XAPP341 (v1.2) November 28, 2000

UARTs in Xilinx CPLDs

Summary

This application note provides a functional description of VHDL and Verilog source code for a UART. The code is used to target the XC95144 and XCR3128XL CPLDs. The functionality of the UART is discussed. To obtain the VHDL (or Verilog) source code described in this document, go to section "VHDL (or Verilog) Code Download" on page 3 for instructions.

Introduction

The Universal Asynchronous Receiver Transmitter (UART) is the most widely used serial data communication circuit ever. UARTs allow full duplex communication over serial communication links as RS232. The reference VHDL and Verilog code implements a UART in Xilinx CPLDs. UARTs are available as inexpensive standard products from many semiconductor suppliers, making it unlikely that this specific design is useful by itself.

The basic functions of a UART are a microprocessor interface, double buffering of tranmitter data, frame generation, parity generation, parallel to serial conversion, double buffering of receiver data, parity checking, serial to parallel conversion. The frame format of used by UARTs is a low start bit, 5-8 data bits, optional parity bit, and 1 or 2 stop bits. Some UARTs include modem interface signals. These are pass-through signals which are not done in this design.

The organization of this application note is to provide a section on the receiver and then the transmitter.

The frame format for data transmitted/received by a UART is given in Figure 1. It consists of a high idle state of the line. A character is from 5-8 data bits. The start bit isLow and the single stop bit is High.

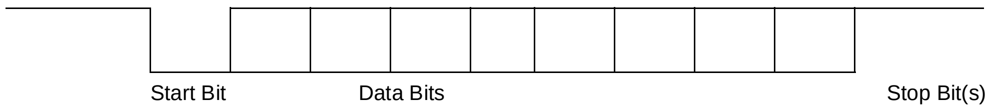

Figure 1: **Frame Format for UART Transmitted/Received Data**

Receiver

The signals used by the receiver are given in Table 1. The receiver interfaces to the data bus **dout[7:0]** with the **rdn** signal. The controller can generate a rdn strobe if data_ready is true. The receiver is double buffered, allowing data to be held in the buffer register **rbr[7:0]** while data is shifted in serially into the receiver shift register **rsr[7:0**]. This provides the controller flexibility with bus read operations.

The receiver detects the character frame and strips the start and stop bits. The **no_bits_rcvd** variable controls the word size.

The clkdiv[3:0] register is used to control the time at which the data is decoded. The receiver uses the 16x local clock and decodes the value of start, data, and stop nits in the center of the data cells. To do this, the start bit initializes a count operation using clkdiv[3:0]. After detecting the low going edge on the start bit, the receiver counts the 16x clock to 8 and decodes, or samples the value of the signal. The clkdiv[3:0] register is then reset to 0, and subsequently counts the 16x clock to 16. This provides center sampling for the data and stop bits.

Three error detection signals are commonly used in UARTs. Parity Error indicates whether an even or odd number of "1s" are present in a data work. Overrun Error indicates whether the receive buffer register is overwritten by the receive shift register prior to the controller reading the receiver buffer register. Overrun Error is not implemented in the VHDL/Verilog source. Framing Error indicates if the stop bit is not High.

Table 1: **Receiver Signals**

Signal	Direction	Function
rst	Input	Resets.
clk16x	Input	16x input clock.
rdn	Input	Read strobe.
dout[7:0]	Output	Output data bus.
framing_error	Output	Framing error status signal.
parity_error	Output	Parity error status signal.
rbr[7:0]	Internal	Receiver buffer register - accepts data from data[7:0] and transfers it to rsr[7:0].
rsr[7:0]	Internal	Receiver shift register - accepts data from rbr[7:0] and transfers it to sdo.
no_bits_rcvd	Internal	Tracks character size and sequences receiver operation.
clk1x_enable	Internal	Enable signal for registers clocked by clk1x.
clk1x	Internal	1x clock used for internal operations.

Transmitter

The signals used by the transmitter are given in the table below. The transmitter interfaces to the data bus with the transmitter buffer register empty (**tbre**) and the **wrn** signals. The controller can generate a **wrn** strobe if tbre is high. The transmitter is double buffered, allowing data on din'7:0] to be written to the buffer register **tbr[7:0**] while data is being shifted out of the shift register **tsr[7:0**]. The transmitter generates a frame which consists of the idle state (high on sdo), low start bit, eight data bits, and a stop bit.

The no_bits_sent controls the word size and sequences the transmitter operations. To change the word size, change the value of no_bits_sent in the verilog source.

Table 2: **Transmitter Signals**

Signal	Direction	Function
rst	Input	Resets wrn1,wrn2,no_bits_sent, clkdiv[3:0],tbr[7:0],tsr[7:0]
clk16x	Input	Local reference clock 16X the data rate
wrn	Input	Control signal which strobes data from din[7:0] to tbr[7:0]
sdo	Output	Serial data output
tbre	Output	Status signal indication that the transmitter buffer register is empty
no_bits_sent	Internal	Controls word_size and sequences transmitter operation
clk1x_enable	Internal	Enables internal clock clk1x.
tbr[7:0]	Internal	Accepts data from din[7:0] and transfers data to tsr[7:0]
tsr[7:0]	Internal	Receives data from tbr[7:0] and shifts to sdo
clkdiv[3:0]	Internal	Used in generation of internal clock

VHDL (or Verilog) Code Download

VHDL (or Verilog) source code and test benches are available for this design. THE DESIGN IS PROVIDED TO YOU "AS IS". XILINX MAKES AND YOU RECEIVE NO WARRANTIES OR CONDITIONS, EXPRESS, IMPLIED, STATUTORY OR OTHERWISE, AND XILINX SPECIFICALLY DISCLAIMS ANY IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, OR FITNESS FOR A PARTICULAR PURPOSE. This design has not been verified on hardware (as opposed to simulations), and it should be used only as an example design, not as a fully functional core. XILINX does not warrant the performance, functionality, or operation of this Design will meet your requirements, or that the operation of the Design will be uninterrupted or error free, or that defects in the Design will be corrected. Furthermore, XILINX does not warrant or make any representations regarding use or the results of the use of the Design in terms of correctness, accuracy, reliability or otherwise.

XAPP341 - http://www.xilinx.com/products/xaw/coolvhdlq.htm

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
03/31/00	1.0	Initial Xilinx release.
04/17/00	1.1	Added VHDL Code Download link.
11/28/00	1.2	Changed XCR3128 to XCR3128XL in Summary.