XAPP762 (v1.0) Sept. 30, 2004

RocketIO X Bit-Error Rate Tester Reference Design

Author: Dai Huang

Summary

This application note describes the implementation of a RocketIO X™ bit-error rate tester (XBERT) reference design. The reference design generates and verifies non-encoded high-speed serial data on one or multiple point-to-point links (2.5 Gb/s to 10 Gb/s) between RocketIO X multi-gigabit transceiver (MGT) ports, embedded within a single Virtex-II Pro™ X FPGA. This high-speed serial data is constructed in FPGA fabric using a pseudo-random bit sequence (PRBS) pattern, a clock pattern, or a user-defined pattern. The reference design provides access to the PMA attribute programming bus on the RocketIO X MGT, which enables real-time control of PMA features, such as the TX output swing, TX pre-emphasis, and RX equalization. The reference design utilizes the UltraController™ embedded processor—a lightweight PowerPC™ microcontroller solution. The embedded PPC405 processor transfers control and status between the XBERT module and the UART core through the General-Purpose Input/Output (GPIO) interface of the UltraController block, and enables a user interface through the software and an external RS-232 serial port. The reference design is built using the Embedded Development Kit (EDK), and it can be easily modified or extended.

Introduction

Figure 1 shows the high-level block diagram of the XBERT reference design. The data plane of the reference design consists of a configurable multi-channel XBERT module that generates and checks high-speed serial data transmitted and received by the MGTs. Each channel in the XBERT module contains two MGTs (MGT A and MGT B). These two MGTs connect to two dedicated pattern checkers, but they share the same pattern generator. The channels in the XBERT module operate independently. Each channel can load a different data pattern. The serial data rate of each MGT depends on the frequency of the reference clock and the PMA_SPEED mode applied. By incrementing the number of channels, the reference design can operate all eight MGTs in an XC2VPX20 and up to 10 MGTs simultaneously in an XC2VPX70 in Virtex-II Pro X FPGAs.

The control plane of the XBERT reference design utilizes an UltraController solution, which consists of the embedded PPC405 processor core, a 32-bit GPIO interface, a data-side block RAM controller (D-Side Controller), 16 KByte data-side block RAMs (D-Side BRAMs), an instruction-side block RAM controller (I-Side Controller), and 32 KByte instruction-side block RAMs (I-Side BRAMs). The control plane also includes an OPB UART Lite Xilinx IP core and a DCR-to-OPB bridge core (DCR2OPB Bridge). The 32-bit input/32-bit output GPIO interface connects the multi-channel XBERT module to the UltraController block, bridging between the data plane and the control plane. The processor reads the status and statistic values from the multi-channel XBERT module through the GPIO interface, then sends the information to the UART. The user can set up each XBERT channel, edit MGT PMA attribute settings, and control the bit-error rate (BER) test through the UART.

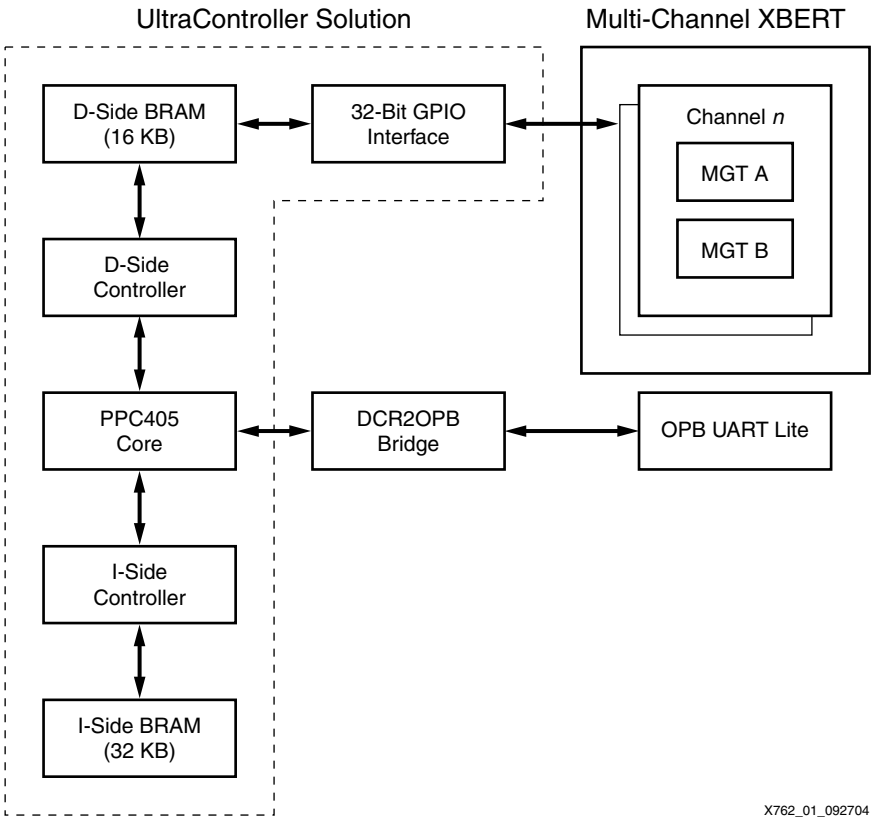


Figure 1: **Block Diagram of the XBERT Reference Design**

Features

The key features of the XBERT reference design are summarized below:

- Implements up to five channels that can enable up to 10 MGTs in Virtex-II Pro X FPGAs.
- Supports eight PMA_SPEED modes of the MGT that enable serial rates from 2.5 Gb/s to 10 Gb/s. Supports changing these PMA_SPEED modes on the fly.
- Supports seven ITU-T standard PRBS patterns (2^9-1, 2^{11}-1, 2^{15}-1, 2^{20}-1, 2^{23}-1, 2^{29}-1, 2^{31}-1). Supports a framed counter pattern to test MGT comma detection and alignment. Generates three clock patterns with frequency equal to 1/2, 1/10th, or 1/20th of the MGT serial rate.
- Supports BER test in both synchronous and asynchronous systems.
- Each channel can load an independent test pattern and use MGTs on the top or the bottom edge of the FPGA. The top MGTs can operate at a different serial rate and/or a different PMA_SPEED mode than the bottom MGTs.
- Supports all three MGT native loopback modes and a back-end echo mode to reflect incoming data back to the transmission in the FPGA fabric.
- Implements a self-synchronized pattern checker that automatically aligns and locks to an incoming PRBS pattern. Capable of linking with an external BER tester.
- Implements the UltraController block as a lightweight PPC405 processor solution.
- Embeds ChipScope™ Pro Integrated Logic Analyzer (ILA) core in each channel.
- Sends control and read status on a PC terminal using a serial port.
- Supports brute force scanning for optimal PMA settings of the MGT through the software.

Data Plane Description

MGT Use Model

To perform a successful bit-error rate (BER) test, some key features of the RocketIO X MGT must be properly controlled. The XBERT reference design incorporates a subset of features provided by the RocketIO X MGT, as listed in Table 1. All RocketIO X MGTs instantiated in the reference design use the recovered clock (RXRECCLK) to clock in the FPGA fabric on the receive side to avoid using clock correction schemes between local and remote ports. This makes the reference design capable of performing both asynchronous and synchronous BER tests between two RocketIO X MGTs or between an MGT to an external BER tester equipment. Most importantly, the reference design must produce standard PRBS patterns in the serial data stream. Such patterns are considered to be most stressful to the MGT on both run lengths and DC balance aspects. The 8B/10B or 64B/66B coding on a RocketIO X MGT must be bypassed to produce PRBS patterns. The use of 8B/10B or 64B/66B coding not only prevents proper line stress but disrupts the operation of the reference design.

Table 1: **Deployment of the RocketIO X MGT Features in the XBERT Reference Design**

Layer	RocketIO X MGT Features	Deployment in the Reference Design(1)	MGT Attribute or Port Settings(2)
PMA	SERDES	Yes	
	Clock and Data Recovery	Yes	
	Comma Detection and Alignment	Optional	PCOMMA_DETECT = TRUE MCOMMA_DETECT = TRUE PCOMMA_10B_VALUE = 0b0101111100 MCOMMA_10B_VALUE = 0b1010000011 COMMA_10B_MASK = 0b1111111111 ALIGN_COMMA_WORD = 4 or 2 RXCOMMADETUSE = 0b1 ENMCOMMAALIGN = 0b1 or 0b0 ENPCOMMAALIGN = 0b1 or 0b0
	Pre-driver and Post-driver Serial Loopback	Optional	LOOPBACK = 0b01 or 0b10
	PMA Attribute Programming Bus	Optional	
	Output Swing and Emphasis	Optional	
	Receive Equalization	Optional	
	TXOUTCLK	Yes	
	RXRECCLK	Yes	

Table 1: **Deployment of the RocketIO X MGT Features in the XBERT Reference Design *(Continued)***

Layer	RocketIO X MGT Features	Deployment in the Reference Design(1)	MGT Attribute or Port Settings(2)
PCS	RX Elastic Buffer	Yes	RX_BUFFER_USE = TRUE
	TX FIFO	Yes	TX_BUFFER_USE = TRUE
	8B/10B Encoder/Decoder	No	RXDEC8B10BUSE = 0b0 TXENC8B10BUSE = 0b0 TXBYPASS8B10B = 0b11111111
	64B/66B Encoder/Decoder, Scrambler, Gearbox, and Block Sync	No	TXENC64B66BUSE = 0b0 RXDEC64B66BUSE = 0b0 TXSCRAM64B66BUSE = 0b0 RXDESCRAM64B66BUSE = 0b0 TXGEARBOX64B66BUSE = 0b0 RXBLOCKSYNC64B66BUSE = 0b0
	Clock Correction	No	CLK_CORRECT_USE = FALSE
	Channel Bonding	No	CHAN_BOND_MODE = OFF
	Internal Parallel Loopback	Optional	LOOPBACK = 0b11

Notes:

1. Yes – This feature is always enabled in the reference design.
   Optional – This feature is enabled as an option in the reference design.
   No – This feature is disabled and is not supported in the reference design.
2. Only the most important MGT attributes and ports related to these features are listed in this table.

The XBERT reference design supports using either a differential BREF clock (BREFCLK) or a single-ended REFCLK input to drive the MGTs. While only one of these reference clocks is needed to drive the MGT, BREFCLK is recommended for the best operation. The BREFCLK configuration uses dedicated routing resources that reduce jitter. REFCLK usage results in performance degradation and is not recommended. Refer to the *RocketIO X Transceiver User Guide* (UG035) for details regarding the choice between BREFCLK and REFCLK.

Figure 2 illustrates the MGT clocking scheme in a typical four-channel XBERT reference design. Since the reference design always sets the external and internal data widths on the MGTs to be the same, it can use the 1:1 clock model on the MGTs. TXOUTCLK and RXRECCLK from the MGT are fed to global clock buffers (BUFGs) and are used to clock the transmitting and receiving side logic and the USRCLK/USRCLK2 ports on the MGT. To reduce the number of DCMs and BUFGs required in the system, the reference design does not use a DCM to generate USRCLK/USRCLK2 and shares one BUFG for the TXUSRCLK/TXUSRCLK2 signals among all MGTs on the same bank. Therefore, the channels used to generate the TXOUTCLK for the rest of MGTs (for example, channel 0 and channel 1 shown in Figure 2) must be always activated, and all MGTs on the same bank must operate at the same serial speed using the same PMA_SPEED mode.

