XAPP661 (v1.1) February 5, 2003

RocketIO Transceiver Bit-Error Rate Tester

Author: Dai Huang and Michael Matera

Summary

This application note describes the implementation of a RocketIO transceiver bit-error rate tester (BERT) reference design demonstrating a serial link (1.0 Gb/s to 3.125 Gb/s) between two RocketIO multi-gigabit transceivers (MGT) embedded in a single Virtex™-II Pro FPGA. To build a system, an IBM CoreConnect™ infrastructure connects the PowerPC™405 processor (PPC405) to external memory and other peripherals using the processor local bus (PLB) and device control register (DCR) buses. The reference design uses a two-channel Xilinx bit-error rate tester (XBERT) module for generating and verifying high-speed serial data transmitted and received by the RocketIO transceivers. The data to be transmitted is constructed using pseudo-random bit sequence (PRBS) patterns. The receiver in XBERT module compares the incoming data with the expected data to analyze for errors. The XBERT supports several different types of user selectable PRBS patterns. Frame counters in the receiver are used to track the total number of data words (frames) received, and total number of data words with bit errors. The processor reads the status and counter values from the XBERT through the PLB Interface, then sends out the information to the UART.

Hardware Implementation

Overview

Figure 1 provides a high-level view of the hardware contents of the RocketIO transceiver BERT reference design. This design demonstrates a system that uses PLB and DCR devices. The PLB consists of three slave devices. A block RAM (BRAM) controller connects 32KBytes BRAM to the bus serving as the data and instruction memory of the processor. Also attached to the bus are a UART module and a two-channel XBERT module each bonded with a Xilinx Intellectual Property Interface (IPIF) providing standardized connections to the bus. The processor has two PLB master connections, one for instruction cache and one for data cache. The only DCR device in the reference design is the bus error address and status register of the PLB Arbiter.

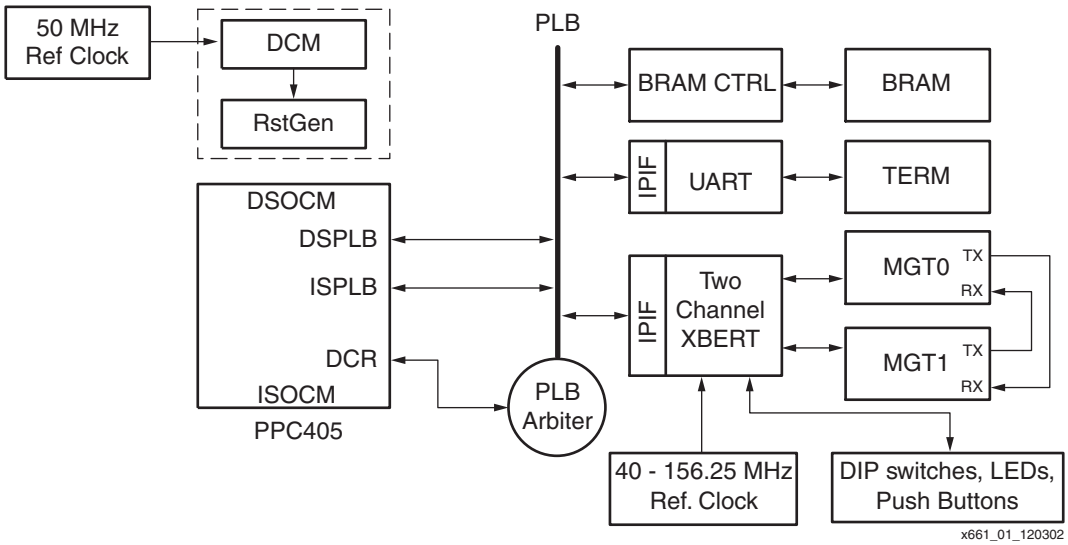

Figure 1: **High-Level Hardware View: RocketIO Transceiver BERT Reference Design**

Processor Local Bus (PLB)

The PLB connects the CPU to high-performance devices such as memory controllers. The PLB protocol supports higher-bandwidth transactions and has a feature set that better supports memory operations from the CPU. Highlights of the PLB protocol include synchronous architecture, independent read/write data paths, and split transaction address/data buses. The reference design includes a 64-bit PLB infrastructure with 64-bit master and slave devices attached.

The UART and the two-channel XBERT module do not require the high performance available over PLB, but are connected to PLB to demonstrate a simple system without OPB and keep the size of the design small.

The PLB devices in the reference design include:

- PLB Masters
  - CPU Instruction Side PLB Interface (Master ID = 0)
  - CPU Data Side PLB Interface (Master ID = 1)
- PLB Slaves
  - BRAM Controller (Slave ID = 0)
  - 16450 UART (Slave ID = 1)
  - Two-Channel XBERT Module (Slave ID = 2)
- PLB Arbiter
  - 8 Master, 64 bit Xilinx PLB Arbiter
- PLB Bus Logic
  - The specification for the PLB protocol requires creating a PLB slave bus. This is accomplished by using some additional logic to OR together the outputs of slave devices. This reference design supports up to nine PLB slave devices, but can easily be expanded to support additional slave devices.

Device Control Register (DCR)

A DCR is used for accessing control and status registers in various devices. The CPU contains a DCR master interface accessible through special move-to-DCR and move-from-DCR instructions. The only DCR device in this reference design is the bus-error address and status register of the PLB Arbiter.

Clock/Reset Distribution

Figure 2 illustrates the use of the digital clock manager (DCM) to generate clocks for the CPU and PLB bus in the design. A 50 MHz input reference clock is used to generate the main PLB clock. The CPU clock of 200 MHz is a multiple of the PLB clock. Since each clock is generated from the same 50 MHz reference clock input, they are phase-aligned with each other. This synchronous phase alignment is required by the CPU.

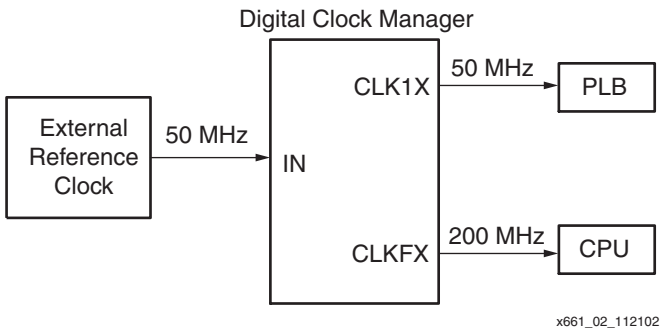

Figure 2: **CPU and PLB Clock Generation**

The startup reset is triggered by an external reset input and held by a block of internal logic until the DCM has locked onto its reference clock. Once the DCM is locked and the clocks remain stable for several cycles, the reset condition is released to allow the system logic to begin operating. As an example, the CPU will begin fetching instructions a few cycles after reset is released.

The two-channel XBERT module requires a separate high-quality clock source to drive RocketIO transceivers. The data transfer rate of the RocketIO transceiver will be 20 times this clock rate. Therefore, running the RocketIO transceiver at 3.125 Gb/s requires a 156.25 MHz clock source. This clock source must be at least 40 MHz, with a duty cycle between 45% and 55%, and a 100 ppm or better frequency stability, with as low as possible jitter. For lower jitter, it is necessary to use differential clock inputs. At speeds of 2.5 Gb/s or greater, the REFCLK configuration introduces more than the maximum allowable jitter to the RocketIO transceiver. For these high-speed designs, the BREFCLK configuration is required. The BREFCLK configuration uses dedicated routing resources to reduce jitter. BREFCLK must enter the FPGA through a dedicated clock I/O. There are two groups of dedicated differential clock input pads on the top of the device for the top MGTs, named BREFCLK and BREFCLK2 respectively. This is also true on the bottom. Setting the REF_CLK_V_SEL attribute of the RocketIO transceiver to "1" enables a BREFCLK path. The REFCLKSEL pin can be used to select between BREFCLK and BREFCLK2. Consult the **RocketIO Transceiver User Guide** for more information.

