A Serial Communication Link Between FPGAs

A synchronous point-to-point serial communication link between two FPGAs.

by Paul Gigliotti, Field Applications Engineer, Xilinx, paul.gigliotti@xilinx.com

This article describes a method for transmitting serial data between FPGAs, as shown in Figure 1. The data is transmitted in packets to allow for error detection and re-synchronization of the serial data link. This example design assumes a transfer of 32 bits, but the size can vary as needed.

A Serial Link Between FPGAs

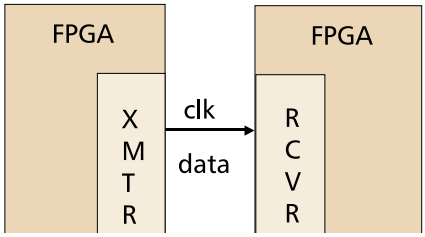

Figure 1

The Start Sequence and Zero-Insertion

The basic trick is to begin the transaction with a start sequence, and ensure that the sequence does not occur within the remainder of the packet. This is easier than it sounds; it's just a matter of selecting the right start sequence. In this example, I transmit a start sequence of 111110 and then insert a zero into the data whenever four ones have occurred, to ensure that the start sequence is not repeated within the data, as shown in Figure 2. The receiver circuit, on the other hand, waits until it sees the start sequence before receiving data, and while receiving data, it throws away the zero that follows four ones. Furthermore, while receiving data after the start sequence has been detected, if five ones in a row are received then an error has occurred.

Zero Insertion

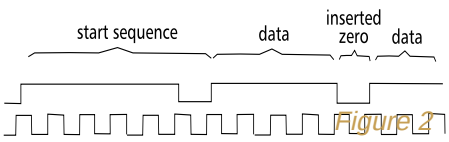

Figure 2

Transmitter State Machine

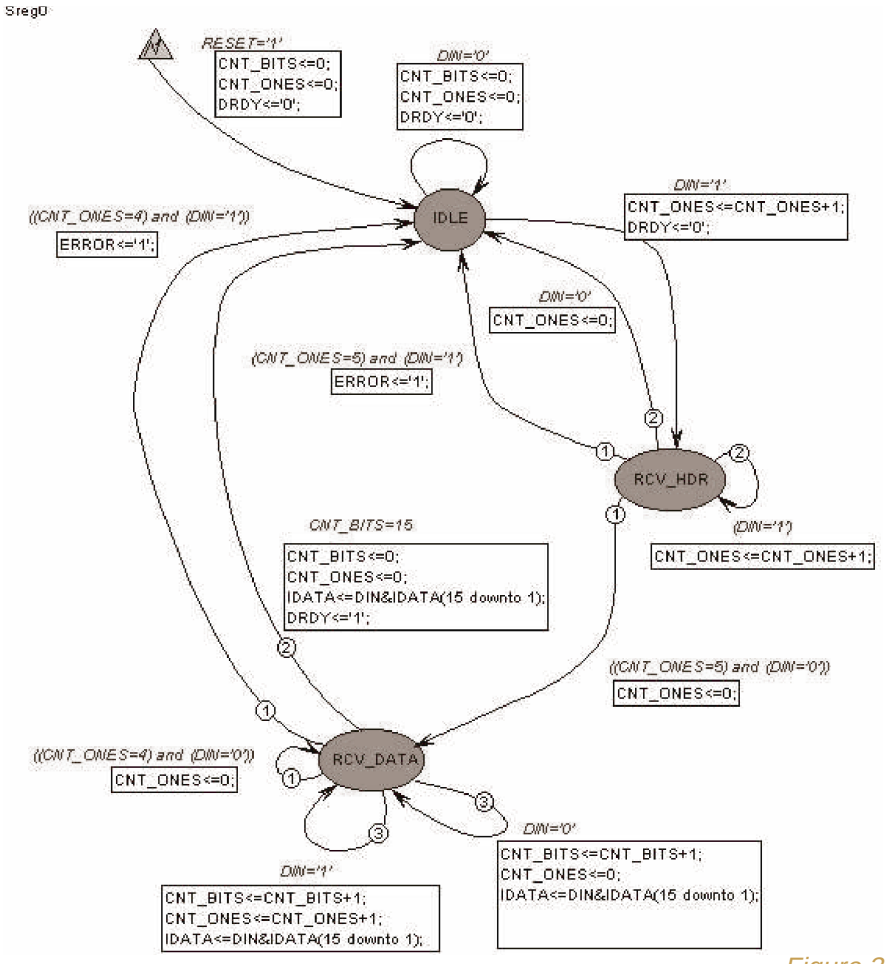

Figure 3

This application offers a simple yet effective method for achieving synchronization between the transmitter and receiver circuits. You can easily extend the design to append parity and CRC bits to the transmitted data.