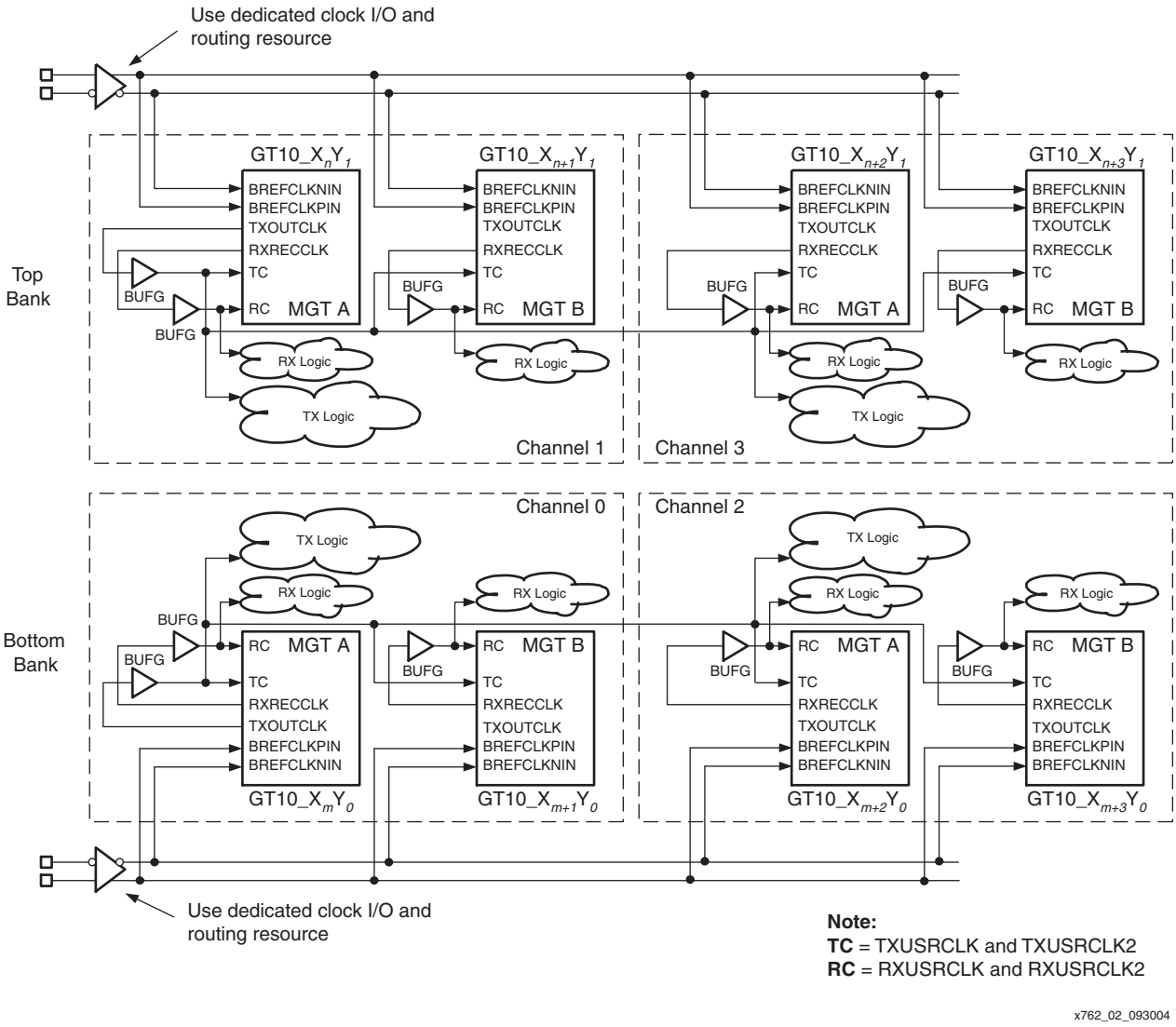


Figure 2: **MGT Clocking Scheme and Placement in the XBERT Reference Design**

The XBERT reference design does not use the reference clock (BREFCLK or REFCLK) to generate TXUSRCLK/TXUSRCLK2 because it requires a frequency divider when using PMA_SPEED modes 27_20, 24_20, 21_40, and 20_40, which adds further complexity in the reference design.

The XBERT reference design can enable multiple MGTs by incrementing the number of channels implemented in the design. Due to the limited availability of BUFGs in a Virtex-II Pro X FPGA, a maximum of five channels can be implemented using the reference design. Each channel contains two MGTs, which should be placed next to each other in the FPGA. Each channel is assigned a channel number. The reference design allows placement of MGTs in all channels at the same edge or opposite edges of the FPGA. Placing MGTs at opposite edges is the default configuration. In this case, even channels instantiate MGTs at the bottom edge of the FPGA, and odd channels instantiate MGTs at the top edge of the FPGA. Figure 2 illustrates a typical placement of MGTs relative to the channel configurations in the reference design, as seen from the viewpoint of the Xilinx FPGA editor or the Xilinx floor planner.

Table 2 lists the supported PMA_SPEED modes in the XBERT reference design. It is divided into two groups (20BIT and 40BIT) based on the data width configuration. Group 20BIT includes four PMA_SPEED modes that use 20-bit internal and external bus widths on the MGTs. This group typically is used when the serial speed is below 5 Gb/s. Group 40BIT

includes four PMA_SPEED modes that use 40-bit internal and external bus widths on the MGTs. This group typically is used when the serial speed is 5 Gb/s or greater.

Table 2: **Supported PMA_SPEED Modes in XBERT Reference Design**

Supported PMA_SPEED Mode in XBERT	XBERT Data Width Configuration	MGT External Bus Width (bits)	MGT Internal Bus Width (bits)	Frequency Ratio of USRCLK/ USRCLK2	Optimal Serial Speed (Gb/s)	REFCLK/ BREFCLK Frequency (MHz)	TXOUTCLK/ RXRECCLK/ USRCLK/ USRCLK2 Frequency (MHz)
28_20	20BIT	20	20	1:1	2.5	125	125
27_20					2.5	250	125
25_20					3.125	156.25	156.25
24_20					3.125	312.5	156.25
23_40	40BIT	40	40		3.1875	159.375	159.375
21_40					5.0	250	125
20_40					6.25	312.5	156.25
13_40					10.0	250	250

The XBERT reference design implements the following loopback functions on MGTs. The first three loopback modes are native features provided by the MGT. Refer to the *RocketIO X Transceiver User Guide* for details on these three loopback modes.

- Internal Parallel Loopback
- Post-Driver Serial Loopback

  This loopback mode requires termination of TXN and TXP for proper operation.
- Pre-Driver Serial Loopback

  This loopback mode does not toggle the TXN and TXP, and does not require TXN and TXP to be terminated.
- Back-end Echo

  The XBERT reference design implements this feature in the FPGA fabric to allow reflection of incoming data. The received data is buffered, synchronized on the local reference clock, and then sent back through the MGT transmitter. The pattern checker and the embedded ChipScope ILA core in the reference design still operate in this mode, allowing the user to check incoming data. Using the back-end echo mode requires the remote clock source and the local reference clock source to be frequency locked. The buffer used in this mode has limited depth and does not support clock compensation. A slight frequency difference between read and write ports on the buffer eventually results in buffer underflow or overflow, hence some data is dropped in the buffer and cannot be reflected properly.

PRBS Pattern Generation

Bit-error measurements are important means of assessing 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 lengths of $2^n - 1$ bits are the most common solution to this problem.

The PRBS pattern is produced using a linear-feedback shift register (LFSR) with appropriate feedback. If the LFSR has n stages, the maximum sequence length is $2^n - 1$ bits. If the digital

signal is taken directly from the output of the LFSR (non-inverted signal), the longest string of consecutive zeros is equal to n – 1. If the signal is inverted, n consecutive zeros and n – 1 consecutive ones are produced. In addition to strings of consecutive zeros and ones, the PRBS pattern contains any possible combination of zeros and ones within a string length depending on n.

There are two types of LFSR implementations 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 LFSRs to produce a PRBS pattern can be interpreted by a polynomial in a math perspective. For example, the polynomial $x^{15} + x^{14} + 1$ can represent a Fibonacci LFSR implementation of a 15-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. The polynomial $1 + x^{14} + x^{15}$ can represent the equivalent Galois LFSR implementation from the previous example. For LFSRs with only a few taps, the Fibonacci implementation generally achieves 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. The PRBS pattern generator designed in the XBERT reference design deploys a Fibonacci (Type-I) LFSR by default.

Table 3 summarizes all the patterns and their respective polynomials implemented in the XBERT reference design.

Table 3: **Supported Patterns in XBERT Reference Design**

Number	Pattern Name	Pattern or Polynomial	Length of Sequence (bits)	Consecutive Zeros	Notes
0	1/2X Clock	10101010 ……	2	0	This pattern can be used as a clock pattern whose frequency is equal to 1/2 of the MGT serial speed, generating up to a 5 GHz differential clock on the transceiver serial outputs. This pattern also can be used as a high-frequency test pattern as defined in IEEE Std 802.3-2002.
1	1/10X Clock	5 ones, 5 zeros	10	5	This pattern can be used as a clock pattern whose frequency is equal to 1/10th of the MGT serial speed, generating up to a 1 GHz differential clock on the transceiver serial outputs. This pattern can also be used as a low-frequency test pattern as defined in IEEE Std 802.3-2002.
2	1/20X Clock	10 ones, 10 zeros	20	10	This pattern can be used as a clock pattern whose frequency is equal to 1/20th of the MGT serial speed, generating up to a 500 MHz differential clock on the transceiver serial outputs.

Table 3: **Supported Patterns in XBERT Reference Design *(Continued)***

Number	Pattern Name	Pattern or Polynomial	Length of Sequence (bits)	Consecutive Zeros	Notes
3	2^7-1 PRBS	x^7+x^6+1 (non-inverted signal)	2^7-1	6	N/A
4	2^9-1 PRBS	x^9+x^5+1 (non-inverted signal)	2^9-1	8	ITU-T Recommendation O.150, Section 5.1.
5	2^{11}-1 PRBS	$x^{11}+x^9+1$ (non-inverted signal)	2^{11}-1	10	ITU-T Recommendation O.150, Section 5.2.
6	2^{15}-1 PRBS	$x^{15}+x^{14}+1$ (inverted signal)	2^{15}-1	15	ITU-T Recommendation O.150, Section 5.3. This is one of the recommended test patterns in the SONET specification.
7	2^{20}-1 PRBS	$x^{20}+x^3+1$ (non-inverted signal)	2^{20}-1	19	ITU-T Recommendation O.150, Section 5.4. This is one of the recommended test patterns in the SONET specification.
8	2^{23}-1 PRBS	$x^{23}+x^{18}+1$ (inverted signal)	2^{23}-1	23	ITU-T Recommendation O.150, Section 5.6. This is one of the recommended test patterns in the SONET specification.
9	2^{29}-1 PRBS	$x^{29}+x^{27}+1$ (inverted signal)	2^{29}-1	29	ITU-T Recommendation O.150, Section 5.7.
10	2^{31}-1 PRBS	$x^{31}+x^{28}+1$ (inverted signal)	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 Std 802.3ae-2002.
11	Reserved				
12	K28.5+ K28.5-	K28.5+, K28.5-, K28.5+, . . .	20	5	This pattern consists of two or four 10-bit values, which are K28.5+ and K28.5-.
13	User defined	User-defined pattern	1 to 20	0 to 19	This pattern consists of two or four 10-bit values, which can be picked from the 8B/10B table, or any value the user defines. By default, they are K28.5+ and K28.5-. Note: Do not use an all zero or an all one pattern.
14	Reserved				
15	Framed counter	Framed counter pattern	Approximately 320 (in 20-bit group) or 640 (in 40-bit group)	20 or 40	This pattern has a controllable frame length and an inter-frame gap (IFG).

Pattern Generator

The pattern generator in the XBERT reference design generates either 20-bit or 40-bit patterns to work with a 20-bit or a 40-bit MGT fabric interface. Figure 3 shows a block diagram of a pattern generator. The pattern generator contains 14 individual pattern generation blocks. The outputs of these blocks are multiplexed and registered, and they are provided as either 20-bit or 40-bit data outputs (data_out).

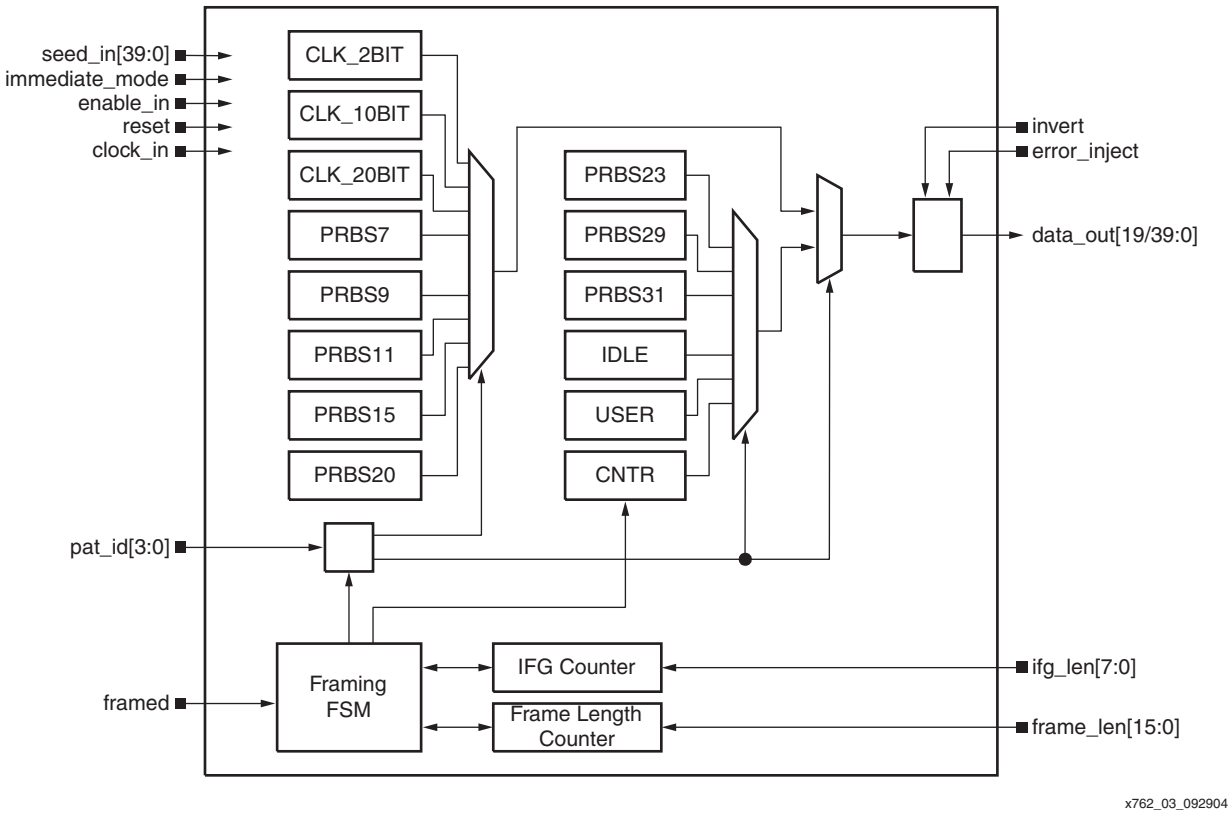


Figure 3: **Pattern Generator Block Diagram**

The pattern generator implements seven different polynomials as specified in ITU-T Recommendation O.150 for PRBS pattern generation. The International Telecommunication Union (ITU) is an international organization within the United Nations System, where governments and the private sector coordinate global telecom networks and services. ITU-T Recommendation O.150 contains general requirements applicable to instrumentation for performance measurements on digital transmission equipment. By alternating PRBS patterns, the reference design can produce different levels of line stress on the RocketIO X MGTs.

