WP425 (v1.0) October 8, 2012

Transceiver Bit Error Isolation Methodology

By: Martin Gilpatric

The ecosystem for high-speed serial interfaces is pushing towards ever-increasing line rates while maintaining or reducing costs for the system as a whole. As a result, designers are pushing the system margins of their links to the limit.

Xilinx has responded to this trend by adding features both to the transceivers themselves and to the design environment. This enables the savvy designer to push the limits of performance quickly and accurately while continuing to ensure robust link reliability.

© Copyright 2012 Xilinx, Inc. Xilinx, the Xilinx logo, Artix, ISE, Kintex, Spartan, Virtex, Vivado, Zynq, and other designated brands included herein are trademarks of Xilinx in the United States and other countries. PCI, PCI Express, PCIe, and PCI-X are trademarks of PCI-SIG. All other trademarks are the property of their respective owners.

Introduction

When developing a system that pushes the limits of performance in real-world environments where board space, cost, and time are all competing, problems are inevitable. Fortunately, Xilinx provides an array of features and tools to help identify the source of these problems and address them as quickly as possible.

This white paper describes the typical errors that a serial system is likely to experience and what these errors mean. It then walks through the most direct methods for identifying the causes of those failures through the use of digital tools, including digital simulation, the analysis features built into the Xilinx® transceivers themselves, and a debug tool unique to Xilinx — the ChipScope™ analyzer and the IBERT in-system tester contained within the ChipScope Pro Serial I/O toolkit. Finally, techniques for analyzing the fidelity of the system through hardware measurements are described.