Figure 3 shows the clock generation for the two-channel XBERT module. Only one instantiation of the RocketIO transceiver is shown. The XBERT module uses a 2-byte data path. Therefore, it is capable of using a simple clock scheme with both USRCLK and USRCLK2 running at the same frequency. The buffered BREFCLK or BREFCLK2 is routed directly to the TXUSRCLK, TXUSRCLK2, RXUSRCLK, and RXUSRCLK2 on the RocketIO transceiver. All clock buffers on the BREFCLKs are instantiated inside the XBERT module. The board interface module in the two-channel XBERT requires a slow clock (typically 12.5 MHz) to drive the debounce circuitry and the LED pulse width modulation (PWM) module. This slow clock is derived from the PLB clock and generated from a separate DCM.

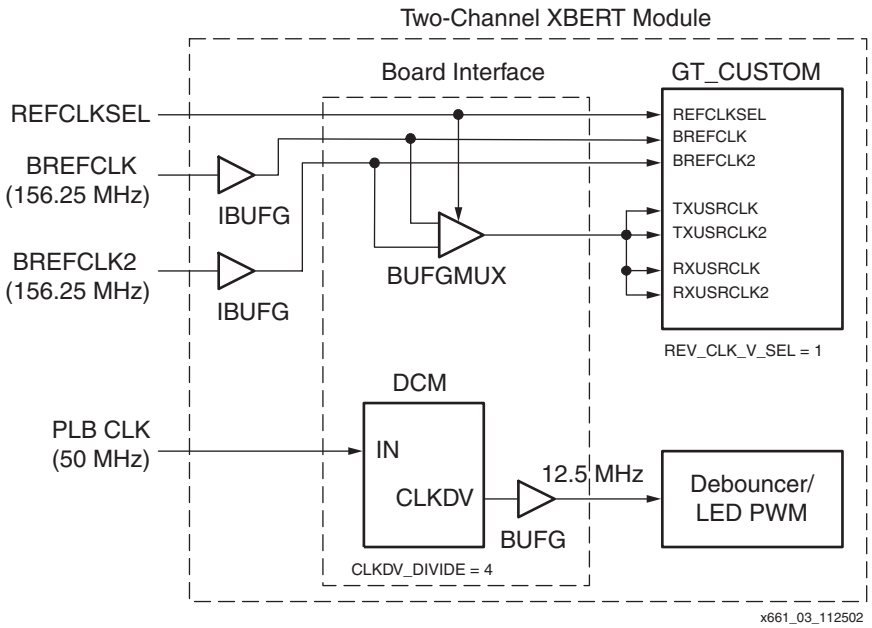

Figure 3: **Clock Generation in Two-Channel XBERT Module**

RocketIO Transceiver

The RocketIO transceiver consists of the physical-media attachment (PMA) and physical-coding sublayer (PCS). The PMA contains the serializer/deserializer (SERDES), TX and RX buffers, clock generator, and clock recovery circuitry. The PCS contains the 8B/10B encoder/decoder and the elastic buffer supporting channel bonding and clock correction. The PCS also handles cyclic redundancy check (CRC). The reference design only implement most necessary features provided by the RocketIO transceiver. Table 1 describes the RocketIO transceiver features deployed in the reference design.

Table 1: **Deployment of the RocketIO Transceiver Features in the Reference Design**

Layer	RocketIO Transceiver Features	Deployment in the Reference Design	Note
PMA	SERDES	Yes	
	SERDES_10B Mode	No	
	Comma Detect and Realign	Yes	PCOMMA only
	Clock Generation and Recovery	Yes	
	External Serial Loopback	Yes	False by default
PCS	Rx Elastic Buffer	Yes	
	TX FIFO	Yes	
	Internal Parallel Loopback	Yes	False by default
	Clock Correction	No	
	8B/10B Encoder/Decoder	No	
	CRC	No	
	Channel Bonding	No	

The reference design facilitates two programmable loopback features provided by RocketIO transceivers to allow the user quick testing of the system without having the need to apply external cables or pattern generators. Loopback allows the user to send the data that is being transmitted directly to the receiver of the transceiver. External serial loopback places the transceiver into a state where transmit data is directly fed back to the receiver in PMA layer. An important point to note is that the feedback path is at the output pads of the transmitter. This tests the entirety of the transmitter and receiver. Proper termination (typically 50 Ω) on output pads is required to run a RocketIO transceiver in serial loopback mode. The second loopback option is internal parallel loopback. It checks the digital circuitry only by feeding back the data before the SERDES. Termination on output pads is not required in parallel loopback although transmitter outputs still remain active.

The RocketIO transceiver BERT reference design implements 2-byte data path on the RocketIO transceiver parallel interface to the FPGA fabric. With 8B/10B encoding bypassed, the parallel interface on the RocketIO transceiver is 20 bits wide. The reference design supports using different level of TX_DIFF_CTRL and TX_PREEMPHASIS settings on the RocketIO transceiver to improve output waveform shaping for various load conditions.

Xilinx Bit-Error Rate Tester (XBERT)

The Xilinx Bit-Error Rate Tester (XBERT) is a set of HDL modules designed to test the bit-error rate of a RocketIO transceiver in Virtex-II Pro. A simple setup for doing such test is to apply SMA cables between two RocketIO transceivers so that transmitted data on each RocketIO transceiver can be looped back to the receiver of the other RocketIO transceiver over the established link. Note that such cable loopback is different from the two programmable loopback features provided by a RocketIO transceiver because it tests the entire hardware system. The XBERT is a self-contained module that can be implemented in FPGA fabric without CPU interference. The RocketIO transceiver BERT reference design gives an example of XBERT application in a PPC405 based system.

The bit-error rate measured by the XBERT is an estimate of the true bit-error rate. The XBERT is implemented in FPGA fabric communicating with the RocketIO transceiver through the parallel interface, so it cannot work on the serial stream directly. Using parallel data introduces the concept of frame. In the case of the XBERT, the frame is a 20-bit word. The XBERT checks incoming data frame-by-frame and counts for errors. Thus, one frame error implies multiple bit errors in a frame. However, in a very low error rate system, the chance of getting multiple random bit errors in a 20-bit frame is very low since errors are more likely to be scattered in multiple frames. Using good lab practices, the XBERT, and the formulas later in this document (see section **Derivation of the Bit-Error Rate**), user will be able to calculate with a high confidence the bit-error rate of their system.

A single channel XBERT contains three primary modules called GigabitBER_TX, GigabitBER_RX and MGT_BERT_4, as shown in Figure 4. GigabitBER_TX is to generate and transmit the data. GigabitBER_RX is used simply to check the received data and count errors over a serial link. MGT_BERT_4 provides a placeholder for the RocketIO transceiver instantiation.