In addition to PRBS patterns, the pattern generator supports the following special patterns:

- The clock pattern (CLK_2BIT, CLK_10BIT, or CLK_20BIT) contains a constant 20-bit or 40-bit word composed with interleaving zeros and ones. The number of consecutive zeros or ones is 1, 5, or 10. Applying such a pattern to the MGT results in a clock output with frequency equal to 1/2, 1/10th, or 1/20th of the MGT serial rate. For example, if the MGT operates at 5 Gb/s, a CLK_20BIT pattern generates a 250 MHz differential clock on the MGT TXP and TXN outputs.
- The idle pattern (IDLE) consists of interleaving K28.5 + (0b0011111010) and K28.5 – (0b1100000101) symbols. Such a pattern contains only comma symbols as defined in the XBERT reference design.
- The counter pattern (CNTR) consists of 20-bit or 40-bit incremental counter values and idle words, providing a traceable and predicable test pattern different from the PRBS

pattern. Since the framed transmission of a counter pattern requires completion of MGT comma detection and alignment, the counter pattern can be used to test comma detection and alignment functions provided by the MGTs.

- The user pattern contains a constant 20-bit or 40-bit word that can be a combination of any 10-bit control (K) and/or data (D) symbols picked from the 8B/10B table.

The pattern generator contains a finite state machine (FSM) to delimit the counter pattern by periodically inserting idle words. A word is either a 20-bit or 40-bit vector determined by the data width of the MGT fabric interface. Idle words are used to fill in inter-frame gaps (IFGs) and consist of K28.5 +/- symbols only. The frame length and the IFG are configurable through the frame length input (frame_len) and the IFG length input (ifg_len).

The pattern generator works in either immediate mode (immediate_mode = 1) or self-advance mode (immediate_mode = 0). In immediate mode, the pattern generator takes the 40-bit seed value (seed_in) to calculate and output the pattern in three clock cycles. The output pattern is an immediate outcome of the seed value and the internal polynomial. In self-advance mode, the pattern generator calculates and updates the output pattern every clock cycle. The output pattern is an accumulated result of the initial seed value and the internal polynomial.

The pattern generator can inject errors into the output pattern through the error injection control port (error_inject). When this port is asserted High, all bits on the output are flipped to introduce a burst of bit errors into transmission. This function is implemented to self-check the integrity of pattern generator and checker in the reference design.

The pattern generator can invert the output pattern controlled by the invert port. PRBS 2^{15}-1, 2^{23}-1, 2^{29}-1, and 2^{31}-1 are specified as inverted patterns by default in ITU-T Recommendation O.150. 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. The pattern generator inverts these patterns by default, but allows the user to invert these patterns back in order to link up with a non-standard external BER tester.

The pattern generator contains heavy combinatorial logic (for example, a 40-bit carry chain) for parallel PRBS pattern generation that must meet the 250 MHz timing target in order to support a 10 Gb/s data rate.

Pattern Checker

The pattern checker in the XBERT reference design can verify either 20-bit or 40-bit incoming data from a 20-bit or 40-bit MGT fabric interface. Figure 4 shows a block diagram of a pattern checker. The pattern checker contains another instance of the pattern generator that is identical to the one on the transmission side. This pattern generator generates an expected pattern to compare with the incoming data. Depending on the comparison result, the link detection and bit-error measurement are conducted at every clock cycle.

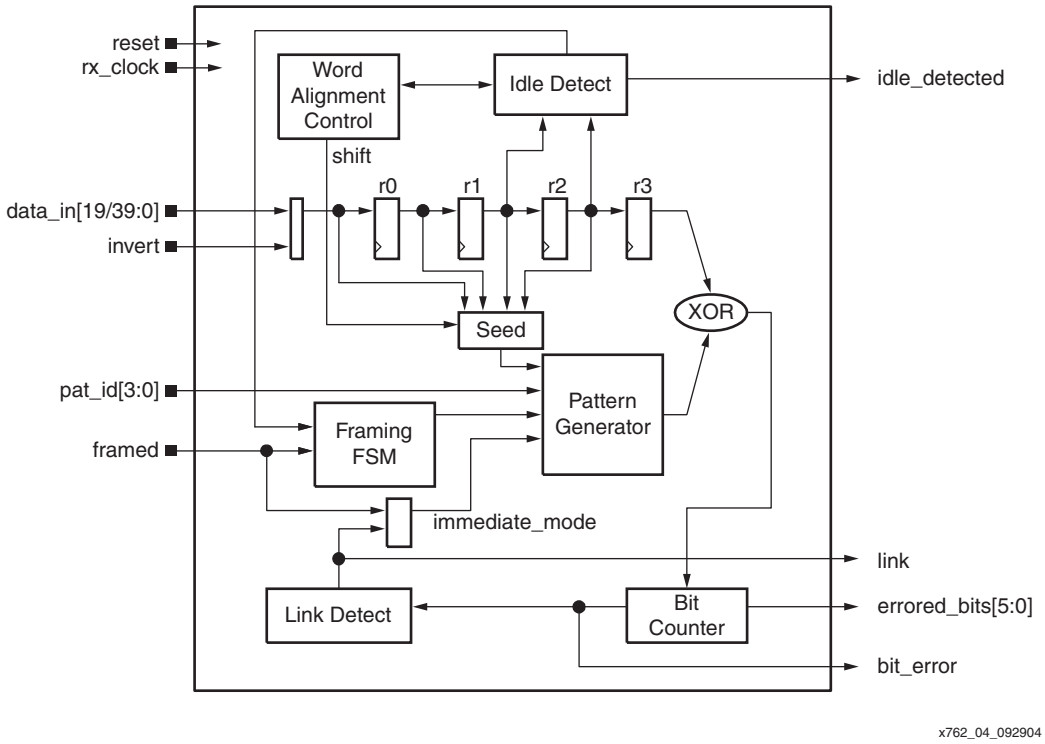


Figure 4: **Pattern Checker Block Diagram**

When the counter pattern is selected in the XBERT reference design, the pattern checker needs to recover the frame boundary by detecting the idle words within the IFG of the incoming data. The counter pattern is the only pattern that exercises the MGT comma detection and alignment function. The reference design defines a comma as a K28.5 symbol. Since the idle word is a 20-bit or 40-bit value, searching the idle word in the incoming data stream requires completion of word alignment. Word alignment, handled in the pattern checker, works in conjunction with the comma alignment inside the MGT. The pattern checker contains a finite state machine (FSM) that delimits the received counter pattern and skips checking the data within the IFG.

The pattern checker works in either immediate mode (immediate_mode = 1) or self-advance mode (immediate_mode = 0). The pattern checker always starts in the immediate mode when link is down or reset is asserted. As illustrated in Figure 5, in the immediate mode, the pattern checker uses the incoming data to generate the seed value for the embedded pattern generator. From immediate mode, this pattern generator takes the seed value to generate an output pattern in three clock cycles. This output pattern is expected to match the next incoming data. In real practice, several pipeline stages are introduced on the incoming data path to compensate for the three cycle delays in the pattern generator and meet an optimal timing target for 10 Gb/s operation. Therefore, the pattern checker can automatically adapt the internal state of the polynomial for the incoming data to generate the expected data. The pattern checker repeats this process for every cycle until the expected data is aligned and locked on a recognized pattern in the incoming data, which is determined by the link detection logic. When the link detection logic declares the link up, the pattern checker switches to the self-advance mode so that the embedded pattern generator starts to calculate and update the expected pattern every clock cycle on its own. As long as the link is up and stable, any bit error on the incoming data cannot disturb the actual output of the pattern checker, guaranteeing proper pattern verification and BER result. This auto-aligning and self-locking mechanism eliminates the need of transferring a special sequence during the initialization of a BER test, and makes the reference design capable of linking with an external BER tester.

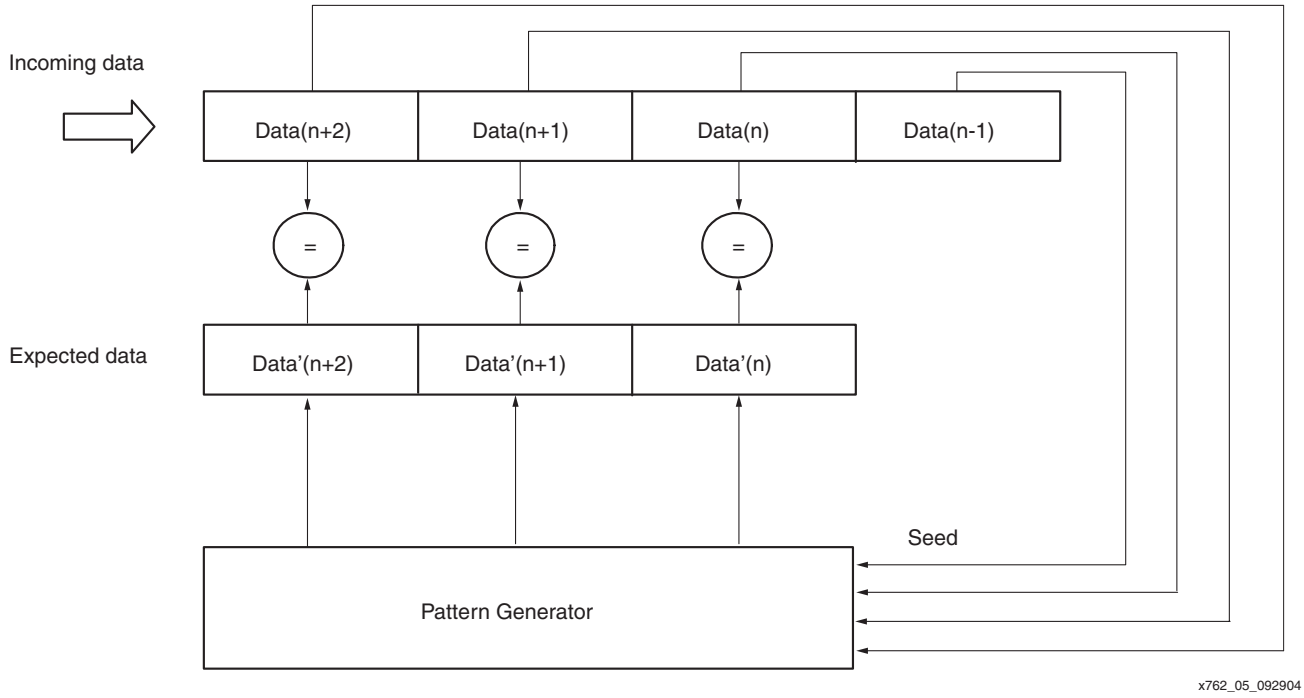


Figure 5: **Self-Synchronized Pattern Checker**

Figure 6 shows the state diagram of the link detection logic implemented in the pattern checker. The link detection logic monitors the number of bit errors in the received data at each cycle to report the link status. The logic declares a “link up” whenever it sees error-free incoming data for seven or more consecutive clock cycles. It declares a “link down” whenever it sees one or more bit errors at each cycle for seven or more consecutive clock cycles. The link status remains unchanged for all other conditions. The link detection logic filters out any scattered random bit errors occurring at a medium or low bit-error rate (BER) to keep the link relatively stable during a BER test.

