SERDES Interfaces in an FPGA World: What Do I Need To Know to Get Started?

By understanding the abilities of both FPGA products and SERDES interfaces, a design team can increase the reliability and functionality of its products while reducing time to market.

By Dan Sides & Bertrand Leigh, Lattice Semiconductor

For some years now, increasing levels of chip functionality and data throughput requirements have propelled the chip industry in a migration from lower data rate

[1]

parallel connections to higher speed serial connections. The concept is called SERDES (abbreviation for "Serializer-Deserializer"), and it involves transmitting serialized data over high speed, differential pairs rather than lower speed parallel buses. An excellent example is the replacement of a legacy 32-bit, 64MHz PCI bus achieving 2.112 Gb/s, with a single PCI-Express lane that achieves an overall data rate of 4 Gb/s using only 4 wires running at 2.5 GHz. Simply put, SERDES protocols allow higher data rates with fewer pins.

Typical SERDES FPGA Implementation

Figure 1 illustrates the variety of possible SERDES interfaces in a complex FPGA implementation. This example shows a high-performance board with a network processor at the heart of the system. SERDES applications are highlighted in purple. Chips that could be implemented with FPGAs are shown in yellow.

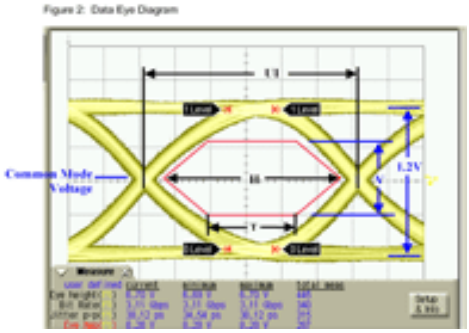
Types of SERDES

There are two basic categories of SERDES interfaces: source synchronous (SS) protocols and clock data recovery (CDR) protocols. The fundamental difference between the two is how each implements clocking. Source synchronous interfaces have a discrete clock signal that accompanies the transmitted data. CDR, on the other hand, has no separate clock signal but instead embeds the clock in the switching of the data. That is to say, a CDR receiver will phase lock on the data

signal itself to derive the clock. The following table outlines some of the basic differences between the two. CDR protocols generally run at higher speeds and for longer distances, and therefore present greater design challenges. For that reason, this article will focus mainly on CDR issues.

Clock Data Recovery Basics

As the name implies, a CDR receiver must recover the embedded clock from the data. More accurately, the clock is derived from the switching of the data signaling. A CDR transmitter begins

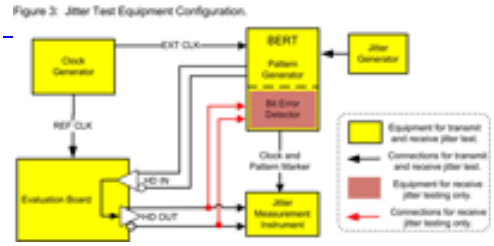


[2]

by serializing its data, then converting the data to an 8b/10b (or similar) encoding scheme. This encoding takes 8-bit data and converts it to a 10-bit symbol. The 8b/10b encoding provides an equal number of 0s and 1s transmitted on the data line. This mitigates inter-symbol interference (differential signal drift toward a dominating logic level) and provides enough data edges to allow the receiver to phase lock on the incoming stream. The transmitter will multiply the system clock to the transmit bit rate and send the 8b/10b data on the TX differential pair at that rate. A CDR receiver's job begins with phase locking onto the RX differential bit stream. The receiver then bit aligns the data per this recovered clock. Next, the data is word aligned in the receiver's reference clock. Finally, the data is 8b/10b decoded and is ready for system use. In CDR systems, the transmitting and receiving systems typically have totally independent system clocks. It is critical that both clocks be within a specified variation requirement. This threshold is on the order of a few hundred PPM.

CDR Circuits and Jitter

The main design challenge of CDR interfaces is jitter, which is simply the placement of an actual data transition versus where it ideally is expected. Total jitter (TJ) is a composite of deterministic jitter and random jitter. Most jitter is deterministic and its components include intersymbol interference, crosstalk, duty-cycle distortion and periodic jitter (e.g. interference from a switching power supply). Random jitter is often the byproduct of



[3]

semiconductor thermal issues and is far less predictable. The transmit reference clock, transmit PLL, serializer and high-speed output buffers all contribute to transmit jitter. Transmit jitter usually is specified in terms of a percentage of a unit-interval, or UI, for a given bit period or data eye (explained in the next section). For example, a transmit jitter of .2 UI indicates that jitter comprises 20% of the bit period. For transmit jitter, lower UI numbers are better because they indicate less jitter.

Similarly, a CDR receiver will specify a maximum amount of jitter that it can tolerate at a given bit rate. A typical bit error rate criterion is 1e-12 (or one error per 1e12 bits, trillion). Receive jitter also is specified in unit interval (UI). A larger UI indicates that a receiver can tolerate more jitter. A typical receiver specification is .8 UI, meaning that up to 80% of the bit period can be noise and the receiver still will be able to receive the data reliably. Jitter is usually quantified in terms of a statistical bell distribution that has ideal edge placement at its peak.