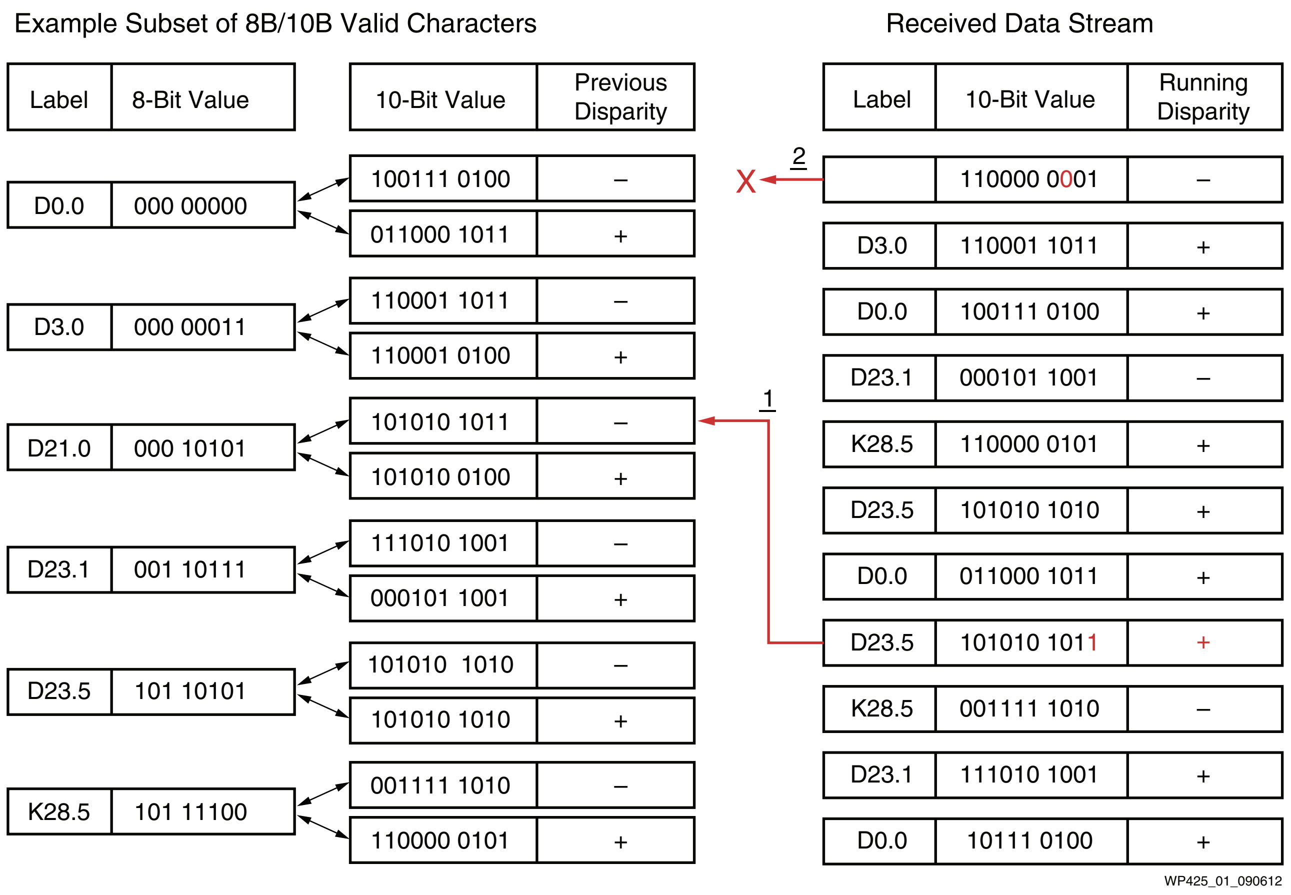
Impact of Hardware Issues

Any designer implementing a link expects to interface with that link at a relatively high level, where any report of errors, including how and where the errors actually happened, are thoroughly obscured. Examples of this include receiving a NACK in PCI Express® systems or a CRC error in many Ethernet implementations. While these broad error-reporting techniques are important, they do not provide the granularity of word-by-word error reporting that an encoding scheme such as 8B/10B provides. It is important to segment these high-level reports into the smallest detectable errors before continuing with debug.

In a high-performance system, accurate reception of the data being transmitted is paramount. To combat forwarding invalid data to other portions of a system, a variety of methods can be used to detect errors, from symbol-specific 8B/10B errors to high-level parity or CRC error checking. Regardless of the method used for detecting an error, analyzing how the error occurs can yield important information on how to correct it.

A protocol that uses 8B/10B encoding offers the easiest and one of the most granular forms of error detection through Not-in-Table errors and Disparity errors. Many of the most popular protocols such as PCIe, Serial RapidIO, and the Xilinx Aurora protocol, utilize 8B/10B encoding, making familiarity with these error types very useful. Both error types leverage the basic principles of 8B/10B encoding:

Each 8-bit character can map to one of two 10-bit values. Each value represents a positive or negative disparity, which is used to maintain a DC balance on the serial link. Not-in-Table errors occur when a received 10-bit value does not correspond to any 8-bit character. Disparity errors result when a received 10-bit value does not match the expected disparity based on previous data. See Figure 1.



Label	8-Bit Value	10-Bit Value	Previous Disparity
D0.0	000 00000	100111 0100	–
		011000 1011	+
D3.0	000 00011	110001 1011	–
		110001 0100	+
D21.0	000 10101	101010 1011	–
		101010 0100	+
D23.1	001 10111	111010 1001	–
		000101 1001	+
D23.5	101 10101	101010 1010	–
		101010 1010	+
K28.5	101 11100	001111 1010	–
		110000 0101	+

Label	10-Bit Value	Running Disparity
	110000 0001	–
D3.0	110001 1011	+
D0.0	100111 0100	+
D23.1	000101 1001	–
K28.5	110000 0101	+
D23.5	101010 1010	+
D0.0	011000 1011	+
D23.5	101010 1011	+
K28.5	001111 1010	–
D23.1	111010 1001	+
D0.0	10111 0100	+

Figure 1: **Disparity Error and Not-in-Table Error in an 8B/10B-Encoded Data Stream**

Notes:

1. A Disparity Error occurs when a bit flip changes the symbol to the incorrect disparity, usually assigned to a different 8-bit character.
2. A Not-in-Table Error occurs when a bit flip changes the symbol to a value that does not correspond to an 8-bit value.