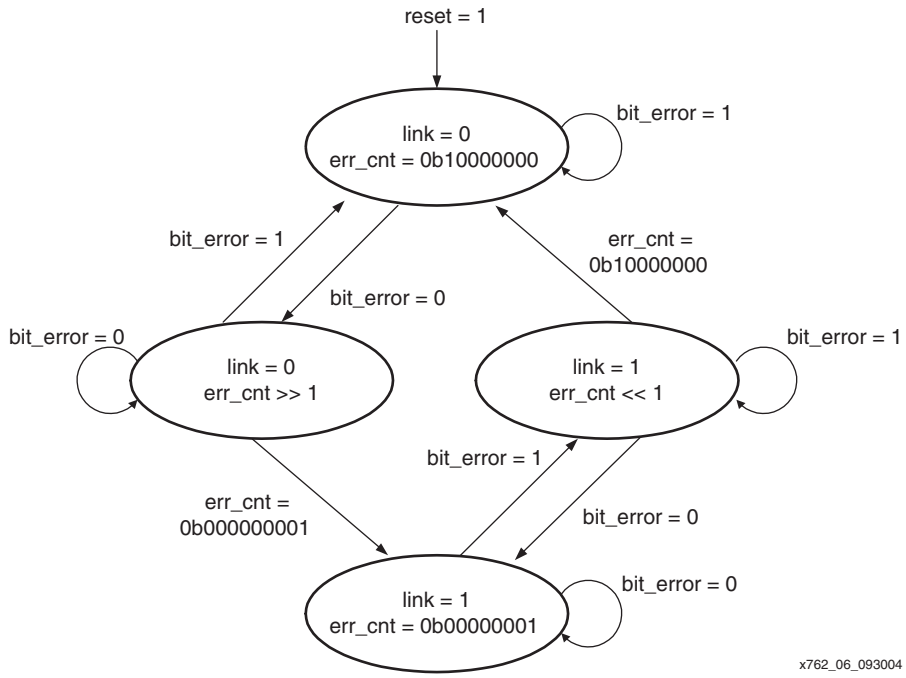


Figure 6: **Link Detection State Diagram**

Single-Channel XBERT Module

The single-channel XBERT is a self-contained module that is independent of the PPC405 subsystem. The single-channel XBERT module can be implemented and replicated in the FPGA fabric providing all necessary BER test functions.

Figure 7 shows a block diagram of a single-channel XBERT module implemented in the XBERT reference design. A single-channel XBERT module consists of the following main components:

- Two RocketIO X MGTs (MGT A and MGT B)

  Both MGTs share the buffered TXOUTCLK from MGT A to drive the TXUSRCLK/TXUSRCLK2 ports. They must operate at the same serial rate using the same PMA_SPEED mode.

- Two pattern checkers connect to two MGTs
- One pattern generator shared by two MGTs
- One clock/reset module

  This module contains two global clock buffers (BUFGs) for buffering RXRECCLK and generating RXUSRCLK and RXUSRCLK2. It also generates clock status outputs (dcm_locked) by detecting RXRECCLK.

- Two comma align control modules

  These modules are enabled when comma_align_enable is High. Each module attaches to one MGT for toggling the ENPCOMMAALIGN and ENMCOMMAALIGN ports. It indicates “aligned” and disables comma alignment on the MGT once the pattern checker indicates “link up” and detects seven idles in the incoming data.

- Two bit error counters

  Each counter aggregates the number of erroneous bits detected in the pattern checker into a 32-bit value. Each counter is updated at every clock cycle and can be cleared by a dedicated reset (ber_cnt_reset).

- Two RX word counters

  Each counter is a free running 32-bit counter on the RXUSRCLK2 domain that produces a total number of received words. This number is necessary for calculating a BER. The user can issue a dedicated reset (ber_cnt_reset) to clear each counter.

- Two echo buffers

  These buffers provide the back-end echo function that reflects the incoming data back to the transmission. Each buffer is a 16-deep asynchronous FIFO constructed using the dual-port distributed RAM. The write port is clocked on RXUSRCLK2, and the read port is clocked on TXUSRCLK2. These buffers can handle phase differences but not frequency differences between two clocks.

- One ChipScope Pro ILA core

  This core is clocked on the RXUSRCLK2 signal of MGT A. Data from both MGTs and associated internal logic are multiplexed by the mgt_sel input before being fed into the ILA core.

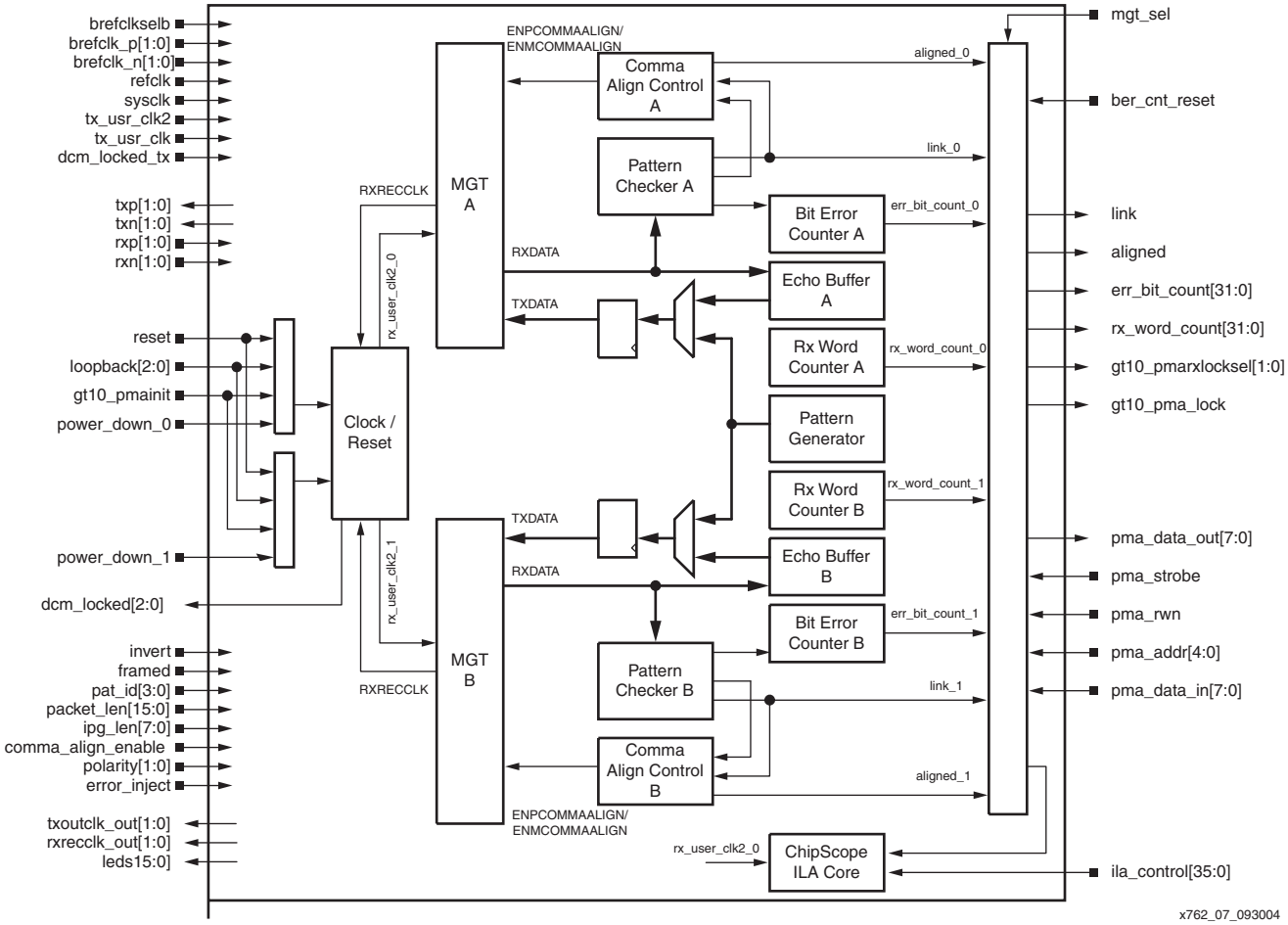


Figure 7: **Single-Channel XBERT Module**

Multi-Channel XBERT Module

The multi-channel XBERT module contains multiple instances (channels) of the single-channel XBERT module, grouped into top channels and bottom channels using a default placement configuration. Figure 8 is a block diagram of the multi-channel XBERT module. In this example, MGTs are placed at opposite edges of the FPGA, where even channels instantiate MGTs at the bottom edge of the FPGA, and odd channels instantiate MGTs at the top edge of the FPGA.

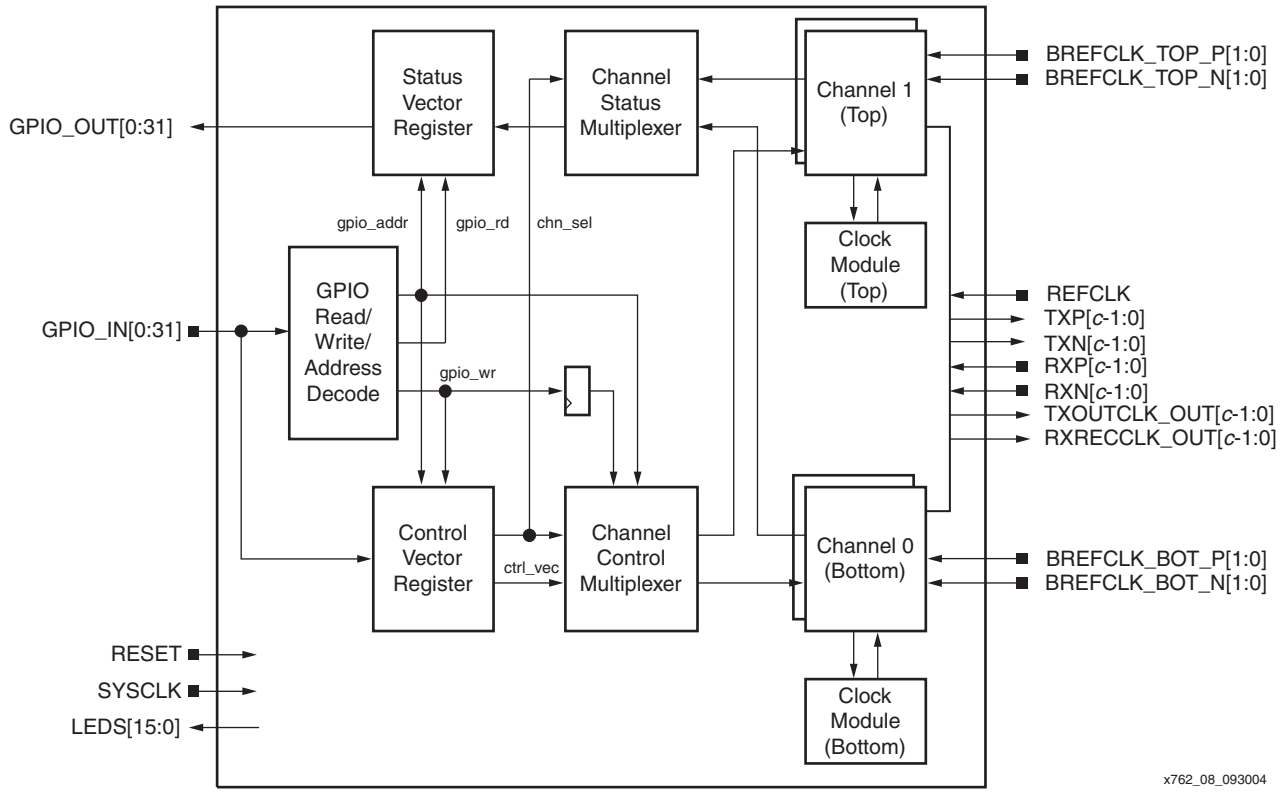


Figure 8: **Multi-Channel XBERT Module**

The user-accessible ports on the multi-channel XBERT module are defined in Table 4. The GPIO interface signals (GPIO_IN and GPIO_OUT) use big-endian bit ordering while the remaining signals on the multi-channel XBERT module use little-endian bit ordering.

Table 4: **Interface Signals of the Multi-Channel XBERT Module**

Name	Direction	Description
SYSCLK	Input	System clock. Typically 100 MHz.
RESET	Input	Active-High reset synchronous to the SYSCLK.
GPIO_IN[0:31]	Input	32-bit GPIO input from the microprocessor, synchronous to SYSCLK.
GPIO_OUT[0:31]	Output	32-bit GPIO output to the microprocessor, synchronous to SYSCLK.
BREFCLK_TOP_P[1:0]	Input	Differential BREFCLK inputs to all MGTs on the top edge of the FPGA. These ports must connect to the dedicated pads on the FPGA. Only bit 0 is used. Bit 1 can be left open.
BREFCLK_TOP_N[1:0]	Input	
BREFCLK_BOT_P[1:0]	Input	Differential BREFCLK inputs to all MGTs on the bottom edge of the FPGA. These ports must connect to the dedicated pads on the FPGA. Only bit 0 is used. Bit 1 can be left unconnected.
BREFCLK_BOT_N[1:0]	Input	
REFCLK	Input	Single-ended REFCLK input to all MGTs in the FPGA.
TXOUTCLK_OUT[*c**2-1:0]	Output	Non-buffered TXOUTCLK outputs from every MGTs implemented in the design. Each channel outputs two bits as follows: Bit 0 is the output of MGT A in the channel, and Bit 1 is the output of MGT B in the channel.
RXRECCLK_OUT[*c**2-1:0]	Output	Non-buffered RXRECCLK outputs from every MGT implemented in the design. Each channel outputs two bits as follows: Bit 0 is the output of MGT A in the channel, and Bit 1 is the output of MGT B in the channel.

