16XCLK before the pulse is set High for three clock cycles (or 3/16th of a bit time) and then subsequently pulled low.

The decoding scheme shown in Figure 11 seeks a High-to-Low transition of the IRRXD line which signifies a 3/16th pulse. This pulse is stretched to accommodate one bit time (16 clock cycles). Every pulse that is received is translated into a "0" on the RXD line equal to one bit period.

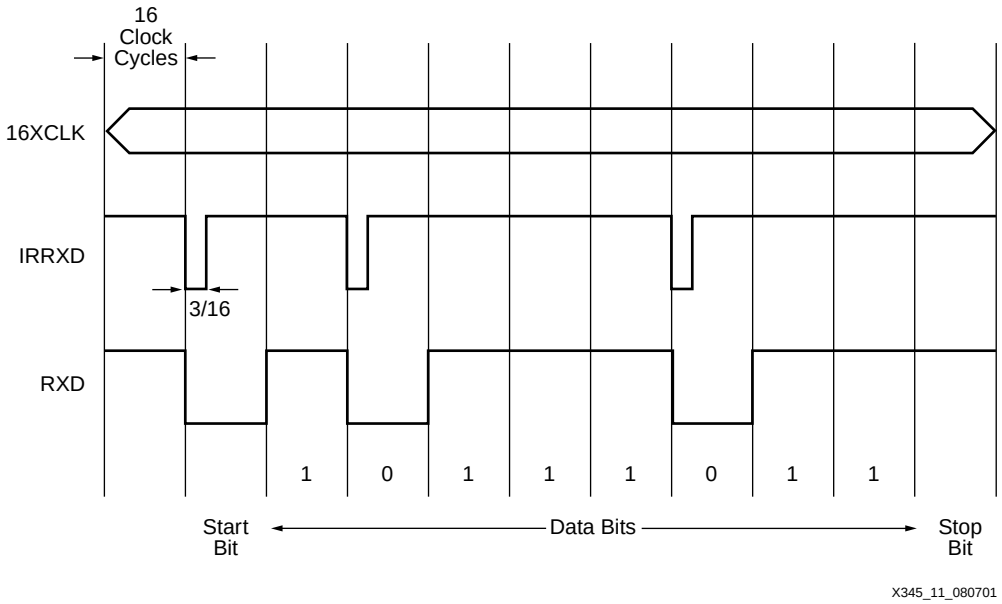

Figure 11: **IrDA Decoding Scheme**

CoolRunner Implementation

The UART and IrDA design was implemented in Verilog as described above. Xilinx Project Navigator was used for compilation, fitting, and simulation of the design in a CoolRunner CPLD. Xilinx Project Navigator, which includes the ModelTech simulation tool, is available free-of-charge from the Xilinx website at www.xilinx.com/products/software/webpowered.htm. The design was targeted for a 3.3V, 128 macrocell CoolRunner XPLA3 CPLD (XCR3128XL-VQ100). The UART and IrDA design utilization is shown in Table 1. These utilizations were

achieved using certain fitting parameters, actual results may vary. As shown, there is area remaining in the CPLD for the implementation of other logic in the system.

Table 1: **UART and IrDA XPLA3 128-Macrocell Utilization**

Resource	Available	Used	Utilization
Function Blocks	8	6	75.0%
Macrocells	128	88	68.75%
Product Terms	384	129	33.60%
Foldback NANDs	64	0	0%
I/O Pins	80	17	21.25%

The Verilog IrDA design can also be targeted as a stand alone module in a 3.3V 32-macrocell CoolRunner XPLA3 CPLD (XCR3032XL). CPLD utilization for implementing the IrDA design is shown in Table 2.