While it is unusual for errors to pass these checks without being detected, any such errors are then detected via higher protocol-level checks. Not-in-Table and disparity errors provide information about the health of a link. Simulation, logic analyzers, and the ChipScope tool can be used to determine if errors occur only on particular data patterns or if there are batches of errors that can be correlated to other in-system events.

Other encoding schemes, such as 64B/66B or scrambling, do not have a similar encoding-level method of detecting bit errors and must rely on higher-level error checks, such as a Cyclical Redundancy Check (CRC). A CRC calculates a value by running previous data through a linear feedback shift register (LFSR) whose final value is then appended to the end of a packet and compared against the receiver's calculated CRC value. If they do not match, it is determined that an error occurred somewhere in the preceding packet. Compared to 8B/10B encoding, the granularity of error reporting is much lower when using a CRC or some other similar error-checking methodology.

Regardless of the encoding scheme or error detection method used, evaluating where in the system an error occurred — and why — is necessary to correct the error. Xilinx provides an array of tools that not only prevent errors from occurring, but identify the sources of the errors in an actual system. Digital simulation examples, ChipScope debug tools, board checklists and examples, and IBIS-AMI simulation models all play important parts in finding and correcting the sources of transceiver errors as quickly as possible.

Bit Error Evaluation

Bit errors fall into two categories:

- Those caused by signal integrity impeding on the receiver's ability to detect the correct `1` or `0`
- Those caused by some cataclysmic failure of the link, either due to logical failures or some infrequent aggression to the signal integrity of the link.

Random bit errors are interspersed throughout the data stream and are difficult to correlate to any particular event, other than perhaps to a particular data pattern. Signal integrity problems, to which they are typically related, can be evaluated through either simulation or some sort of in-system analysis, such as a high-speed oscilloscope or the Xilinx internal eye-scan functionality available in the 7 series transceivers. Many times, link tuning is necessary to correct for these errors, and the ChipScope IBERT core provides an easy way to approach signal integrity problems in an existing system.

Bursts of errors usually occur when something interrupts the reception of the incoming data stream and a large portion of the received characters are corrupted. Identifying how often the bursts of errors occur and how the link recovers from the errors often yields the best results when identifying the cause of such errors. Since this type of error can be related to logic, clocking, or signal integrity, it is important to collect a wide variety of information at the outset of the debug process.

Digital Simulation

Serial design is notorious for creating potential sources of the issue that appear to impact the error but are not directly related to the source of the problem. By using digital simulations of the transceiver and surrounding logic, the designer can isolate where in the transmission path the bit errors originated. Beyond initialization, many protocols implement flow control of some sort that needs to be handled by the FPGA logic.

If a rigorous digital simulation is not performed and problems are seen when the design is placed into hardware, there is a tendency to assume that the serial link itself is at fault, rather than perform due diligence on the logic or clocking. Using both HDL and timing-driven digital simulations to verify that the logic is correct is a fast method of gaining better visibility into the operation of the logic, while obfuscating the analog interconnect itself. Table 1 gives a selection of ports on the 7 series GTX transceiver that can be helpful in understanding where and why an error occurred.

Table 1: **Selection of Ports on the 7 Series GTX Transceiver**

Port	In/Out	Debug Comments
Generic		
QPLLREFCLKLOST /CPLLREFCLKLOST	Out	Key indicator that reference clock routing or selection is not correct.
QPLLLOCK/CPLLLOCK	Out	PLL settings or reference clock rate and routing.
QPLLRESET/CPLLRESET	In	Ensure this corresponds to reset guidelines in the user guide.
QPLLPD/CPLLPD/TXPD[1:0] /RXPD[1:0]	In	Power downs can save power but prevent circuits from operating.
Transmitter		
TXBUFSTATUS[1:0]	Out	Gives information on buffer over/under flow events.
TXPOLARITY	In	Can be used in case TX P/N have been swapped on the board.
Receiver		
RXCDRLOCK	Out	Gives an indication that data is being received at the right rate.
RXPOLARITY	In	Can be used in case RX P/N have been swapped on the board.
RXBYTEISALIGNED	Out	Shows if byte alignment occurred and has stayed consistent.
RXBYTEREALIGN	Out	Byte realignment should happen once during initialization. Multiple alignments can indicate garbage data is being received or that the CDR is losing alignment.
RXCOMMADET	Out	Comma detect should occur only at times delineated by the protocol. Other times indicates bit errors or garbage data.
RXDISPERR[7:0]	Out	Signals when a disparity error has occurred.
RXNOTINTABLE[7:0]	Out	Signals when a Not in Table error has occurred.
RXBUFSTATUS[2:0]	Out	Gives information on buffer over/under flow events