Table 4: **Interface Signals of the Multi-Channel XBERT Module *(Continued)***

Name	Direction	Description
LEDS[15:0]	Output	Clock and link status outputs that can drive external LEDs. Bits 0 to 7 indicate the clock status of each channel. Bit 0 corresponds to channel 0. Bit 7 corresponds to channel 7 (if implemented). Each bit can be: 1: MGT user clocks (TXUSRCLK, TXUSRCLK2, RXUSRCLK, and RXUSRCLK2) on both MGTs for this channel are all detected and steady. 0: At least one of the MGT user clocks is not detected on one or both MGTs for this channel, indicating at least one MGT in this channel does not function properly. Bit 8 to 15 indicate the link status of each channel. Bit 8 corresponds to channel 0. Bit 15 corresponds to channel 7 (if implemented). Each bit can be: 1: Both MGTs in this channel establish steady links. 0: At least one MGT in this channel cannot establish a steady link due to excessive bit errors.
TXP[*c**2-1:0]	Output	MGT differential ports that transmit serial data. Each channel outputs two bits as follows: Bit 0 is the output of MGT A in the channel, and Bit 1 is the output of MGT B in the channel.
TXN[*c**2-1:0]	Output	
RXP[*c**2-1:0]	Input	MGT differential ports that receive serial data. Each channel outputs two bits as follows: Bit 0 is the output of MGT A in the channel, and Bit 1 is the output of MGT B in the channel.
RXN[*c**2-1:0]	Input	

Notes:

1. c = C_NUM_OF_CHANNEL, a parameter that sets the number of channels implemented in the multi-channel XBERT module.

The multi-channel XBERT module provides a GPIO interface that can attach to a microprocessor (for example, PPC405 processor). The GPIO provides register-based control and status over 32 input pins and 32 output pins. To expand the number of control/status bits transferring from/to the microprocessor on the GPIO interface, multiple control/status vectors are memory mapped to the GPIO interface at different addresses.

Table 5 and Table 6 list the bit definitions of the 32-bit GPIO input (GPIO_IN) and GPIO output (GPIO_OUT) registers, respectively.

Table 5: **Bit Definitions of the 32-bit GPIO Input (GPIO_IN[0:31]) Register**

Bit	Name	Description
[0:1]	OPCODE	Operation Code. 0b11: GPIO write operation. GPIO_IN carries a control vector (CTRL) at the current cycle. The address of this control vector is set by ADDR at the current cycle. 0b01: GPIO read operation. GPIO_OUT carries a status vector (STAT) at the next cycle. The address of this status vector is set by ADDR at the current cycle. Others: Reserved
[2:4]	ADDR	Address of the control or status vector.
[5:7]	n/a	Reserved
[8:31]	CTRL	24-bit control vector transferred from the microprocessor to the data plane (the XBERT module).

Table 6: **Bit Definitions of the 32-bit GPIO Output (GPIO_OUT[0:31]) Register**

Bit	Name	Description
[0:31]	STAT	32-bit status vector transferred from the data plane (the XBERT module) to the microprocessor.

Table 7 defines the bits of the 24-bit control vector at various addresses. Table 8 defines the bits of the 32-bit status vector at various addresses.

Table 7: **Bit Definitions of the 24-bit Control Vector (CTRL[8:31])**

Address	Bit	Name	Description
0	[8:13]	n/a	Reserved
	[14]	INVERT	Sets the pattern inversion of the output pattern on both MGTs in the selected channel. 1: Inverts the output pattern 0: Resumes regular output pattern
	[15]	BER_CNT_RESET	When asserted High, resets the counters (RX_WORD_COUNT and ERR_BIT_COUNT) of the selected MGT in the selected channel.
	[16]	ERROR_INJECT	When asserted High, the rising edge of this signal triggers an error injection of one clock cycle on both MGTs in the selected channel.
	[17:18]	POLARITY	Sets the TX and/or RX polarity on both MGTs in the selected channel. 0b01: Reverses RXP/RXN polarity, and resumes regular TXP/TXN polarity 0b10: Reverses TXP/TXN polarity, and resumes regular RXP/RXN polarity 0b11: Reverses both RXP/RXN and TXP/TXN polarities 0b00: Resumes regular polarity on both RXP/RXN and TXP/TXN
	[19:21]	LOOPBACK	Sets the loopback modes on both MGTs in the selected channel. 0b001: Activates internal parallel loopback mode 0b010: Activates post-driver serial loopback mode 0b011: Activates pre-driver serial loopback mode 0b100: Activates back-end echo mode
	22	POWER_DOWN	When asserted High, powers down both MGTs in the selected channel.
	23	BREFCLKSELB_BOT	Sets the reference clock source for the MGTs on the bottom edge of the FPGA. 1: Selects BREFCLK 0: Selects REFCLK
	24	BREFCLKSELB_TOP	Sets the reference clock source for the MGTs on the top edge of the FPGA. 1: Selects BREFCLK 0: Selects REFCLK
	25	n/a	Reserved
	[26:30]	CHN_SEL	Selects the channel implemented in the multi-channel XBERT module, allowing reads and/or writes of the control/status vectors dedicated to this channel.
	31	MGT_SEL	Selects one of MGTs (MGT A and MGT B) in the selected channel, allowing reads and/or writes of the control/status vectors dedicated to this MGT. 0: Selects MGT A 1: Selects MGT B

Table 7: **Bit Definitions of the 24-bit Control Vector (CTRL[8:31]) *(Continued)***

Address	Bit	Name	Description
1	[8:16]	n/a	Reserved
	[17:24]	PMA_DATA_IN	Data input to the PMA attribute programming bus for the selected MGT in the selected channel.
	[25:29]	PMA_ADDR	Address to the PMA attribute programming bus for the selected MGT in the selected channel.
	30	PMA_RWN	Read/write control to the PMA attribute programming bus for the selected MGT in the selected channel. 0: Writes data to the PMA attribute programming bus 1: Reads data from the PMA attribute programming bus
	31	PMA_STROBE	When asserted High, reads or writes (controlled by PMA_RWN) the data from or to the PMA attribute programming bus. PMA_ADDR, PMA_RWN, and PMA_DATA_IN must be stable during assertion of PMA_STROBE.
2	[8:23]	FRAME_LEN	Sets the frame length of the counter pattern on both MGTs in the selected channel. The length of a frame begins from the start word of the counter value following an IFG and ends at the last word of the counter value prior to the next IFG. The length of a frame can be from one word to 65,535 words.
	[24:25]	n/a	Reserved
	[26:29]	PAT_ID	Selects a pattern to generate and verify on both MGTs in the selected channel. Refer to Table 3, page 7 for a list of supported patterns.
	30	FRAMED	Enable the framed transmission on both MGTs in the selected channel. This condition requires using the framed counter pattern and enabling comma alignment in the selected channel.
	31	COMMA_ALIGN_ENABLE	Enable the comma alignment on both MGTs in the selected channel. This condition requires using the idle pattern or counter pattern in the selected channel. Comma alignment is a prerequisite to establishing a link using the counter pattern.
3	[8:23]	n/a	Reserved
	[24:31]	IFG_LEN	Sets the length of the inter-frame gap (IFG) in the counter pattern for both MGTs in the selected channel. The length of an IFG counts from the start word of an IFG, and ends at the last word within the same IFG. The length of an IFG can be from one word to 255 words.
4-7	Reserved		

Table 8: **Bit Definitions of the 32-bit Status Vector (STAT[0:31])**

Address	Bit	Name	Description
0	[0:7]	BITS_PER_WORD	Indicates the number of bits in a word configured in the implementation. Each word is either a 20-bit or a 40-bit vector that corresponds to the data width of the MGT fabric interface.
	[8:11]	n/a	Reserved
	[12:15]	NUM_OF_CHN	Indicates the total number of channels implemented in the multi-channel XBERT module.
	[16:23]	PMA_DATA_OUT	Registers the data output of the PMA attribute programming bus on the selected MGT in the selected channel.
	[24:25]	GT10_PMARXLOCKSEL	Indicates the determination of lock in the receiver PLL for the selected MGT in the selected channel. The determination of PLL lock (that is, the PMARXLOCKSEL value on the MGT) can be: 0b00: The receiver PLL automatically locks to incoming data (when present) or to the local reference clock (when data is not present) 0b01: The receiver PLL locks to the local reference clock 0b10: The receiver PLL locks to the received data 0b11: Reserved
	26	ALIGNED	Indicates the status of the comma alignment and word alignment on the selected MGT in the selected channel. 1: Comma alignment and word alignment are achieved on the selected MGT 0: Comma alignment and word alignment are disabled or in progress on the selected MGT
	27	LINK	Indicates the link status for the selected MGT in the selected channel. 1: The link is established on the selected MGT 0: The link is down on the selected MGT
	28	GT10_PMA_LOCK	Indicates the receiver PLL lock status for the selected MGT in the selected channel. 1: The receiver PLL has locked in the fine loop 0: The receiver PLL has not achieved lock
	[29:31]	DCM_LOCKED	Indicates the status of the MGT user clocks in the selected channel. Bit 29 is the status of RXUSRCLK and RXUSRCLK2 for MGT A. Bit 30 is the status of RXUSRCLK and RXUSRCLK2 for MGT B. Bit 31 is the status of TXUSRCLK and TXUSRCLK2 for both MGTs. Each bit can be: 1: The clock is detected and steady 0: The clock is not detected or unstable
1	[0:31]	RX_WORD_COUNT	Indicates the total number of received words for the selected MGT since the completion of a system reset or a counter reset. Each word is either a 20-bit or a 40-bit vector. This 32-bit value wraps around when it exceeds 4,294,967,295.
2	[0:31]	ERR_BIT_COUNT	Indicates the total number of bit errors in the received data on the selected MGT since the completion of a system reset or a counter reset. This 32-bit value wraps around when it exceeds 4,294,967,295. This number is only valid when the link is up for the selected MGT in the selected channel.

Table 8: **Bit Definitions of the 32-bit Status Vector (STAT[0:31]) *(Continued)***

Address	Bit	Name	Description
3	[0:7]	XBERT_TAG	Provides the FPGA device type. Bit 0 of this word indicates the FPGA type. This bit is always set to 1 to indicate a Virtex-II Pro X FPGA. Bits 1 to 7 is to indicate the device type. The two possible values of this word are: 0b10010100: XC2VPX20 device 0b11000110: XC2VPX70 device
	[8:15]	XBERT_VER1	Provides the first digit of the XBERT reference design version number.
	[16:23]	XBERT_VER2	Provides the second digit of the XBERT reference design version number.
	[24:31]	XBERT_REV	Provides the revision number of the XBERT reference design.
	[0:7]	BUILT_MONTH	Provides the decimal number of the build month for the implementation of this design.
	[8:15]	BUILT_DAY	Provides the decimal number of the build day for the implementation of this design.
	[16:31]	BUILT_YEAR	Provides the decimal number in the build year for the implementation of this design.
4	[0:15]	n/a	Reserved
	[16:23]	MGT_B_LOC	Indicates the MGT placement location of MGT B in the selected channel. 0 places the MGT at X0Y0, 1 places the MGT at X0Y1, 2 places the MGT at X1Y0, 3 places the MGT at X1Y1, and so on.
	[24:31]	MGT_A_LOC	Indicates the MGT placement location of MGT A in the selected channel. 0 places the MGT at X0Y0, 1 places the MGT at X0Y1, 2 places the MGT at X1Y0, 3 places the MGT at X1Y1, and so on.
5	Reserved		

Notes:

1. MGT is selected through the MGT_SEL bit in the control vector (CTRL). Refer to Table 7 for details.
2. Channel is selected through the CHN_SEL bits in the control vector (CTRL). Refer to Table 7 for details.

As shown in Figure 9, to write a control vector to the multi-channel XBERT module, the user sets the operation code (OPCODE = 0b11), the address (ADDR), and the control vector (CTRL) fields in the GPIO_IN register. The data is then held for least two clock cycles. The multi-channel XBERT module decodes OPCODE and ADDR, and stores the control vector in the internal register at the specified address until the next update. CHN_SEL and/or MGT_SEL also can be updated when address 0 is selected during a GPIO write.