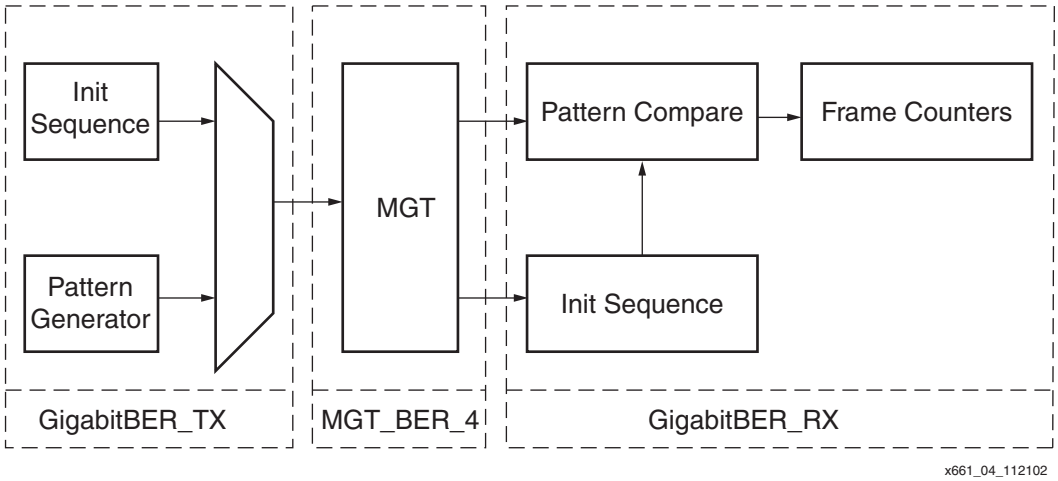

Figure 4: **Single Channel XBERT Block Diagram**

The reference design implements a two-channel XBERT driving two RocketIO transceivers, simply by replicating two XBERTs in the FPGA fabric. The two XBERTs share same clock inputs but take controls from own inputs and provide separate outputs such as the status and the frame counter values. A board interface module debounces inputs from the on-board DIP switches and push buttons, and synchronizes them to a slow clock (typically 12.5MHz) that is derived from the run-time clock input (i.e., the PLB clock in the reference design). This slow clock also drives a pulse width modulation (PWM) module that generates modulated LED outputs to the LEDs. Figure 5 illustrates a wrapper of all these modules instantiated in a two-channel XBERT module. Tables 3 to 7 provide port lists of the two-channel XBERT.

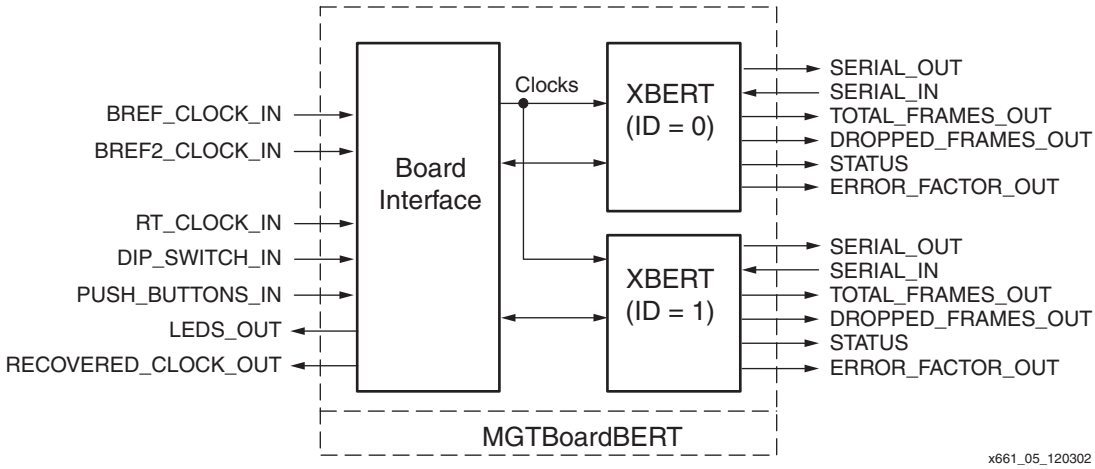

Figure 5: **Two-Channel XBERT Block Diagram**

In order to perform a successful bit-error rate test some key features of the RocketIO transceiver must be properly controlled. Most importantly the serial data stream must exactly match the data stream generated by the pseudo-random bit sequences (PRBS) pattern generator. In most cases this type of pattern violates run length and DC balance rules of the RocketIO transceiver. This violation is intentional in order to produce part stress. In order for the part to be stressed properly, 8B/10B encoding on the RocketIO transceiver must be bypassed. The use of the 8B/10B encoding will not only prevent proper line stress but will disrupt the operation of the XBERT. Other mandatory RocketIO transceiver attribute settings required by the XBERT are listed in Table 7.

Table 2: **List of Two-Channel XBERT Ports**

Port Name	Direction	Width	Clock Domain	Description
bref_clock_p_in bref_clock_n_in	In In	1 1	N/A	High quality RocketIO transceiver differential BREFCLK inputs.
bref2_clock_p_in bref2_clock_n_in	In In	1 1		High quality RocketIO transceiver differential BREFCLK2 inputs as an alternative to BREFCLK.
rt_clock_in	In	1		Board interface run-time clock input. It is used to generate the slow clock for the debouncer and LED PWM.
DIP_switch_in	In	16	Asynchronous	On-board DIP switch inputs. Refer to Table 3 for more information.
Push_buttons_in	In	4		On-board push button inputs. Refer to Table 4 for more information.
serial_p_in serial_n_in	In In	2 2	Analog	High speed differential inputs of two RocketIO transceiver ports. Bit 0 is the input to XBERT 0. Bit 1 is the input to XBERT 1.
serial_p_out serial_n_out	Out Out	2 2		High speed differential outputs of two RocketIO transceiver ports. Bit 0 is the output from XBERT 0. Bit 1 is the output from XBERT 1.
total_frames_out	Out	80	Buffered RocketIO Transceiver Reference Clock	Number of total frames received since link was established. Lower 40 bits are outputs from XBERT 0. Upper 40 bits are outputs from XBERT 1.
dropped_frames_out	Out	64		Number of frames received with error since link was established. Lower 32 bits are outputs from XBERT 0. Upper 32 bits are outputs from XBERT 1.
error_factor_out	Out	80		Shortest gap between two consecutive errors expressed in frames. This number is used to calculate the precision and confidence numbers of the bit-error rate.
Status	Out	16		XBERT status outputs. Lower 8 bits are outputs from XBERT 0. Upper 8 bits are outputs from XBERT 1. Refer to Table 5 for more information.
LEDs_out	Out	16	Internal slow clock running at a quarter frequency of the rt_clock_in	On-board LED outputs. Refer to Table 6 for more information.
recovered_clock_out	Out	3	N/A	Clock outputs. Bit 0 and 1 are the transceiver recovered clocks from the XBERT 0 and XBERT 1, respectively. Bit 2 is the buffered transceiver reference clock output.