Table 2: **Standalone IrDA XPLA3 32-Macrocell Utilization**

Resource	Available	Used	Utilization
Function Blocks	2	1	50.0%
Macrocells	32	14	43.75%
Product Terms	96	26	27.10%
I/O Pins	32	4	12.50%

Design Verification

The UART/IrDA transmit and receive Verilog design verification has been done through simulation using ModelSim XE in Project Navigator. The design has been verified both functionally and with the timing model generated when fitting in a CoolRunner CPLD. The implemented test bench drove the data, control, and timing necessary to test a transmit operation from the UART to the IrDA output and test the received data from the IrDA and UART modules. Implementation in an actual system may require modification of the control signals used in the source code and test benches provided.

ModelSim Implementation

Notes:

Please refer to XAPP338: Using Xilinx WebPack and ModelTech ModelSim Xilinx Edition as a guide to using ModelSim with Project Navigator. The ModelSim Quick Start demo provides a good first step for getting familiar with ModelSim.

Figure 12 illustrates the test environment for transmitting a data byte using the UART and IrDA modules. Upon receiving an active WRITE signal, the UART TXRDY signal is asserted. Data is sent to the UART module and transmitted as shown on the SOUT signal. TXCLK is the internal divided clock signal for the UART module. IRTXD is the data transmitted from the IrDA module.

The IR transmitted data is in the form as shown in Figure 2 and includes the start bit, eight data bits, a parity bit, and a stop bit in each data transmission.