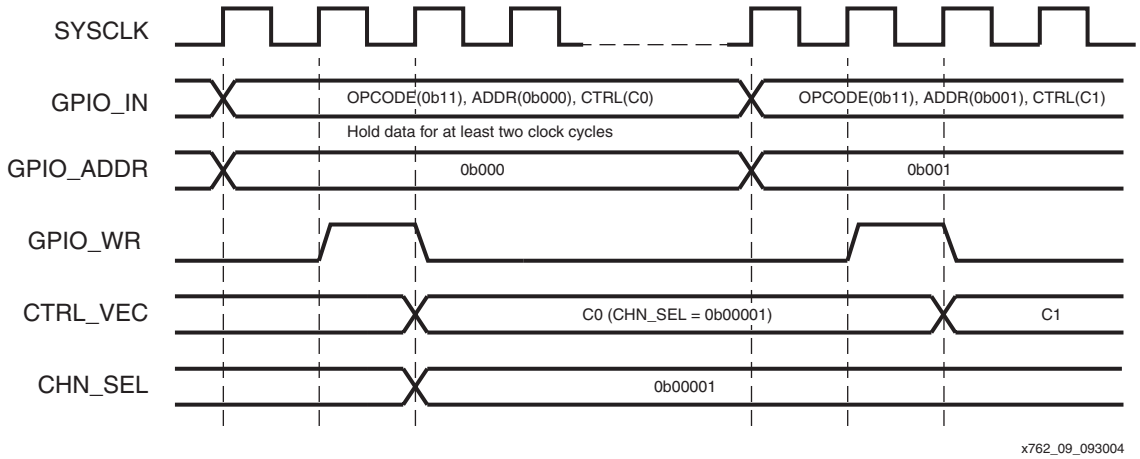


Figure 9: **Timing Diagram of GPIO Write on the Multi-Channel XBERT**

As shown in Figure 10, to read a status vector from the multi-channel XBERT module, the user sets the operation code (OPCODE = 0b01) and the address (ADDR) fields in the GPIO_IN register. The data is then held for at least two clock cycles. The status vector at specified address is present on the GPIO_OUT bus after two clock cycles. To read the status vector from a different channel or a different MGT in the same channel, the user first sets CHN_SEL and/or MGT_SEL in the control vector through a GPIO write. Data on the GPIO_OUT stays until the next GPIO read occurs.

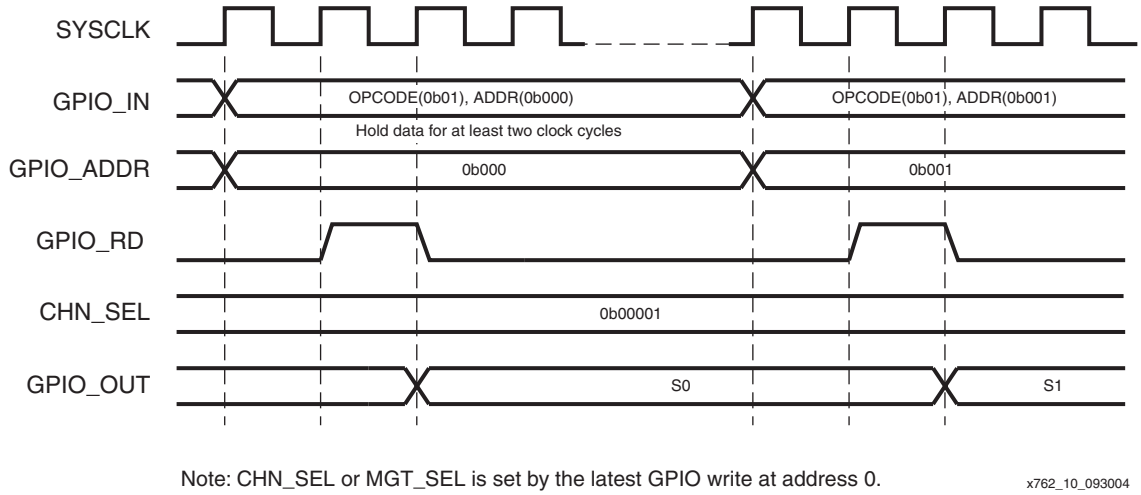


Figure 10: **Timing Diagram of GPIO Read on the Multi-Channel XBERT**

Control Plane Description

The UltraController Solution

As part of the control plane in the XBERT reference design, the UltraController solution utilizes the embedded PPC405 core and its instruction-side and data-side one-chip memory (OCM) interfaces. The OCM provides a direct connection to the PowerPC execution unit, eliminating the need for an interface bus, such as the processor local bus (PLB) or the on-chip peripheral bus (OPB). The UltraController block contains a data-side block RAM controller (D-Side Controller) and an instruction-side block RAM controller (I-Side Controller). Each BRAM controller serves as a dedicated interface between the block RAMs in the FPGA and the OCM signals available on the embedded PPC405 core. The I-Side Controller provides an interface to the 64-bit Instruction-side block RAM (I-Side BRAM), which is configured into 32 KByte memory in the reference design.

The D-Side Controller provides an interface to the 32-bit data-side block RAM (D-Side BRAM), which is configured into 16 KByte memory. Although the reference design supports memory depth expansion, the maximum amount of memory addressable by the I-Side Controller and D-Side Controller is 128 KBytes and 64 KBytes, respectively.

The I-Side and D-Side OCM interfaces operate at a 1:1 ratio of the processor clock. This clock is generated from the 2X output of the DCM module, as shown in Figure 11. To enable the 1:1 clock ratio on OCM interfaces, the user must set ISCNTLVALUE and DSCNTLVALUE to 0x81. Refer to Table 12, page 28 for settings on OCM interfaces.

The UltraController block uses the dual-port feature of the D-Side BRAM to enable a bidirectional data transfer on the 32-bit GPIO interface between the processor and the FPGA fabric. Read and write on the GPIO interface access two different addresses on the D-Side BRAM. A GPIO read transfers data from the FPGA logic (for example, the GPIO_OUT port of the multi-channel XBERT module) to address 4 of the D-Side BRAM. A GPIO write transfers data from address 0 of the D-Side BRAM to the FPGA logic (for example, the GPIO_IN port of the multi-channel XBERT module).

For details on the UltraController solution, refer to XAPP672: "The UltraController Solution: A Lightweight PowerPC Microcontroller."

Clock/Reset Distribution

The XBERT reference design uses a digital clock manager module (DCM_MODULE) to generate clocks for the control plane, as shown in Figure 11. The PPC405 core and the other modules (such as the I-Side Controller, the D-Side Controller, and the GPIO interface) in the control plane run at twice the frequency of the system clock input (sys_clk), which is typically 50 MHz. The reset signals for the PPC405 core and the rest of the design are generated by the PROC_SYS_RESET module and triggered by an external reset input (sys_rst). Both DCM_MODULE and PROC_SYS_RESET blocks are standard components provided in the EDK.

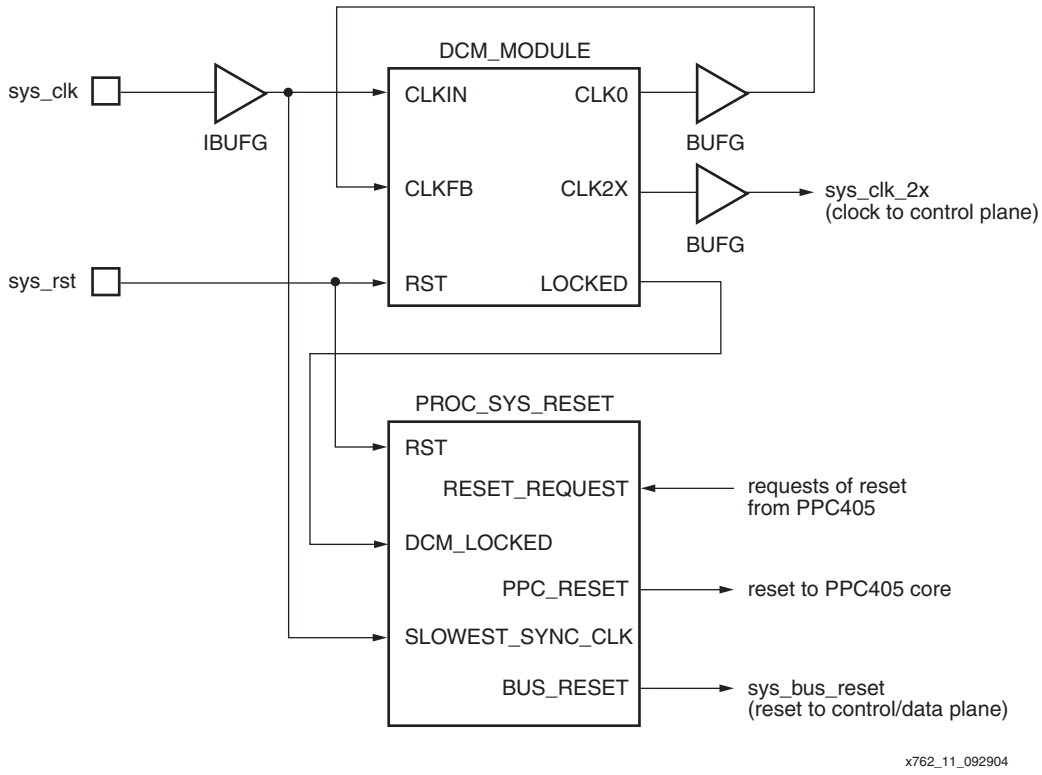


Figure 11: **Clock and Reset Distribution in the Control Plane**

The UART Core

The OPB UART Lite core is a free IP core provided in the EDK. This core requires a connection to the OPB bus. The XBERT reference design implements a DCR2OPB core that bridges between the DCR interface on the PPC405 processor and the OPB UART Lite core, eliminating the need of a PLB and a PLB-to-OPB Bridge and producing a lightweight UART solution.

The OPB UART Lite core sets the baud rate, the number of data bits, and the parity options as part of the hardware configuration. It does not support flow control. Refer to Table 12, page 28 for a list of configuration parameters on this core.

Software Description

Address Map

Software instruction and data are loaded into the I-Side BRAM and D-Side BRAM connected to the instruction-side and data-side OCM interfaces on the PPC405 processor. The UART is memory mapped to the OPB and bridged to the DCR on the processor. The multi-channel XBERT module is connected to the GPIO interface and bridged to the D-Side BRAM. Table 9 lists the software address map of the memory and DCR devices in the reference design.

Table 9: **Address Map of Memory and DCR Devices**

Device	Address Type	Address Boundaries		Size
		Upper	Lower (Base Address)	
I-Side BRAM in UltraController Block	Memory	0xFFFFFFFF	0xFFFF8000	32 KBytes
D-Side BRAM in UltraController Block	Memory	0xFE003FFF	0xFE000000	16 KBytes
32-bit GPIO Output Port in UltraController Block	Memory	0xFE000003	0xFE000000	4 Bytes
32-bit GPIO Input Port in UltraController Block	Memory	0xFE000007	0xFE000004	4 Bytes
I-Side OCM	DCR	0x05B	0x058	16 Bytes
D-Side OCM	DCR	0x05F	0x05C	16 Bytes
DCR2OPB Bridge	DCR	0x03F	0x000	256 Bytes
	Memory	0xA00000FF	0xA0000000	256 Bytes
OPB UART Lite	Memory	0xA00000FF	0xA0000000	256 Bytes(1)
Multi-Channel XBERT	Memory	0xFE000007	0xFE000000	8 Bytes

Notes:

1. The OPB UART Lite core only utilizes the lower 16 bytes of the memory space.

GPIO Software Code

The GPIO software code (`sw/gpio.c` and `sw/gpio.h`) contains three levels of GPIO functions:

- Level 0 functions provide reads and writes to the 32-bit GPIO interface.
- Level 1 functions provide reads and writes to the control and status vectors on the multi-channel XBERT module. These functions use level 0 functions to write OPCODE and ADDR, and then write or read the data on the GPIO interface to access the control and status vectors. Refer to Table 5, page 16 through Table 8, page 19 for bit definitions of these vectors.

- Level 2 functions use level 1 functions to provide channel-level functions such as channel setup, channel status query, and channel control.

Both level 1 and level 2 functions use two data structures, xbert_entry and chn_entry, as defined in the `xbert.h` file. The software allocates memory for one xbert_entry structure, reads the hardware configuration parameters from the multi-channel XBERT module, and stores the settings in this data structure. The software allocates memory for multiple chn_entry structures, depending on the number of channels implemented in the reference design. A chn_entry structure stores values of control and status vectors of one channel. Such data structure serves as an intermediate memory to transfer data between the control plane and the data plane (that is, the multi-channel XBERT module). The software reads the status from the data plane, stores the status data in the chn_entry structure, and then processes the data afterwards. On the other hand, the software reads user inputs, generates and stores the control data in the chn_entry structure for the selected channel, and then writes the data in chn_entry to the data plane, typically in a batch process.

Because the RX_WORD_COUNT and ERR_BIT_COUNT outputs on the multi-channel XBERT module are only 32 bits, these values are not large enough for storing overnight BER test results. For example, the 32-bit RX_WORD_COUNT counter rolls over in 17 seconds at 10 Gb/s. The software resolves this issue by detecting carries on these counters and expanding them to 64-bit counters (Rx Words # and Bit Errors #), stored in the chn_entry structure.

UART Software Code

The UART software code (`sw/uart.c` and `sw/uart.h`) contains two levels of UART functions:

- Level 0 functions provide UART initialization and read and write accesses through the DCR mtdcr and mfdcr instructions.
- Level 1 functions provide more complex UART functions, such as formatted printing, string inputting, and printing of hexadecimal numbers. By printing special ANSI characters, the software also provides functions to clear the terminal screen, reset the cursor position, and clear a line on the screen.