Table 3: **DIP_switch_in Port on Two-Channel XBERT**

Bit	Function	Description	Default Value
0	XBERT0 Tx Inhibit	If a logic High, inhibits the RocketIO transceiver transmitter in XBERT 0	0
1	XBERT0 Power Down	If a logic High, shuts down both the receiver and the transmitter of the RocketIO transceiver in XBERT 0.	0
[3:2]	XBERT0 Loopback	Selects one of the two programmable loopback modes of RocketIO transceiver in XBERT 0. The higher bit is for serial loopback and the lower bit is for internal parallel loopback.	0
[6:4]	Pattern ID	Selects one of PRBS patterns to load in both XBERTs.	0
7	N/A		
8	XBERT1 Tx Inhibit	If a logic High, inhibits the RocketIO transceiver transmitter in XBERT 1	0
9	XBERT1 Power Down	If a logic High, shuts down both the receiver and the transmitter of the RocketIO transceiver in XBERT 1.	0
[11:10]	XBERT1 Loopback	Selects one of the two programmable loopback modes on the RocketIO transceiver in XBERT 1. The higher bit is for serial loopback and the lower bit is for internal parallel loopback.	0
[14:12]	N/A		
15	Clock Select	Selects BREFCLK or BREFCLK2 as the RocketIO transceiver reference clock. If a logic Low, uses BREFCLK. If a logic High, uses BREFCLK2.	0

Table 4: **Push_buttons_in Port on Two-Channel XBERT**

Bit	Function	Description	Default Value
0	Master Reset	Reset on the board interface of two-channel XBERT. If asserted, the DCM for the run-time clock in the board interface will be reset.	0
1	XBERT0 Tx Reset	If asserted, the transmitter in XBERT0 will be reset followed by sending a preamble sequence to the receiver of the other end. The detection of this preamble sequence will cause the receiver on the other end to be reset as well.	0
2	XBERT1 Tx Reset	If asserted, the transmitter in XBERT1 will be reset followed by sending a preamble sequence to the receiver of the other end. The detection of this preamble sequence will cause the receiver on the other end to be reset as well.	0
3	Rx Reset	If asserted, all frame counters in both XBERTs will be reset.	0

Table 5: **Status Port on Two-Channel XBERT**

Bit	Function	Description
0 8	Data Detect	Asserted when comma data is detected in the data stream. It is normal to see commas during a PRBS sequence.
1 9	Tx Reset Detect	Asserted when a transmitter reset is in progress. It indicates the transmitter is sending the preamble sequence to reset the link partner.
2 10	Abort	Asserted when ongoing bit-error rate test has been aborted due to a high bit-error rate detected. A high bit-error rate is simply classified by detecting two consecutive errors in the data stream. The frame counters are still operating when Abort occurs. However the bit-error rate measurement based on these frame counters is not likely to be accurate because of a possible out-of-sync over the link under such high error rate.
3 11	Link	Asserted when link between the transmitter and the receiver is established.
[5:4] [13,12]	Loopback	Indicates if the RocketIO transceiver is running in a loopback mode. The higher bit is for serial loopback and the lower bit is for internal parallel loopback.
6 14	Power Down	Asserted when the RocketIO transceiver is shut down.
7 15	Tx Inhibit	Asserted when the RocketIO transceiver transmitter is inhibited.

Notes:
1. The lower 8 bits are status outputs from XBERT 0 and the higher 8 bits are status outputs from XBERT 1.

Table 6: **LEDs_out port on Two-Channel XBERT**

Bit	Function	Description
0 8	Link	"ON" – Link is up "OFF" – Link is down
1 9	Tx Reset Detect	"ON" – Indicates the XBERT transmitter is sending the preamble sequence "OFF" – XBERT transmitter is in normal operation
2 10	Data Detect	"Blink" – Indicates a comma data is detected in the data stream.
3 11	Error Detect	"Blink" – Indicates a bit error was detected in a frame.
4 12	Abort	"ON" – Indicates that bit-error rate test has been aborted due to a high bit-error rate. "OFF" – No consecutive error is detected. Frame counters are trustworthy.
5 13	Frame Dropped	"ON" – One or more frames has been dropped due to bit errors. "OFF" – No dropped frame so far.
6 14	Tx Inhibit	"ON" – RocketIO transceiver TxInhibit input is a logic High. "OFF" – RocketIO transceiver TxInhibit input is a logic Low.
7 15	Power Down	"ON" – RocketIO transceiver PowerDown input is a logic High. "OFF" – RocketIO transceiver PowerDown input is a logic Low.

Notes:
1. The lower 8 bits are status outputs from XBERT 0 and the higher 8 bits are status outputs from XBERT 1.

Table 7: **XBERT Settings on RocketIO Transceiver Attributes**

Attribute	Value	Description
ALIGN_COMMA_MSB	True	Controls the alignment of detected commas within the transceivers 2-byte wide data path. XBERT assumes aligning comma with 20-bit alignment range.
RX_DECODE_USE	False	This determines if the 8B/10B decoding is bypassed. XBERT bypasses the 8B/10B.
RX_BUFFER_USE	True	Use the Rx buffer.
TX_BUFFER_USE	True	Use the Tx buffer.
RX_DATA_WIDTH TX_DATA_WIDTH	2	Use the 2-byte data path. XBERT is configured only to work in two- byte mode.
CLK_CORRECT_USE	False	Do not use clock correction logic.
CHAN_BOND_MODE	OFF	Do not use channel bonding. XBERT does not support channel bonding.
TX_CRC_USE RX_CRC_USE	False	XBERT do not use CRCs because these options will insert spurious data into the PRBS causing errors.
PCOMMA_DETECT	True	Enable detection of the PCOMMA (plus-comma). This comma is contained in the preamble sequence in link initialization. Without this option the XBERT will not link.
MCOMMA_DETECT	False	Do not detect the MCOMMA (minus-comma).
SERDES_10B	False	Denotes whether the reference clock runs at 1/20 (full rate) or 1/10 (half rate) the serial bit rate. XBERT does not support the half rate mode.
RX_CLK_V_SEL	1	Select BREFCLKs for 2.5 Gb/s or greater serial speeds. XBERT assumes RocketIO transceiver running at 2.5 Gb/s or higher.

Notes:
1. Attributes not listed in this table are set to the default values defined in the GT_CUSTOM cell model.

PRBS Pattern Generation

Bit-error measurements are important means to assess the performance of digital transmission. It is necessary to specify reproducible test sequences that simulate real traffic as closely as possible. Reproducible test sequences are also a prerequisite to perform end-to-end measurement. Pseudo-random bit sequences (PRBS) with a length of $2^n - 1$ bits are the most common solution to this problem.

PRBS pattern is produced using a linear-feedback shift register (LFSR) with appropriate feedback. If the LFSR has n stages, the maximum sequence length will be $2^n - 1$ bits. If the digital signal is directly taken from the output of the LFSR (non-inverted signal) the longest string of consecutive ZEROs will be equal n – 1. If the signal is inverted, n consecutive ZEROs and n-1 consecutive ONEs will be produced. In addition to strings of consecutive ZEROs and ONEs, PRBS pattern will contain any possible combination of ZEROs and ONEs within a string length depending on n.

There are two types of LFSR implementation for a certain polynomial that yields the equivalent result. *Fibonacci* LFSR or Type I LFSR uses exclusive OR (XOR) gates outside the shift register loop. *Galois* LFSR, or Type II LFSR uses exclusive OR gates inside the shift register chain. An implementation of multiple stage LFSR to produce PRBS pattern can be interpreted by a polynomial in a math perspective. For example, a polynomial $x^{15} + x^{14} + 1$ can represent *Fibonacci* LFSR implementation of a fifteen-stage shift register whose 14th and 15th stage outputs are added in a modulo-two addition stage, and the result is fed back to the input of the first stage. A polynomial $1 + x^{14} + x^{15}$ can represent equivalent *Galois* LFSR implementation

from the previous example. For LFSRs with only a few taps, the *Fibonacci* implementation will generally achieve a faster clock speed than its *Galois* counterpart. Although faster for a small number of taps, the *Fibonacci* implementation's performance degrades as the number of taps increases. The *Galois* implementation, however, sees hardly any performance loss with an increase in the number of taps. PRBS pattern generator designed in XBERT deploys *Galois* LFSR implementation.