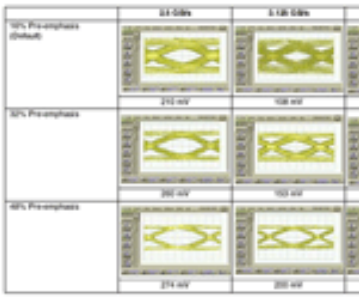
SERDES Testing and Eye Diagrams

Since jitter is a major challenge in SERDES systems, it is also a major focus of test and measurement. Jitter is measured by attaching a high performance oscilloscope to SERDES signals and viewing a "data eye diagram" (or just "eye diagram"). An eye diagram is simply an overlay of many state transitions for a given differential pair. The sampling window is wide enough to include two crossover points in the diagram. The resulting picture looks like an eye and it provides an intuitive visualization of signal quality and jitter. In general, the wider the eye opening, the better the signal.

Figure 2 shows a typical eye diagram as seen on an oscilloscope. In this diagram, V measures the height of the eye opening, versus the total voltage swing (from logic 0 to logic 1) of 1.2 V. There are three width (or time) measurements: UI measures the full bit period, H measures the full opening at the common mode voltage and T measures the width at the minimum and maximum trip voltages. Larger values for H, T and V indicate a wider eye, and, therefore, a better signal and less jitter.

Jitter Measurement Setups

To check transmit jitter, a test pattern is generated by a bit error rate tester (BERT) and sent to an evaluation board's SERDES receive ports. Similarly, a clock generator is connected to the evaluation board's SERDES clock. The FPGA is



[4]

configured to internally loopback the channel under test, so the received test pattern is

transmitted on the TX pins. An oscilloscope is connected to the TX SERDES connectors so that a transmit jitter eye diagram can be analyzed. All evaluation board SERDES connections are with 50 Ω SMA connections. This configuration is shown in Figure 3.

Receive jitter tolerance is measured by inducing jitter at the FPGA SERDES receive port and monitoring the bit errors produced on the looped back SERDES output. The setup (shown in Figure 3) begins by connecting a jitter generator to a BERT pattern generator. Then the BERT generator sends a pseudo-random pattern sequence (PRBS) to the evaluation board's SERDES SMA input. This configuration will allow the engineer to introduce jitter to the SERDES RX port in a controlled manner. The FPGA is configured for loopback and the SERDES TX port is connected to the bit error detector port on the BERT. The engineer then can introduce jitter and watch the bit error ratio (BER) produced by the FPGA. When the BER goes out of specification, then the engineer knows that the jitter tolerance threshold has been exceeded. This number should be equal to or greater than the specified receive jitter tolerance for the FPGA.

High Speed Backplane Performance Measurements

An engineer can measure an FPGA's backplane SERDES link transmit characteristics by driving an FPGA's TX signal through a backplane configuration, then analyzing an eye diagram on the backplane output. The setup first uses a BERT pattern generator to feed a PRBS bit stream into an FPGA evaluation board's SMA RX port. With the FPGA configured for loopback, the PRBS will appear on the evaluation board TX port and will be driven into a coax cable that feeds a backplane board such as a Tyco XAUI Z-Pack HM-Zd. The

[5]

backplane then feeds the bit stream into another length of coax cable that connects to the oscilloscope. The system requirements dictate test parameters such as PRBS pattern choice, trace lengths of the backplane and FPGA evaluation board, coax cable lengths, pre-emphasis and equalization settings, operating temperatures and VCC.

Figure 4 shows nine eye diagrams sampled with this test setup using a LatticeSC FPGA. The number below each eye is the measured eye height (measurement V from Figure 2). Note that increasing pre-emphasis improves the eye diagrams at each bit rate. Pre-emphasis is the SERDES transmitter compensation for signal attenuation occurring in the cabling and backplane. The required eye height for this LatticeSC FPGA is 85 mV, so the eye diagram taken at 3.8 Gb/s with 16% pre-emphasis is the only sample that does not meet the requirement.

In conclusion, it should be mentioned that SERDES interfaces add a dimension to the FPGA product selection process. Although a detailed discussion is beyond the scope of this article, when selecting an FPGA the engineer should consider the

possible number of channels, configuration flexibility of these channels, interface speeds, SERDES IP (e.g.PCS), transmission specifications and electrical requirements. Both FPGA products and SERDES interfaces are growing in speed and marketplace presence. By understanding the abilities and challenges of both, a design team can increase the reliability and functionality of its products while reducing time to market.

About the Author

Dan Sides is an applications engineer for Lattice Semiconductor. He has 15 years experience in FPGA and ASIC design and applications engineering. Contact Dan at dan.sides@latticesemi.com *[6]*

Bertrand Leigh is director of applications engineering at Lattice Semiconductor Corporation. Bertrand has more than 15 years of experience in the PLD industry. He can be reached at bertrand.leigh@latticesemi.com *[7]*

Source URL (retrieved on *12/08/2013 - 11:21am*):
http://www.wirelessdesignmag.com/product-releases/2007/06/serdes-interfaces-fpga-world-what-do-i-need-know-get-started

Links:
[1] http://www.wirelessdesignmag.com/sites/wirelessdesignmag.com/files/legacyimages/0706/feat4_1_lrg.gif
[2] http://www.wirelessdesignmag.com/sites/wirelessdesignmag.com/files/legacyimages/0706/feat4_2_lrg.gif
[3] http://www.wirelessdesignmag.com/sites/wirelessdesignmag.com/files/legacyimages/0706/feat4_3_lrg.gif
[4] http://www.wirelessdesignmag.com/sites/wirelessdesignmag.com/files/legacyimages/0706/feat4_4_lrg.gif
[5] http://www.wirelessdesignmag.com/sites/wirelessdesignmag.com/files/legacyimages/0706/feat4_5_lrg.gif
[6] mailto:dan.sides@latticesemi.com
[7] mailto:bertrand.leigh@latticesemi.com