Top-Level Software Code

The top-level software code (`sw/xbert.c` and `sw/xbert.h`) provides a high-level software application for the XBERT reference design. Figure 12 illustrates the functional flow diagram of the top-level software application. After power-up, system reset, or a reconfiguration of the FPGA, the PPC405 processor starts executing the software to perform an initialization and displays the main menu on the terminal window. The main menu provides the following software functions:

- Display System Information

  Displays XBERT hardware configuration, such as the number of enabled MGTs, the number of channels.

- Setup Channels

  Queries and/or sets up PMA_SPEED mode, pattern, clock source, frame length, and so forth, on both MGTs in the selected channel.

- Test BER

  Runs the BER test on all channels and displays channel status and test results. All channel status is updated approximately every second by the PPC405 processor and is displayed on the terminal. However, the counter numbers (Rx Words # and Bit Errors #) are captured in real-time in the FPGA fabric logic.

- Edit PMA Attributes

  Views and/or edits PMA attributes on a selected MGT.

- Enter ChipScope Mode

  Uses the ChipScope core to monitor data on a selected MGT.

For detailed usage on the software application, refer to UG137: *RocketIO X BERT Reference Design User Guide*.

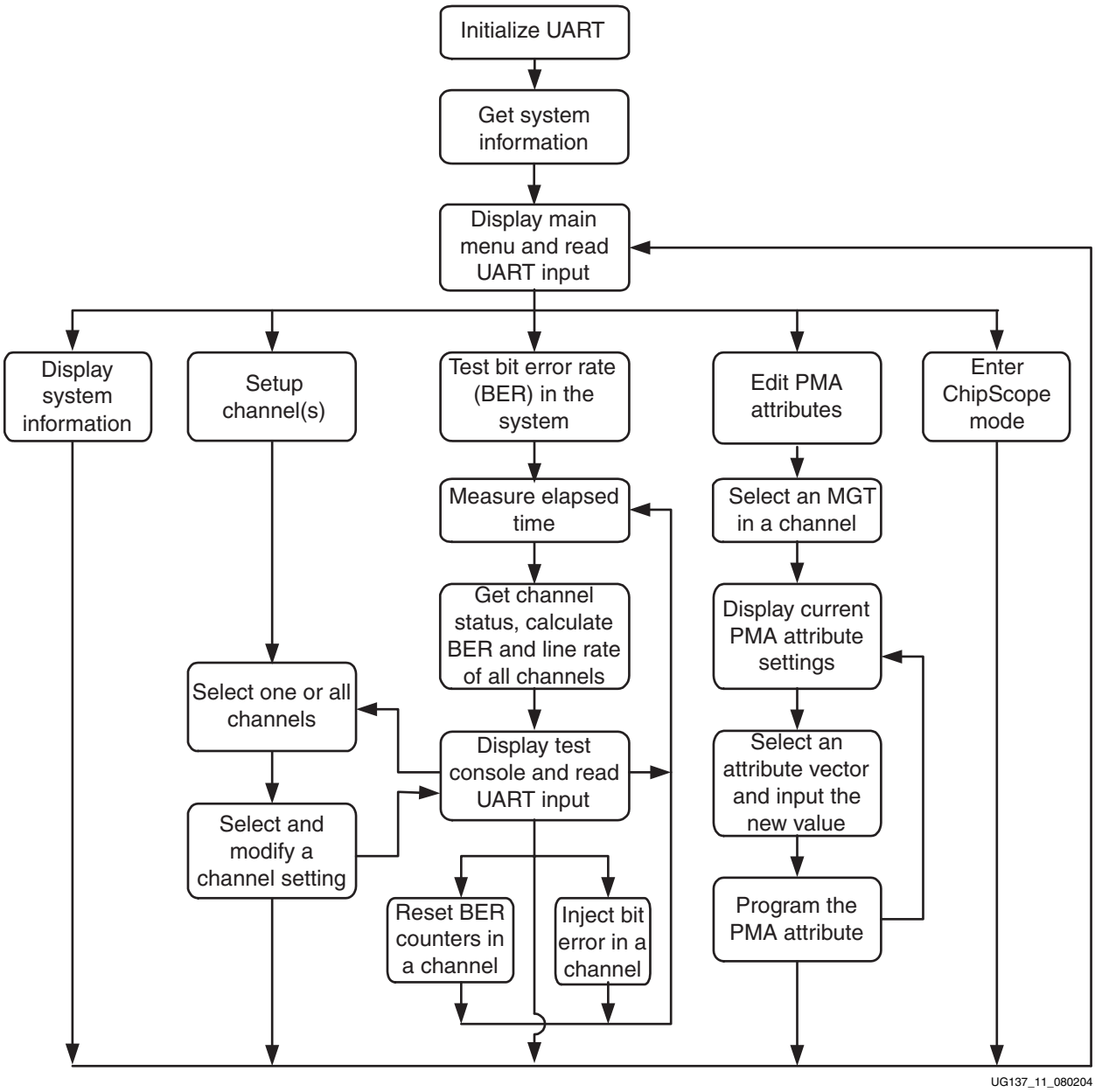


Figure 12: **Functional Flow Diagram of Top-Level Software Application**

The software uses the time functions supplied in the standard C library to calculate the elapse of time. These time functions provide access to the 64-bit time base counter inside the PPC405 processor core. The counter increases by one at every processor cycle. The software also uses a timing function called *usleep* to delay the execution of the following instructions by microseconds. Both time and usleep functions require the processor frequency (in Hz) to be defined in the MSS file as follows (the default processor frequency is 100 MHz):

```
BEGIN PROCESSOR
  PARAMETER HW_INSTANCE = PPC405_i
  PARAMETER DRIVER_NAME = cpu_ppc405
  PARAMETER DRIVER_VER = 1.00.a
  PARAMETER CORE_CLOCK_FREQ_HZ = 100000000
END
```

Calculation of Bit-Error Rate (BER)

The BER is the probability that a given bit is received in error. It also can be interpreted as the average number of bit errors that occur in a sequence of n bits received in a given period of time. To measure a statistically valid BER result, enough bits with enough bit errors must be received on the MGT. For example, a 5-minute BER test at 10 Gb/s receives 3 trillion bits. Such a test ensures the BER is less than 10^{-12} with 95% accuracy, if no errors are observed. A longer 30-minute test ensures that the BER is less than 10^{-12} with 99.999999% accuracy, if no errors are observed. Refer to XAPP661: "RocketIO Transceiver Bit-Error Rate Tester" for an illustration of the derivation of precision and confidence numbers for a BER result based on principles of stochastic methods.

The software in the XBERT reference design calculates the BER in real-time based on the counter numbers (Rx Words # and Bit Errors #) using the formula below. The software assumes that the next received bit contains an error. This hypothetical erroneous bit takes into account the calculation of the BER. Therefore, the BER can never equal 0 and should decrease when the BER test lasts (that is, when Rx Words # increases). The BER test requires the pattern checker in the multi-channel XBERT module to align and lock to the incoming data, so the calculated BER is only valid when the link is up.

$$\text{Bit Error Rate} = \frac{\text{Bit Errors \#} + 1}{\text{Rx Words \#} \times \text{BITS\_PER\_WORD} + 1}$$

Modification of PMA Attributes

The software in the XBERT reference design allows the user to view and modify some fractions (vectors) of PMA attributes at various addresses and bit locations, as listed in Table 10. *This feature is for advanced users only.* Direct modification of these attributes should only be done with a thorough understanding of the capabilities, performance, and side effects of the resulting settings. For details of these PMA attribute vectors, refer to the *RocketIO X Transceiver User Guide*.

Table 10: **Modifiable PMA Attribute Vectors in XBERT Reference Design**

Vector Name	PMA Attribute Address	Description
TXDOWNLEVEL[3:0]	0x5	Selects the transmit line driver current (i.e. output voltage swing).
PRDRVOFF	0x5	Disables and enables the line driver. The default is 0 (Enabled). When disabled, the output is kept at the common mode voltage.
EMPOFF	0x5	Disables and enables the emphasis feature in the line driver. The default is 0 (Enabled). When enabled, the emphasis level is set by TXEMPHLEVEL[3:0].
SLEW	0x5	Sets the slew rate of the line driver.
TXEMPHLEVEL[3:0]	0x6	Selects the transmit line driver emphasis current level (that is, emphasis voltage level)
RXLOOPFILTERC[1:0]	0x9	Selects the receiver PLL filter capacitor setting.
RXLOOPFILTERR[2:0]	0x9	Selects the receiver PLL filter resistor setting.

Table 10: **Modifiable PMA Attribute Vectors in XBERT Reference Design *(Continued)***

Vector Name	PMA Attribute Address	Description
RXFLTCPT[4:0]	0xB	Controls the phase adjustment of the receiver sample clock relative to the data.
VSELAFE[1:0]	0xC	Sets the receiver analog front end common mode input voltage.
RXFEI[1:0]	0xC	Sets the receiver front end/equalizer current.
RXFER[9:0]	0xC, 0xD	Sets the equalization in the receiver front end/equalizer, based on the adjustment of four basic boosts in four different frequency ranges.

The software performs a masked write operation on the MGT PMA attribute programming bus, that is, it performs a read-modify-write software operation to preserve the "reserved" values. The software also allows the user to scan for the optimal settings for a group of PMA attribute vectors. The software performs a brute force scan for all possible settings on these vectors, programs these settings into MGT PMA attributes, runs a series of BER tests for a short period for each setting, and then reports the optimal setting that yields the best BER result. This feature provides an auxiliary tool for users to find the best PMA attribute settings for their applications. However, a thorough understanding and characterization of their system must determine true optimal settings. The software allows combining of up to five PMA attribute vectors in a scan cycle, which can include multiple TX and/or RX settings on the MGT. For example, the output swing level (TXDOWNLEVEL) and the emphasis level (TXEMPHLEVEL) are often combined in a scan since they are related to each other.

Changing PMA_SPEED Mode

The XBERT reference design supports using several MGT PMA_SPEED modes on a single bitstream in order to target multiple MGT serial speeds. These PMA_SPEED modes must use the same data width and clock ratio on the MGT fabric interface. Refer to Table 2, page 6 for a list of PMA_SPEED modes supported in the XBERT reference design.

The software in the XBERT reference design uses an array (pma_mode_attr_value) to store default PMA attribute values for the supported PMA_SPEED modes. Changing the PMA_SPEED mode loads the default PMA attribute values from this array to the target MGT instantiated in the multi-channel XBERT module, overriding any previous changes on the PMA attributes. When changing from a slower BREFCLK to a faster BREFCLK during a PMA_SPEED mode switch, it is recommended to keep the slower clock during the mode switch. When changing from a faster BREFCLK to a slower BREFCLK, change the clock before switching the mode.

Design Configuration

Hardware Configuration Parameters

The XBERT reference design is delivered in two typical configurations. Each comes with a Microprocessor Hardware Specification (MHS) file. These two MHS files accommodate a single processor and a dual processor Virtex-II Pro X device, as shown in Table 11.

Table 11: **XBERT Reference Design MHS Files**

MHS File	Applicable Device	Number of Channels(1)
`system_2ch_1cpu.mhs`	Virtex-II Pro X FPGA with single processor (XC2VPX20)	2
`system_4ch_2cpu.mhs`	Virtex-II Pro X FPGA with dual processors (XC2VPX70)	4

Notes:

1. The number of channels is configurable (see Table 12 for details).

Table 12 lists the available hardware configuration parameters in the reference design. These parameters can be modified at the indicated locations before an implementation of the reference design. Various files that contain these hardware configuration parameters include:

- The XMP file: `system.xmp`
- The implementation script: `Makefile`
- The MHS file: `system.mhs`
- The MSS file: `system.mss`
- The UCF file: `data/system.ucf`
- The Verilog configuration file: `data/config.v`
- The linker script: `sw/linker_script`

Table 12: **Hardware Configuration Parameters**

Module	Parameter	Description	Acceptable or Typical Settings	Where and What to Change
n/a	Device	The device and package type of the target FPGA	xc2vpx20 or xc2vpx70	Change these parameters in the XMP file or modify them in the Platform Studio.
	Package		FF896 or FF1704	Change this parameter in the implementation script.
XBERT	C_NUM_OF_CHANNEL	The number of channels implemented in the reference design	1 to 5	• Change this parameter on the xbert module in the MHS file. • Change the port data width on signals TXP, TXN, RXP, RXN, TXOUTCLK_OUT, and RXRECCLK_OUT in the MHS file. • Modify the UCF file to accommodate all channels.
	USE_STEPPING_0	Selects whether to use stepping-0 FPGA or stepping-1 FPGA.	Defined or undefined	Change this parameter in the Verilog configuration file by defining or undefining this parameter.
	XBERT_VER1, XBERT_VER2, XBERT_REV	Defines XBERT version numbers.	8'd3, 8'd0, 8'd2	Define these parameters in the Verilog configuration file.
	BUILT_MON, BUILT_DAY, BUILT_YEAR	Defines the build date of the reference design	8'd8 8'd23 16'd2004	Define these parameters in the Verilog configuration file.
	XBERT_TAG	Indicates the device type to the software. Bit 7 selects the type of FPGA: 1 is Virtex-II Pro X FPGA, 0 is invalid. Bits 6 to 0 set the device type: 20 is 2vp20, 70 is 2vp70.	XC2VPX20: 8'h94 XC2VPX70: 8'hC6	• Define this parameter in the Verilog configuration file. • Modify device and package type in the XMP file or the Platform Studio to match this setting.
	USE_PMA_13_40, USE_PMA_24_20	Sets the target PMA_SPEED mode used in the reference design	Defined or undefined	• Define only one of these parameters in the Verilog configuration file. • Modify the UCF file on MGT PMA_SPEED settings.