The PRBS pattern generator designed in XBERT implements eight different polynomials specified in ITU-T Recommendation O.150. The ITU (International Telecommunication Union) is an international organization within the United Nations System where governments and the private sector coordinate global telecom networks and services. The ITU-T Recommendation O.150 contains general requirements applicable to instrumentation for performance measurements on digital transmission equipment. When setting the 3-bit Pattern ID on DIP_switch_in port on the two-channel XBERT module, the PRBS pattern generator will activate the corresponding LFSR. Such LFSR will generate PRBS pattern for the specific polynomial. By alternating PRBS patterns XBERT can produce different level of line stress on the RocketIO transceivers. Table 8 describes eight PRBS patterns and their respective polynomials implemented in XBERT. Among eight PRBS patterns there are four of them using inverted patterns. ITU-T considers the inverted patterns as being more stressful than non-inverted patterns when testing the clock recovery circuit in the network terminating devices.

Table 8: **PRBS Patterns and Polynomials**

ID [2:0]	Polynomial	Length of Sequence (bits)	Consecutive Zeros	Citation and Notes
0	$1+x^5+x^9$ (non-inverted signal)	2^9-1	8	ITU-T Recommendation O.150 section 5.1
1	$1+x^9+x^{11}$ (non-inverted signal)	2^11-1	10	ITU-T Recommendation O.150 section 5.2
2	$1+x^{14}+x^{15}$ (inverted signal)	2^15-1	15	ITU-T Recommendation O.150 section 5.3. It is one of the recommended test patterns in SONET specification
3	$1+x^3+x^{20}$ (non-inverted signal)	2^20-1	19	ITU-T Recommendation O.150 section 5.4. It is one of the recommended test patterns in SONET specification
4	$1+x^{17}+x^{20}$ (non-inverted)	2^20-1	19	Simplified ITU-T Recommendation O.150 section 5.5. ITU-T recommends forcing the output bit of this polynomial to be a ONE whenever the next 14 bits are all ZEROs so that the number of consecutive ZEROs will decrease to 14. This is not implemented in XBERT due to its additional complexity.
5	$1+x^{18}+x^{23}$ (inverted)	2^23-1	23	ITU-T Recommendation O.150 section 5.6. It is one of recommended test patterns in SONET specification
6	$1+x^{27}+x^{29}$ (inverted)	2^29-1	29	ITU-T Recommendation O.150 section 5.7.
7	$1+x^{28}+x^{31}$ (inverted)	2^31-1	31	ITU-T Recommendation O.150 section 5.8. This is a recommended PRBS test pattern for 10 Gigabit Ethernet. See IEEE Draft P802.3ae/D5.0.

PLB XBERT

The PLB XBERT is designed by connecting a two-channel XBERT module to a Xilinx PLB IPIF slave SRAM module, as illustrated in Figure 6. The PLB IPIF provides standardized PLB bus connections on one side and SRAM-like connections on the other side. The PLB IPIF makes it very easy to attach the two-channel XBERT module to the PLB CoreConnect Bus and uses very little glue logic. For more information regarding the Xilinx PLB IPIF slave SRAM module, see Virtex-II Pro Platform FPGA Developer's Kit[5], Volume 7, Hardware IP Specifications. The PLB IPIF used in PLB XBERT provides a slave SRAM protocol as described in the IPIF specification. The version of the PLB IPIF used by the PLB XBERT only supports PLB non-burst memory transactions (PLB_size[0:3] = 0000, PLB_type[0:2] = 000) of one to four bytes. It ignores all other PLB transaction sizes and types.

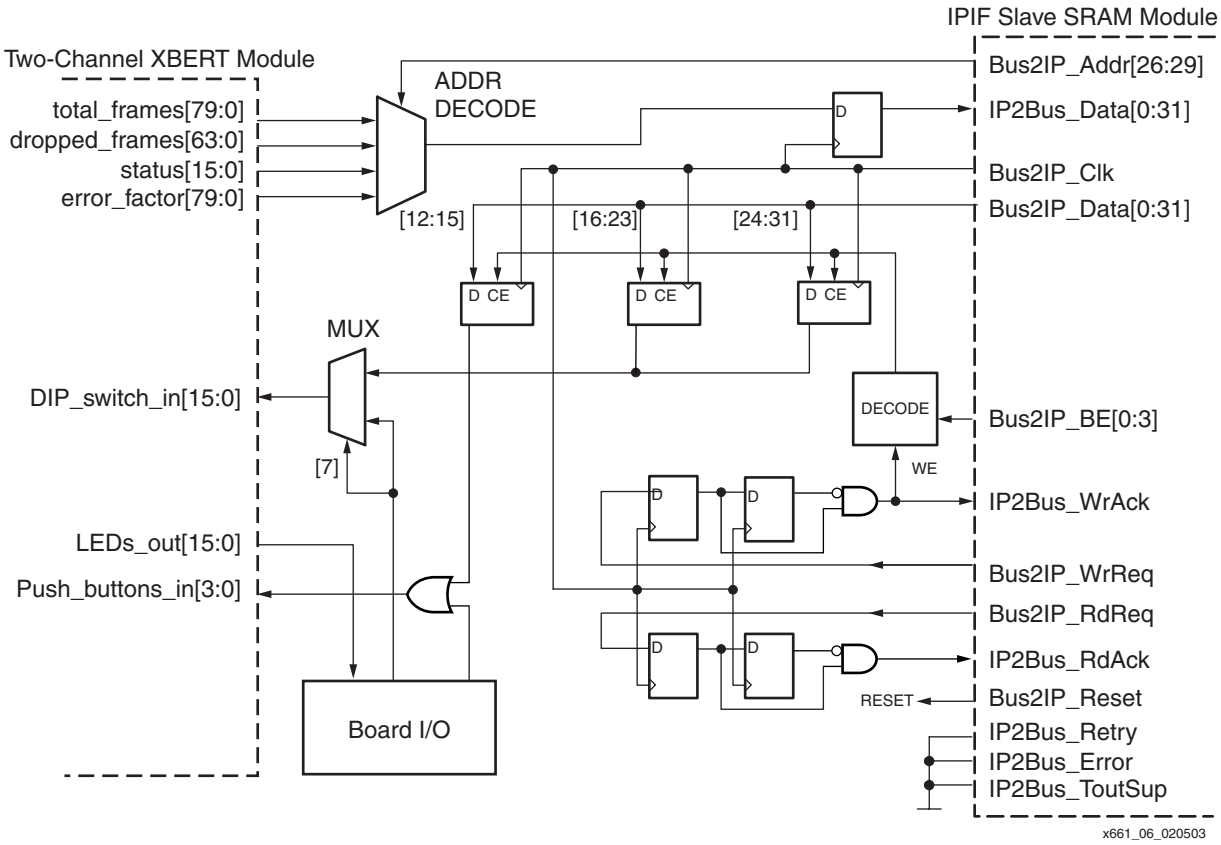

Figure 6: **Two-Channel XBERT PLB Interface**

Clock Domain

The interface is synchronized on Bus2IP_Clk, the PLB clock.

Read/Write Acknowledgement

The IPIF module only supports single read or write operation on the PLB. Proper read and write acknowledgement are required from the attached device. The two-channel XBERT PLB interface uses a simple technique to generate the acknowledgement responses. The acknowledgement of read and write cycles is handled by generating a single clock delayed envelope for the respective request signal. This envelope is also used to qualify the write transaction on the two-channel XBERT module. Any slave device attached on the PLB is assigned a base address so that read or write request can be distinguished from one slave device to another.