Resets

Errantly issued resets can be unforeseen causes of any number of problems, and conversely, can also resolve those same problems. It is important to understand when each reset should be used. UG476, *7 Series FPGAs GTX/GTH Transceivers User Guide* provides outlines that specify when and how each should be utilized, either on initialization or when data has been interrupted for whatever reason. Transceiver initialization is effectively a sequence of clocks becoming stable, followed by resetting their associated logic — starting with the internal PLL being used, followed by any external clock generation (such as an MMCM), then finishing with the phase alignment circuits (if they are used). The transmit and receive paths can be considered separately, though since they often share both the transceiver PLL resources and fabric clocking resources, it is important to know how both sides of the link are being reset.

The 7 series transceiver wizard is a valuable resource for generating reset logic. The code that the wizard generates contains the necessary reset state machines based on the options selected. In any case where initialization does not finish correctly, the wizard outputs can be used directly or as reference for the correct initialization sequence. Table 2 lists ports to observe via simulation or the ChipScope analyzer to ensure that resets are being issued correctly.

Table 2: **Ports to Observe via Simulation or ChipScope Analyzer**

PLL Resets	Transmitter Resets	Receiver Resets
CPLLRESET	GTTXRESET	GTRXRESET
QPLLRESET	TXPMARESET	RXPMARESET
	TXPCSRESET	RXDFELPMRESET
	TXUSERRDY	RXPCSRESET
	TXRESETDONE	RXBUFRESET
		RXUSERRDY
		RXRESETDONE

These signals can be readily observed in digital simulations free of any problems that can be present in hardware, and are a good first step for any debug effort. To assist in generating a simulation of any transceiver design, many of the Xilinx IP cores, including the 7 series Transceiver Wizard, include example simulation scripts. These scripts can provide a starting point to implement any design specific simulation.

ChipScope Tool for Serial Debug

When a digital simulation does not reproduce an error, more highly granular analysis tools need to be applied. Xilinx provides a unique tool that provides a level of visibility similar to that of digital simulation: ChipScope analyzer. By using a ChipScope ILA core, all of the same signals in Table 1 can be observed in hardware. With the capability of triggering on specific events, like an error, it becomes a simple matter to see the conditions that surround that event.

The 7 series Transceiver Wizard provides an example design that implements both a ChipScope ILA core (for observing operation) as well as a Virtual I/O (VIO) core that allows the user to interact with the design by issuing resets or changing the loopback status. This design can be implemented directly and used as a simple test case to prove out a board, a starting point for recreating an error, or as an example for how to implement the ChipScope analyzer connectivity.

The ChipScope analyzer also provides a tool specific to Xilinx transceivers: the IBERT core. IBERT is a tool used to validate a board design and determine the ideal board settings. At its base, IBERT allows a serial designer to modify, on the fly, the data pattern being generated and checked against, as well as all of the Tx driver and Rx equalization settings. At a more advanced level, IBERT can allow a designer to sweep ranges of emphasis and equalization settings and take bit error measurements for each point, allowing the designer to determine the most optimal settings for that particular system. By using the IBERT core in the CORE Generator™ system, a designer can enable the transceivers in question and generate a bit file that is specific to the board under test.