Transmitter State Machine

Data is transmitted whenever the state machine detects a "Send" request, as shown in Figure 3. The state machine begins by transmitting the header or start sequence and then follows with the data. Notice, that while transmitting the data, it counts the number of "ONES-IN-A-ROW' that it has transmitted, and if the limit has been reached, a zero is inserted and transmitted as part of the data.

Receiver State Machine

The receiver state machine is constantly looking for a start sequence, as shown in Figure 4. Once a start sequence has been received, it then begins receiving data, looking for inserted zeros and flagging errors as they occur.

Simulation Results

Figure 5 shows the results of the transmitter and receiver working in tandem with an FFFF and then a 0000 being sent/received

Receiver State Machine

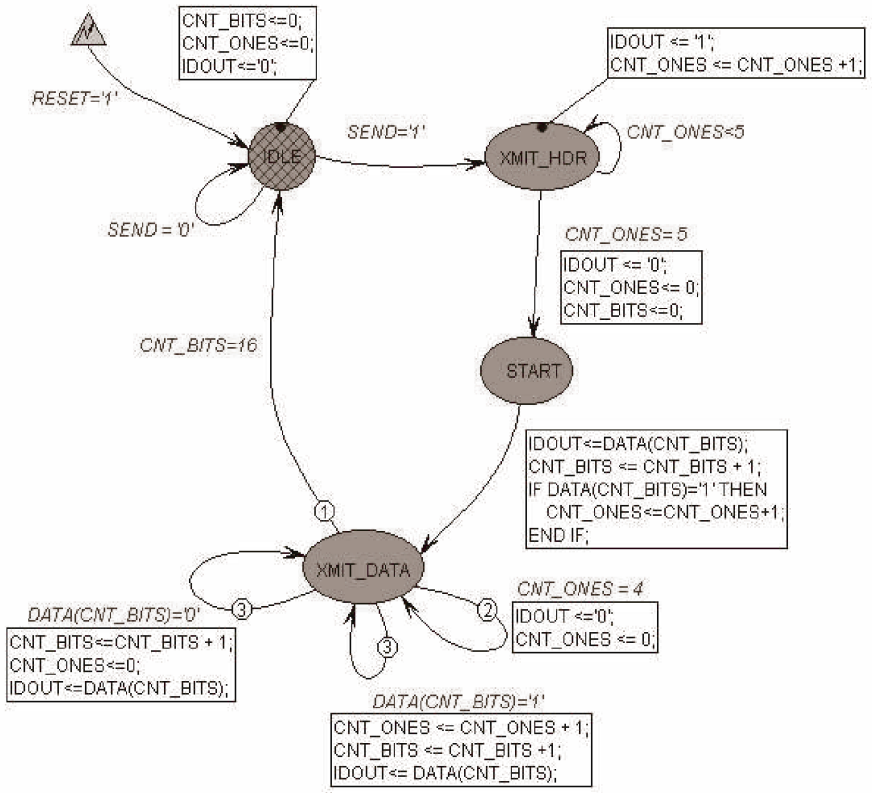

Figure 4

Sample Transmission

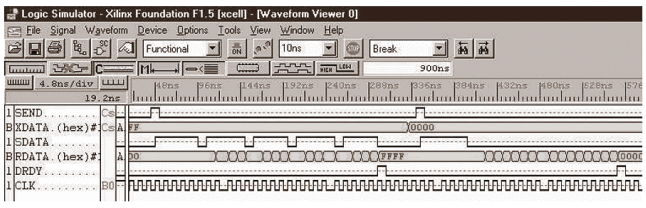

Figure 5

Conclusion

This application offers a simple yet effective method for achieving synchronization between the transmitter and receiver circuits. You can easily extend the design to append parity and CRC bits to the transmitted data.