Read Operation

The IPIF module compares the read address with the base address of the two-channel XBERT module before it issues the read request to the XBERT module. The data outputs of the XBERT module consist one 96-bit total frame counter value, one 96-bit error factor value, one 64-bit dropped frame counter value, and one 16-bit status value. The Bus2IP_Addr bus is decoded to select multiple 32-bit vectors from the XBERT module outputs. Since the IPIF module uses 32-bit data path and it always provides byte addresses, the lower 2 address lines on Bus2IP_Addr are left unconnected, leaving the next four address lines (Bus2IP_Addr[26:29]) to provide up to sixteen 32-bit words. The output from the data multiplexer is registered at Bus2IP_Clk and then fed into IP2Bus_Data[0:31]. Even though the XBERT refreshes the frame counter outputs at the fast BREFCLK clock speed (e.g., 156.25 MHz), the CPU can only sample the data at PLB clock speed (i.e., 50 MHz).

Write Operation

The IPIF module compares the write address with the base address of the two-channel XBERT module before it issues the write request to the XBERT module. Since there is only a single 32-bit word to write to the XBERT, multiplexing on the Bus2IP_Addr in each write transaction is not necessary. However, the reference design requires the ability to write individual bytes of data to the XBERT module. The IPIF module provides the byte enables by the way of the Bus2IP_BE signals. To register individual byte from the 32-bits data bus, each bit of the Bus2IP_BE signals is ANDed with the write enable (WE) to create an enable signal on each register. The data inputs of the XBERT module consist of a 16-bit DIP switch input and a 4-bit push button input. The CPU can write data to these ports to override the inputs from the board I/O (i.e., DIP switches and push buttons) once the board I/O is disabled. Inputs from board I/O or the CPU are selected by bit 7 of external DIP switch inputs (DIPs [7]).

Other Devices on PLB

The PLB Arbiter, PLB UART and PLB BRAM controller used in RocketIO transceiver BERT reference design are standard components provided in the Virtex-II Pro Platform FPGA Developer's Kit[5]. The reference design does not have an interrupt controller, so the UART interrupt output is directly fed into the CPU's non-critical interrupt input. The critical interrupt input on the CPU is tied Low.

Software Implementation

Overview

The RocketIO transceiver BERT reference design provides a stand-alone software application called MGT_DEMO to work with the hardware platform. Figure 7 illustrate the flow diagram of the MGT_DEMO software application. The major function of this software is to read the two-channel XBERT statistics registers and print the register values (including the frame counter and status values) to the UART. Although the XBERT updates the frame counter and status outputs at the fast BREFCLK (typically 156.25 MHz) clock, XBERT statistics registers are only updated at the PLB clock (typically 50 MHz). Moreover the MGT_DEMO software only samples the statistics registers approximately every four seconds. In addition the software application reads the UART input then executes the corresponding command that is issued by the user on the PC Terminal.

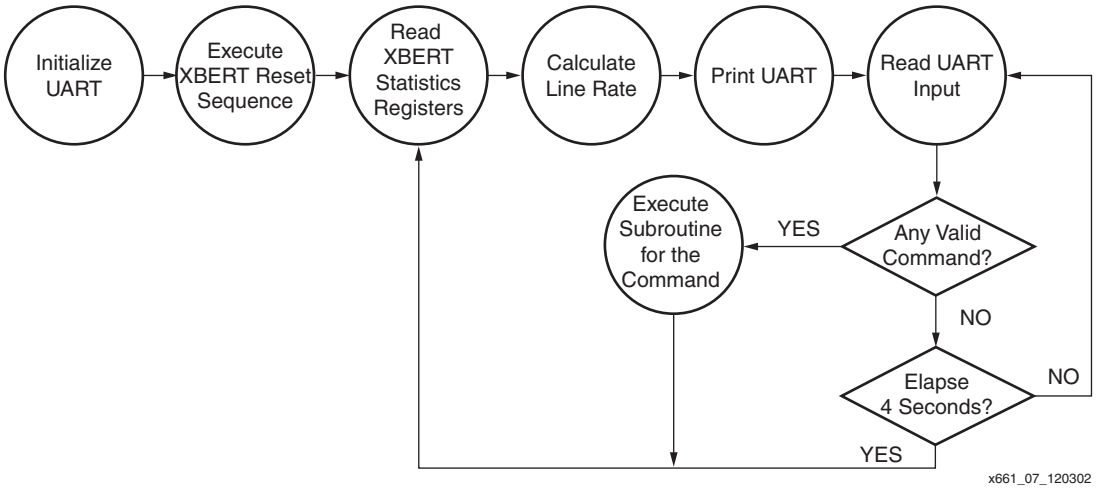

Figure 7: **Flow Diagram of MGT_DEMO Software Application**

The application is linked with the board support package (BSP) supplied in the Virtex-II Pro Platform FPGA Developer's Kit[5]. The BSP offers the interface to peripheral devices and to low-level PowerPC405 core functions. The BSP also provides functionality that allows the C library to access the hardware. For more information consult Virtex-II Pro Platform FPGA Developer's Kit[5], Volume 8-Software IP and Applications.

Software instruction is loaded into the BRAM on the PLB. UART and the two-channel XBERT are memory mapped to the PLB. Table 9 lists the software memory map and base addresses of these PLB devices.

Table 9: **PLB Device Memory Map**

Device	Address Boundaries		Size
	Upper	Lower (Base Address)	
PLB Slave 0 – BRAM	FFFFFFFF	FFFF8000	32 KBytes
PLB Slave 1 – UART	A000101F	A0001000	32 Bytes
PLB Slave 2 – Two-Channel XBERT	A001001F	A0010000	32 Bytes

UART Function

The application uses basic UART functions provided in the BSP library as well as some supplemental functions (e.g., uart_printf) provided in the MGT_DEMO application to communicate through the UART. The software initializes the UART with the baud rate, the number of data bits, the desired parity, and the number of stop bits. It then prints a menu showing the frame counter numbers and status of the two-channel XBERT to the UART. It also reads characters from the UART and calls the subroutine for the specific command. Table 10 lists all the supported characters and corresponding commands. In this implementation the software continuously polls on the UART for about 4 seconds between each print out of the menu. Typing ahead more than one character during the polling results in multiple executions of corresponding commands. This can alter result (e.g., switch the clock source twice).