Using IBERT as a first-pass board validation tool can be done utilizing the data pattern generators and checkers and the multitude of loopback options available. Many third-party transceivers have test modes that generate and check a variety of PRBS or clocking patterns, which can be used with IBERT to determine if a link can be established. If a link fails to come up, there are a few things that can be checked directly in the GUI:

1. *Do the Near-End loopback modes link up correctly?* If there is a problem with a loopback mode, then any issues are contained to the local board.
2. *Did the PLL used for the link lock?* If it did not, then there might be some clock connection or rate inconsistencies.
3. *Does inverting the Tx or Rx datapath correct the problem?* This indicates that the P and N traces have been swapped on the board and can be corrected easily via the RX/TXPOLARITY port.
4. *Does a higher transition density pattern work?* If moving to a lower-order PRBS pattern or clocking pattern allows a link to be established, it might indicate that link tuning is necessary to ensure a robust link

Serial Channel Tuning

Hand tuning a serial channel via pre-emphasis, post-emphasis, and receiver equalization has historically been a difficult and time-consuming process requiring a thorough understanding of the channel and the effects that tuning has on the data. IBERT, however, has a number of tools to simplify the process. IBERT gives users the ability to sweep the tuning parameters and generate statistics to measure the quality of the link for each combination of those parameters.

When hand tuning a link, it is possible to tune to a local minimum or corner case that yields an acceptable Bit Error Ratio in the lab—but causes errors in the field. In the Rx Margin Analysis tab, IBERT gives the user the ability to sweep the analog functions to automate tuning of the link. By using the sweep functionality in IBERT, designers can find ranges of settings that yield error-free performance, avoid local minimums, and select their production settings based on mid-points in acceptable ranges — all to help ensure a robust link when the production system reaches the field.

In-System Eye Scan

Beyond sweeping parameters and observing bit rates, it is interesting to know the amount of margin available with any set of settings. Every transceiver in the 7 series FPGAs has circuitry that implements hardware eye scan, allowing a design to determine the size and shape of the eye in real time. IBERT gives direct access to that circuitry and can show the calculated eye, giving the user direct feedback on how tuning has affected the received signal. This functionality is available through the Rx Margin Analysis tab with the Tx Sweep functionality. Bit errors due to signal integrity problems can be the result of many different board-related issues, though the most common are related either to discontinuities or lossiness in the serial traces themselves, or in the transmitter's and receiver's ability to compensate for them. A variety of patterns can add or remove link stress to help exercise these elements and assist in link tuning.

A Pseudo-Random Bit Sequence (PRBS) is typically generated by an LFSR using industry-defined polynomials. These include PRBS-7, PRBS-9, PRBS-11, PRBS-23, PRBS-31, and others. As the polynomials increase in order, the transition density decreases, increasing the stress on the link. PRBS-7 and PRBS-31 are most commonly used, since their transition density resembles 8B/10B and 64B/66B encoding respectively. The Xilinx 7 series transceivers all contain PRBS generators and checkers for assisting in testing serial links. IBERT can dynamically select between any of these patterns on the fly for either the transmitter or receiver.

Board-Level Analysis

When the functionality provided is not sufficient to diagnose the source of an error, the digital designer needs to consider board-level issues. The three aspects of board-level design that require the most attention are:

- The power supplies
- The reference clocks
- The serial traces themselves

Xilinx provides a set of board-level checklists to ensure that critical board-level requirements are met. These can be found in UG476, *7 Series FPGAs GTX/GTH Transceivers User Guide*.

Taking analog measurements of the output serial stream can provide a lot of information on the quality of system fidelity. However, a high-speed oscilloscope with CDR, jitter decomposition, and eye-capture software is required to make meaningful eye diagrams or jitter measurements. Using "persist" mode while simply triggering on the data, for example, can make the captured data look much better than it actually is. If random or total jitter are higher than expected, there could be a power supply or reference clock problem. Similarly, making frequency domain measurements can be helpful in identifying strong aggressors if a particular peak matches another frequency in the system.

Reference Clocks

Reference clocks are the foundation of a transceiver system. As such, strong jitter performance on this clock is necessary to keep excess jitter out of the system. While most digital designers are familiar with cycle-to-cycle or peak-to-peak measurements of jitter, it is the frequency-domain measurement, or phase noise measurement, that is most important when considering a reference clock. Consequently, Xilinx pioneered the use of a phase noise mask for specifying an adequate reference clock in the Virtex®-6 FPGA, and continues to use this approach in the 7 series families.