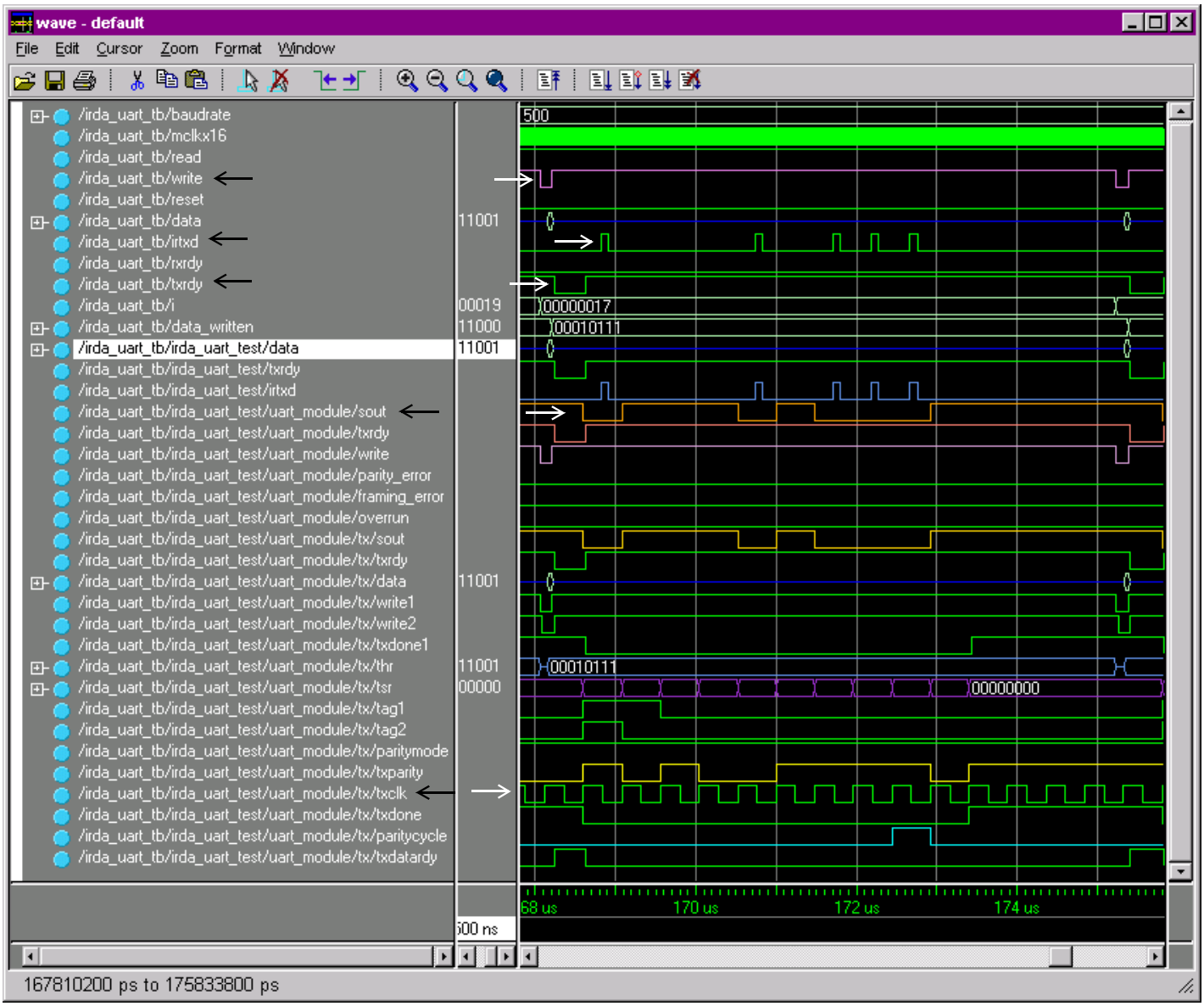

Figure 12: **UART and IrDA Transmit Simulation**

Figure 13 illustrates receiving data on the IrDA IRRXD input and presenting the parallel data byte from the UART to the system. The IrDA receive module recognizes the format of incoming data and sends the translated serial stream to the UART, as illustrated in Figure 11 on the SIN

signal. The UART module shifts the incoming serial data into a holding register. Upon the UART receiving an active READ signal, the UART places the parallel data onto the data bus.