Table 10: **Supported UART Input Character and Software Commands**

Character	Software Command	Equivalent Board I/O
"1"	Toggle RocketIO transceiver 0 Tx Inhibit. Type once to turn on Tx Inhibit. Type again to turn it off.	Bit 0 on DIP Switch Input
"2"	Toggle RocketIO transceiver 0 Power Down. Type once to power down. Type again to power up.	Bit 1 on DIP Switch Input
"3"	Toggle RocketIO transceiver 0 serial loopback. Type once to turn on serial loopback. Type again to turn it off.	Bit 3 on DIP Switch Input
"4"	Toggle RocketIO transceiver 1 Tx Inhibit. Type once to turn on Tx Inhibit. Type again to turn it off.	Bit 8 on DIP Switch Input
"5"	Toggle RocketIO transceiver 1 Power Down. Type once to power down. Type again to power up.	Bit 9 on DIP Switch Input
"6"	Toggle RocketIO transceiver 1 serial loopback. Type once to turn on serial loopback. Type again to turn it off.	Bit 11 on DIP Switch Input
"9"	Switch clock source. Type once to use BREFCLK2 for the RocketIO transceivers. Type again to switch back to BREFCLK.	Bit 15 on DIP Switch Input
"0"	Reset Two-Channel XBERT. This command will execute XBERT Reset Sequence.	Combination of bit 1, 2, and 3 on Push Button Input
"+" or "="	Switch to the next PRBS pattern. If it reaches the last PRBS pattern, it will roll back to the first one.	Bit 6 down to 4 on DIP Switch Input
"–" or "_"	Switch to the previous PRBS pattern. It it reaches the first PRBS pattern, it will move to the last PRBS pattern.	
ESC (0x1b)	Terminate the software.	N/A

XBERT Reset Sequence

To establish a link between two XBERT modules the software executes a XBERT Reset Sequence that includes a Master Reset, multiple attempts of Tx Reset following by an Rx Reset at the end on both XBERT modules. Since XBERT requires long delays after any Master Reset or TX Reset it is easy to deploy such delays in software by simply using delay loops. Figure 8 illustrates the flow diagram of the XBERT Reset Sequence.

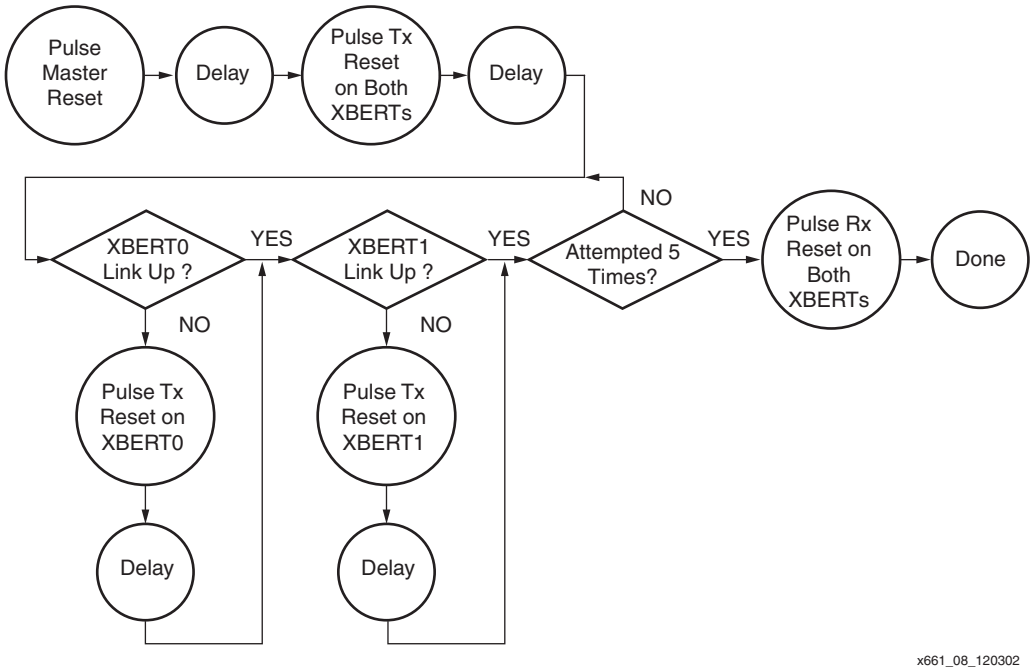

Figure 8: **Flow Diagram of XBERT Reset Sequence**

XBERT Statistics and Control Registers

A complete list of supported two-channel XBERT control and statistics registers is described in Table 11 and Table 12. The bit mapping of these registers are based on big Endian.

Table 11: **Two-Channel XBERT Statistics Register Address Map**

Address	Description	Attributes	Size
A0010000	Higher 32 bits of XBERT 0 total received frame count value	Read	32 bits
A0010004	Lower 32 bits of XBERT 0 total received frame count value	Read	32 bits
A0010008	XBERT 0 total dropped frame count value	Read	32 bits
A001000C	Higher 32 bits of XBERT 1 total received frame count value	Read	32 bits
A0010010	Lower 32 bits of XBERT 1 total received frame count value	Read	32 bits
A0010014	XBERT 1 total dropped frame count value	Read	32 bits
A0010018	XBERT status value (see note)	Read	32 bits
A001001C	Higher 32 bits of XBERT 0 error factor value	Read	32 bits
A0010020	Lower 32 bits of XBERT 0 error factor value	Read	32 bits
A0010024	Higher 32 bits of XBERT 1 error factor value	Read	32 bits
A0010028	Lower 32 bits of XBERT 1 error factor value	Read	32 bits

Notes:

1. In the 32-bit status value only the lower 16 bits are valid. See Table 5 for description of each valid bit.

Table 12: **Two-Channel XBERT Control Register Address Map**

Address	Description	Attributes	Size
A0010000	Register input to override push button inputs on board I/O. Bits 12 to 15 of this register are used to override bits 3 to 0 of the push button inputs, respectively. Other bits of this register are not used. See Table 4 for more information regarding push button inputs.	Write	16 bits
A0010002	Register input to override DIP switch inputs on board I/O . This input is only valid when bit 7 of the DIP switch inputs (DIPs [7]) is High. Bits 9 to 15 of this register are used to override bits 6 to 0 of the DIP switch inputs, respectively. Bits 0 to 7 of this register are used to override bit 15 to 8 of the DIP switch inputs, respectively. Bit 8 of this register is not used. DIP switch inputs on board I/O have one more valid bit (i.e., bit 7) than the DIP switch inputs on a two-channel XBERT. See Table 3 for more information regarding DIP switch inputs on a two-channel XBERT.	Write	16 bits

Calculation of RocketIO Transceiver Line Rate

The software calculates line rates on both RocketIO transceivers by the following equation. It first subtracts two sampled total received frame counter numbers (total_frame, total_frame_prv). Then it divides the result by the CPU cycles between two sampling point (delta_cycles), and multiply the result with parameters such as CPU_PERIOD, FRAME_BYTE_LENGTH to get the line rate in Megabit Per Seconds (Mb/s). FRAME_BYTE_LENGTH is a user-defined parameter. It gives the number of bytes in each frame. There are two in the reference design. CPU_PERIOD is another user-defined parameter that gives the CPU clock (typical 200MHz) in nanoseconds. Using functions (e.g., mltime_gettime) provided in the Virtex-II Pro Platform FPGA Developer's Kit[5], a user can acquire the CPU cycles for executing a group of instructions.

$$\text{LineRate(Mbps)} = \frac{(\text{total frames} - \text{total frames prv}) \times 8 \times \text{FRAME BYTE LENGTH} \times 1000}{\text{delta cycles} \times \text{CPU PERIOD}}$$

Calculation of RocketIO Transceiver Bit-Error Rate

RocketIO transceiver bit-error rate can be calculated through the frame counter numbers using the following simple equation. As described before the result given by such measurement is an estimate of the true bit-error rate in a low error rate system.

$$\text{BitErrorRate} = \frac{\text{dropped frames}}{\text{total frames}}$$

The following section describes how to calculate the precision and confidence of the bit-error rate test. It also illustrates the derivation of precision and confidence numbers from elementary principles of stochastic methods.

Derivation of the Bit-Error Rate

The bit-error rate measured by the XBERT is an estimate of the true bit-error rate. For completeness when reporting any statistical value you must also report the precision of your measurement and the probability your measurement is within that precision (the confidence). This section will explain how to calculate the precision and confidence of your bit-error rate test. Also this section will illustrate the derivation of these two numbers from elementary principles of stochastic methods. It assumes a basic working understanding of random variables.