Historically, reference clock selection for a serial transceiver provided a small number of expensive oscillators to choose from. By using a phase noise mask, as long as the reference clock selected does not exceed the mask, minimal additional jitter is passed onto the transmitted data, allowing designers to choose the least expensive clock that meets the phase noise mask. In-system phase noise measurements can be made by probing the clock as close to the FPGA as possible, usually a via on the back side of the board, with a scope that supports a frequency analysis of the measurements.

Power Supplies

Power supplies also need to be tightly controlled to avoid any aggression onto the transmitter jitter generation and the receiver jitter tolerance. Strong single-frequency aggressors on a power supply are especially worrisome in transceiver applications because they can inject similar frequencies of noise into the rest of the system, adding stress and reducing margin. Taking frequency domain measurements of each of the transceiver power supplies can highlight where any sources of noise originate. Noise can be the result of aggressors on the board, poorly isolated switching regulators, or even sharing the supply with digital logic (which is why this practice is typically discouraged).

Analog Simulation

Bit errors resulting from inconsistencies in boards, connectors, or other pieces of the serial interconnect are best identified through a thorough analog simulation, ideally before the boards are even built. The best way to vet a transceiver-based system prior to generating hardware is through analog simulations. Xilinx was one of the first companies to provide IBIS-AMI models for transceiver simulation (in Virtex-5 FPGAs). Leveraging the experience from past product generations, Xilinx is able to provide the most accurate IBIS-AMI models for the 7 series transceivers.

Analog simulation is an extensive subject and is described in the following white papers from Xilinx:

- WP419, *Equalization for High-Speed Serial Interfaces in Xilinx 7 Series FPGA Transceivers*
- WP382, *SerDes Channel Simulation in FPGAs Using IBIS-AMI*

Conclusion

Line rates are increasing at an exponential pace, and with them, the chances for bit errors to occur within a system are increasing. Xilinx provides users with state-of-the-art tools to evaluate the source of bit errors in logic with the ChipScope ILA core and in the transceiver links themselves with the ChipScope IBERT core. Combined with board-level checklists to simplify the board design process and analog simulation tools to help ensure a successful first board spin, these tools expedite the bring-up process of any serial system, from 1 Gb/s to 28 Gb/s.

For more information, go to www.xilinx.com/transceivers.

Revision History

The following table shows the revision history for this document:

Date	Version	Description of Revisions
10/08/12	1.0	Initial Xilinx release.

Notice of Disclaimer

The information disclosed to you hereunder (the "Materials") is provided solely for the selection and use of Xilinx products. To the maximum extent permitted by applicable law: (1) Materials are made available "AS IS" and with all faults, Xilinx hereby DISCLAIMS ALL WARRANTIES AND CONDITIONS, EXPRESS, IMPLIED, OR STATUTORY, INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, OR FITNESS FOR ANY PARTICULAR PURPOSE; and (2) Xilinx shall not be liable (whether in contract or tort, including negligence, or under any other theory of liability) for any loss or damage of any kind or nature related to, arising under, or in connection with, the Materials (including your use of the Materials), including for any direct, indirect, special, incidental, or consequential loss or damage (including loss of data, profits, goodwill, or any type of loss or damage suffered as a result of any action brought by a third party) even if such damage or loss was reasonably foreseeable or Xilinx had been advised of the possibility of the same. Xilinx assumes no obligation to correct any errors contained in the Materials or to notify you of updates to the Materials or to product specifications. You may not reproduce, modify, distribute, or publicly display the Materials without prior written consent. Certain products are subject to the terms and conditions of the Limited Warranties which can be viewed at http://www.xilinx.com/warranty.htm; IP cores may be subject to warranty and support terms contained in a license issued to you by Xilinx. Xilinx products are not designed or intended to be fail-safe or for use in any application requiring fail-safe performance; you assume sole risk and liability for use of Xilinx products in Critical Applications: http://www.xilinx.com/warranty.htm#critapps.