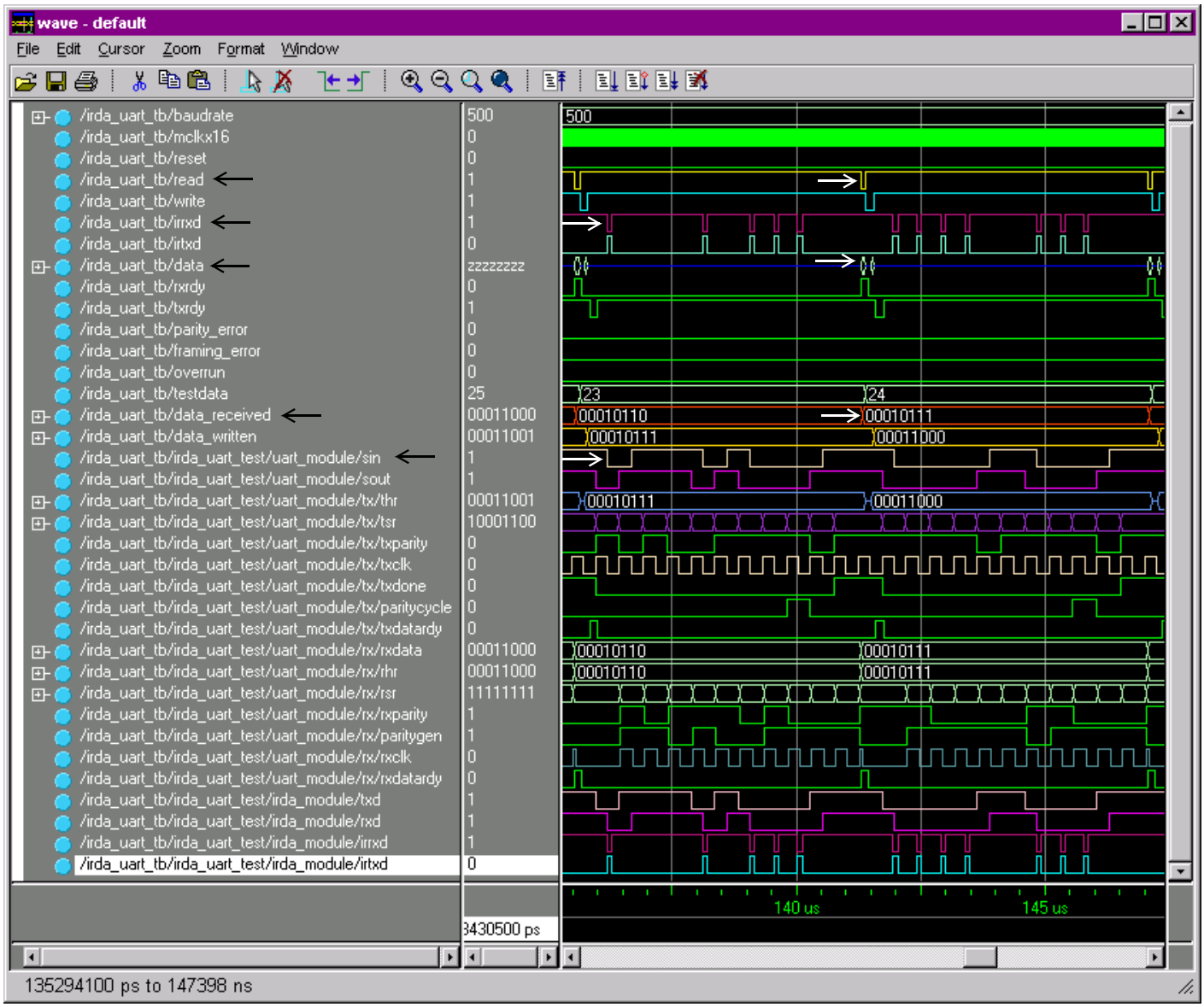

Figure 13: **Transmit and Receive Simulation**

HDL Code

THIRD PARTIES MAY HAVE PATENTS ON IRDA. BY PROVIDING THIS HDL CODE AS ONE POSSIBLE IMPLEMENTATION OF THIS DESIGN, XILINX IS MAKING NO REPRESENTATION THAT THE PROVIDED IMPLEMENTATION OF THIS DESIGN IS FREE FROM ANY CLAIMS OF INFRINGEMENT BY ANY THIRD PARTY. XILINX EXPRESSLY DISCLAIMS ANY WARRANTY OR CONDITIONS, EXPRESS, IMPLIED, STATUTORY OR OTHERWISE, AND XILINX SPECIFICALLY DISCLAIMS ANY IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, OR FITNESS FOR A PARTICULAR PURPOSE, THE ADEQUACY OF THE IMPLEMENTATION, INCLUDING BUT NOT LIMITED TO ANY WARRANTY OR REPRESENTATION THAT THE IMPLEMENTATION IS FREE FROM CLAIMS OF ANY THIRD PARTY. FURTHERMORE, XILINX IS PROVIDING THIS REFERENCE DESIGNS "AS IS" AS A COURTESY TO YOU.

XAPP345 - **http://www.xilinx.com/products/xaw/coolvhdlq.htm**

Conclusion

IrDA is a low cost, walk-up, point-to-point method of IR communication protocol used in applications ranging from laptops to phones to fax machines. This design is an example implementation of an IrDA interface for data ranges less than 115.2 Kbps connected to a UART interface. Version 1.1 extends the IrDA specification to 4 Mbps and can be implemented using pulse position modulation.

References

1. Infrared Data Association (IrDA).
2. Hewlett Packard IrDA Data Link Design Guide.
3. Agilent HSDL-7000 Data Sheet: IR 3/16 Encode/Decode IC.
4. Agilent Application Note 1119: IrDA Physical Layer Implementation for Agilent Technologies' Infrared Products.
5. Xilinx Application Note XAPP341: UARTs in Xilinx CPLDs.
6. QuickLogic Application Note: QAN20. Digital UART Design in HDL.
7. Faulkner, Lawrence. IrDA "More than Wireless".
8. Evans, David James. IrDA Applications in CoolRunner CPLDs.

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
08/08/01	1.0	Initial Xilinx release.