The next section uses the following variables:

p = The probability of a bit error

n = total_frames_out $\times$ 20

e = dropped_frames_out

k = error_factor_out $\times$ 20

$\in$ = Interval of certainty (i.e., $\pm\in$)

c = The confidence

Some Useful Random Variables

The Bernoulli random variable describes the outcome of a Bernoulli trial. The most famous example of a Bernoulli trial is a coin flip. In the case of bit errors an outcome of 0 means the data is fine, and an outcome of "1" indicates an error. An error occurs with probability p. Thus, p is the theoretical bit-error rate.

Definition of the Bernoulli Random Variable

Domain

$$B_j \in \{0,1\} \quad (1.1)$$

Expected Value and Variance

$$E[B_j] = p \quad (1.2)$$

$$V[B_j] = p(1-p) \quad (1.3)$$

Probability Density Function

$$P[B_j = 1] = p \quad (1.4)$$

The Binomial random variable describes the outcome of two or more Bernoulli trials. The "0" or "1" outcomes are added together to produce a number X_n representing the number of occurrences of a bit error in n trials. The XBERT in effect does this by keeping track of the number of trials and the number of bit errors.

Definition of the Binomial Random Variable

Domain

$$X_n \in \{0, 1, 2, ..., n\} \quad (2.1)$$

Definition

$$X_n = \sum_{j=1}^{n} B_j \quad (2.2)$$

Expected Value and Variance

$$E[X_n] = nE[B_j] = np \quad (2.3)$$

$$V[X_n] = nV[B_j] = np(1-p) \quad (2.4)$$

Probability Density Function

$$P[X_n = x] = \binom{n}{x} p^x (1-p)^{(n-x)} \quad (2.5)$$

The geometric random variable is similar to the binomial random variable, except that instead of counting the number of bit errors the binomial random variable is counts the number of good bits between bit errors. The error_figure_out gives the minimum value of *G* measured in a given test period. As you will see later, this number is very useful in calculating a bit-error rate.

Definition of the Geometric Random Variable

Domain

$$G \in Z^{+} \tag{3.1}$$

Expected Value and Variance

$$E[G] = \frac{1}{p} \tag{3.2}$$

$$V[G] = \frac{1-p}{p^2} \tag{3.3}$$

Probability Density Function

$$P[G = x] = p\,(1-p)^{x-1} \tag{3.4}$$

Cumulative Distribution Function

$$P[G \le x] = 1 - (1-p)^{x} \tag{3.5}$$

Sampling

To measure the bit-error rate, samples of bit errors over a finite time interval are taken, then the results are used to make an assertion about the system at all times. This can be justified if the data gathered on the finite interval is representative of operation of the system and the interval is long enough. A special case of the Binomial random variable is used to understand how the interval effects the measurement.

Definition of the Sampled Binomial Random Variable

Domain

$$0 \le M_n \le n \tag{4.1}$$

Definition

$$M_n = \sum_{j=1}^{n} \frac{B_j}{n} \tag{4.2}$$

Expected Value

$$E[M_n] = E\left[\sum_{j=1}^{n} \frac{B_j}{n}\right] = \frac{1}{n}\sum_{j=1}^{n} E[B_j] = p \tag{4.3}$$

The expected value of M_n is *p*. The sampling process becomes an unbiased estimator of the bit-error rate.

Variance

$$V[M_n] = E\left[\sum_{j=1}^{n}\left(\frac{M_j - E[M_n]}{n}\right)^2\right] \tag{4.4}$$

$$V[M_n] = \frac{1}{n^2}\sum_{j=1}^{n} E\left[(M_j - E[M_n])^2\right] \tag{4.5}$$

Substituting for the variance of the Binomial random variable (2.4) produces:

$$V[M_n] = \frac{1}{n^2}\sum_{j=1}^{n} V[X_n] = \frac{p(1-p)}{n} \quad (4.6)$$

From Equation 4.6, the variance of the sample gets smaller with the number of samples. Intuitively, from equation (4.6) the larger the sample taken, the more representative the sample.

Making an Estimate of the Bit-Error Rate

To derive the precision and confidence of the experiment, an estimate of the upper bound of p must be made. This estimate, with the p_0 parameter is derived using the binomail random variable. The calculation requires specification a confidence value. That value, c_1, is used later to calculate the overall confidence value.

Calculation of p_0

Using the CDF of the binomial random variable (3.5), solving for p and substituting the variables yields:

$$p_0 = 1 - (1 - c_1)^{\frac{1}{k}} \quad (5.1)$$

From Equation (5.1), the value of p is less than or equal to p_0 with the probability of c_1. The higher the value of c_1 the looser the bound becomes, but the calculation becomes more credible.

Calculating the Precision

Chebychev's inequality is used in order to calculate the value of ε (the precision). Similar to the calculation of p_0, calculating ε requires a confidence value. This value is denoted c_2.

Derivation of ε

Chebychev's inequality:

$$P[\left|M_n - E[M_m]\right| \geq \varepsilon] \leq \frac{V[M_n]}{\varepsilon^2} \quad (6.1)$$

Substituting for the expected value and variance of the sampled binomial random variable (4.3) and (4.6) also using c_2 to represent the confidence:

$$c_2 \leq \frac{p(1-p)}{n\varepsilon^2} \quad (6.2)$$

Equation (6.2) requires knowing the theoretical value of p. Instead, take advantage of the inequality and substitute p with the previously calculated upper bound p_0. Simple algebra then yields:

$$\varepsilon \leq \sqrt{\frac{p_0(1-p_0)}{c_2 n}} \quad (6.3)$$

With these equations, the scientific quality results are calculated for the experiment.

The Bit-Error Rate With Precision and Confidence

Calculate the upper bound:

$$p_0 = 1 - (1 - c_1)^{\frac{1}{k}} \quad (5.1)$$

Then calculate the bit-error rate and the precision:

$$BER = \frac{e}{n} \pm \sqrt{\frac{p_0(1-p_0)}{c_2 n}}$$

Then determine the confidence:

$c = c_1 c_2$

Conclusion

This application note describes a RocketIO transceiver BERT reference design implemented in a Virtex-II Pro FPGA. This reference design demonstrates a 1.0 Gb/s to 3.125 Gb/s serial link between two RocketIO transceivers exchanging PRBS pattern. The reference design builds a simple PPC system that can be easily modified or extended. The hardware platform of the reference design is developed using HDL. The software application for the reference design is written in C language. The reference design can be downloaded from the Xilinx web site at:

ftp://ftp.xilinx.com/pub/applications/xapp/xapp661.zip

Reference

1. Xilinx, Inc., "RocketIO Transceiver User Guide".
2. Xilinx, Inc., "Linear Feedback Shift Register v2.0", October 2001.
3. ITU-T Recommendation O.150, "General Requirements for Instrumentation for Performance Measurements on Digital Transmission Equipment", May 1996.
4. IEEE Draft P802.3ae/D5.0, 10, "Media Access Control (MAC) Parameters, Physical Layer, and Management Parameters for 10 Gb/s Operation", May 2002.
5. Xilinx, Inc., "Virtex-II Pro Platform FPGA Developer's Kit", v1.0, March 2002.
6. Miron Abramovici, Melvin A. Breuer, Arthur D. Friedman, "Digital Systems Testing and Testable Design", 1990.

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
01/13/03	1.0	Initial Xilinx release.
02/05/03	1.1	Edits to Table 2 and Figure 6.