Table 12: **Hardware Configuration Parameters *(Continued)***

Module	Parameter	Description	Acceptable or Typical Settings	Where and What to Change
XBERT (cont.)	CHN_*_MGT_A, CHN_*_MGT_B	Sets placement of the two MGTs (MGT A and MGT B) in each channel. 0 places the MGT at X0Y0, 1 places the MGT at X0Y1, 2 places the MGT at X1Y0, 3 places the MGT at X1Y1, and so on.	0 to 19	• Define these parameters in the Verilog configuration file. • Modify the UCF file on MGT LOC settings.
	TOP_TXOUTCLK_CHN, BOT_TXOUTCLK_CHN	Selects a channel on the top or the bottom edge of the FPGA to drive TXUSRCLK and TXUSRCLK2 for all MGTs on the same edge.	0 to C_NUM_OF_CHANNEL - 1	Define this parameter in the Verilog configuration file.
	USE_ILA	Selects whether to implement or remove the embedded ChipScope ILA core.	Defined or undefined	• Define this parameter in the Verilog configuration file. • Modify the UCF file to include constraints for ChipScope ILA core.
	USER_PATTERN_H, USER_PATTERN_L	Defines the content of a user pattern. A 20-bit user pattern is a combination of these two parameters. A 40-bit user pattern is a replication of two 20-bit user patterns. USER_PATTERN_H is transmitted and received first.	0b0011111010, 0b1100000101	Define this parameter in the Verilog configuration file.
	USE_BACKEND_LOOPBACK	Selects whether to implement or remove the back-end echo loopback function.	Defined or undefined	Define this parameter in the Verilog configuration file.
	WITH_TYPE2	When defined, implements the Galois LFSR in the PRBS pattern generation. When undefined, implements the Fibonacci LFSR.	Defined or undefined	Define this parameter in the Verilog configuration file.

Table 12: **Hardware Configuration Parameters *(Continued)***

Module	Parameter	Description	Acceptable or Typical Settings	Where and What to Change
PPC405	C_ISOCM_DCR_BASEADDR, C_ISOCM_DCR_HIGHADDR	I-Side OCM DCR base address and high address	0x058, 0x05B	Change these parameters on the ppc405 module in the MHS file.
	C_DSOCM_DCR_BASEADDR, C_DSOCM_DCR_HIGHADDR	D-Side OCM DCR base address and high address	0x05C, 0x05F	Change these parameters on the ppc405 module in the MHS file.
	CORE_CLOCK_FREQ_HZ	Processor frequency in Hz	100000000 decimal	• Change this parameter on the processor module in the MSS file or modify it in the Platform Studio (`Options-> Compiler Options -> Environment`) • Change the dcm_module in the MHS file to adjust the clock multiplier ratio. • Change the UCF file to modify the clock constraint on sys_clk_1x and sys_clk_2x.
	StackSize	The stack size of the processor	1024 decimal	Change this parameter in the XMP file or modify it in the Platform Studio (`Options-> Compiler Options -> Details`)
	HeapSize	The heap size of the processor	4	Change this parameter in the XMP file or modify it in the Platform Studio (`Options-> Compiler Options -> Details`)
OPB UART Lite	C_BASEADDR, C_HIGHADDR	OPB UART Lite registers base address and high address	0xA0000000, 0xA00000FF	Change these parameters for the opb_uartlite module in the MHS file.
	C_DATA_BITS	The number of data bits in the serial frame of the UART	8	Change this parameter for the opb_uartlite module in the MHS file.
	C_CLK_FREQ	Clock frequency of the OPB system clock driving the UART Lite peripheral in Hz	100000000 decimal	Change this parameter on the opb_uartlite module in the MHS file.
	C_BAUDRATE	Baud rate of the UART Lite in bits per second	57600 decimal	Change this parameter for the opb_uartlite module in the MHS file.
	C_USE_PARITY	Determines whether or not parity is used on the UART Lite.	0	Change this parameter for the opb_uartlite module in the MHS file.
	C_ODD_PARITY	If parity is used on the UART Lite, determines whether parity is odd or even	0	Change this parameter for the opb_uartlite module in the MHS file.
I-Side OCM	C_ISCNTLVALUE	Power-on configuration of the PowerPC I-Side OCM interface controller	0x81	Not recommended. Change this parameter for the isocm_v10 module in the MHS file.
D-Side OCM	C_DSCNTLVALUE	Power-on configuration of the PowerPC D-Side OCM interface controller	0x81	Not recommended. Change this parameter for the dsocm_v10 module in the MHS file.
I-Side BRAM	C_BASEADDR, C_HIGHADDR	I-Side BRAM base address and high address, that is, the BRAM size	0xFFFF8000, 0xFFFFFFFF	• Only change the base address on the isbram_if_cntlr module in the MHS file. Do not change the high address. • Change the linker script on the size and the start address of the I-Side OCM.

Table 12: **Hardware Configuration Parameters *(Continued)***

Module	Parameter	Description	Acceptable or Typical Settings	Where and What to Change
D-Side BRAM	C_BASEADDR, C_HIGHADDR	I-Side BRAM base address and high address, that is, the BRAM size	0xFE000000, 0xFE003FFF	• Only change the high address for the dsbram_if_cntlr module in the MHS file. Do not change the base address. • Change the linker script on the size of the D-Side OCM.
DCR2OPB	C_BASEADDR, C_HIGHADDR, C_ADDRMASK	DCR2OPB bridge, DCR base address, high address, and address mask	0x000, 0x03F, 0x3C0	Change these parameters for the dcr2opb_bridge module in the MHS file.
	C_OPB_BASEADDR, C_OPB_HIGHADDR, C_OPB_OFFSET	DCR2OPB bridge, OPB base address, high address, and address offset	0xA0000000, 0xA00000FF, 0xA0000000	Change these parameters for the dcr2opb_bridge module in the MHS file.

Software Configuration Parameters

Table 13 lists the available software configuration parameters in the reference design. These parameters can be modified at the indicated locations before a compilation of the software.

Table 13: **Software Configuration Parameters**

Parameter	Description	Where and What to Change
DEFAULT_PMA_SPEED_MODE_40BIT	The default PMA_SPEED mode loaded after a system reset when using the 40BIT configuration(1)	Change this setting in the `sw/xbert.h` file
DEFAULT_PMA_SPEED_MODE_20BIT	The default PMA_SPEED mode loaded after a system reset when using the 20BIT configuration(1)	Change this setting in the `sw/xbert.h` file
pma_mode_attr_value	The default PMA attribute settings for each PMA_SPEED mode	Not recommended. Change this array in the `sw/gpio.c` file.
MAX_COMBINED_ATTR_NUM	The maximum number of PMA attribute vectors that can be combined to scan for optimal setting	Change this setting in the `sw/xbert.h` file

Notes:

1. Refer to Table 2, page 6 for details on 40BIT and 20BIT configurations.

Reference Design

Design Hierarchy

The directory structure of the XBERT reference design is shown in Figure 13. This tree does not show temporary directories that are generated during the design implementation.

```
<XBERT_ROOT>                        Readme file, environment setup scripts,
|                                   EDK project files (XMP, MHS, MSS) and
|                                   implementation scripts (Makefile)
|
|--- __xps                          Optional files for various EDK tools
|
|--- etc                            Optional files for implementation tools
|
|--- chipscope                      ChipScope netlists
|
|--- data                           Verilog configuration files and UCF files
|
|--- demo                           Demo files (bitstreams, ACE files,
|                                   ChipScope project file) for MK32x platform
|--- pcores                         Customized IP cores for the EDK project
|    |
|    |--- dcr2opb_bridge_v1_00_a    DCR2OPB bridge core
|    |
|    |--- uc_gpio_ref_v1_00_a       GPIO interface core
|    |
|    |--- xbert_v1_00_a             Multi-channel XBERT core
|
|--- sw                             Software codes
|
|--- testbench                      MTI behavioral Simulation testbench and
|                                   scripts
```

Figure 13: **XBERT Directory Structure**

Table 14 lists some important source files delivered with the reference design.

Table 14: **Important Source Files Included with the XBERT Reference Design**

File Name	Description
./readme.txt	Readme file that contains the version number, the revision history, and the EDA tool version of the reference design.
./system.xmp	The Xilinx Microprocessor Project (XMP) file for the reference design. The Platform Studio (XPS) can load this project file.
./system.mhs	The Microprocessor Hardware Specification (MHS) file for the reference design. Use ./system_2ch_1cpu.mhs or ./system_4ch_2cpu.mhs to replace this file to layout a typical XBERT reference design specification.
./system.mss	The Microprocessor Software Specification (MSS) file for the reference design.
./Makefile	The EDK implementation script used in place of system.make and system_incl.make. Some dedicated design implementation options are included in this script but not in the system.make and system_incl.make files.
./config.csh	A UNIX environment setup file for the reference design.
./config.linux	A Linux environment setup file for the reference design.
./config.bash	A Windows environment setup file for the reference design. Use BASH shell to execute this script.

Table 14: **Important Source Files Included with the XBERT Reference Design *(Continued)***

File Name	Description
./build_mk32x.sh	A script to build all demo bitstreams on the MK32x platform.
./data/config.v	The Verilog configuration file that contains most XBERT hardware configuration parameters. A series of predefined configuration files (config_mk32x_*bit.v) are provided in the same directory. Use one of these files to replace the config.v file when building MK32x demos.
./data/system.ucf	The user constraint file (UCF). A series of MK32x platform UCF files (mk32x_*bit_*.ucf) are provided in the same directory. Use one of these files to replace the system.ucf file when building MK32x demos.
./sw/linker_script	The linker script for compilation of the software application.
./sw/xbert.c	The C code of the top-level software application.
./sw/xbert_sim.c	The simplified C code of the top-level software application for simulation purposes only. All UART function calls were removed from this application to speed up the simulation.

Installation of the Reference Design and Tools

The following steps provide the procedure for installing the reference design and related tools:

1. Install the Xilinx ISE software, ModelSim SE, and EDK tools. Refer to the Readme file in the reference design for the recommended versions.
2. Install the SmartModel Library supplied in the Xilinx ISE software. Xilinx Answer Records #15501 and #14019 give further information regarding the SmartModel/Swift Interface and the installation of SmartModels.
3. Extract xapp762.zip into a directory <XBERT_ROOT>.
4. Modify the config.bash (on Windows) or config.csh (on UNIX) file under <XBERT_ROOT> to update all paths of the tools and libraries as specified in these files. Most importantly, set the DUT_ROOT, MTI_LIBS, and edk_nd_libs variables to point to valid paths.
5. Call the config.bash or config.csh script to set up the environment on Windows or UNIX:

   For Windows, launch XYGWIN, enter <XBERT_ROOT>, and then type the following command:

   ```
   > source config.bash
   ```

   For UNIX, enter <XBERT_ROOT>, and then type the following command:

   ```
   > source config.csh
   ```

6. Run the compxlib tool to compile simulation libraries (unisim, simprim, etc.) of Virtex-II Pro family for ModelSim. Library should be compiled into the $MTI_LIBS directory defined in the config.bash or config.csh file. The following is an example use of the compxlib tool:

   ```
   > compxlib -s mti_se -f virtex2p -l all -o $MTI_LIBS
   ```

7. Run compedklib tool to compile behavioral libraries of IP cores provided in EDK for ModelSim. Library should be compiled into the $edk_nd_libs directory defined in config.bash or config.csh. The following example uses the compedklib tool:

   ```
   > compedklib -s mti_se -o $edk_nd_libs -X $MTI_LIBS
   ```

   Note: Steps 6 and 7 also can be done using Xilinx Platform Studio (Options->Project Options-> HDL and Simulation -> Simulation Libraries Path -> Click on "Compile").

8. Open the <XBERT_ROOT>/system.xmp file from a text editor. Change all paths in this file to valid paths on UNIX or Windows based on your installation of the reference design.
9. After completing the above steps, run Step 5 again after restarting Windows or the UNIX system.

Behavioral Simulation

Follow these steps to perform a behavioral simulation of the reference design:

1. Go to the root directory of the reference design.
2. Call the config.bash or config.csh script to set up the environment on Windows or UNIX.
3. Copy system_2ch_1cpu.mhs to system.mhs. The behavioral simulation on the reference design only supports simulating the single processor in a two-channel configuration.
4. Launch the Xilinx Platform Studio (XPS) GUI. If using Windows, be sure to launch XPS in the XYGWIN shell by typing the following command to ensure the MTI_LIBS, edk_ndlibs, MODELSIM, and other environment variables set in config.csh or config.bash take effect:

   ```
   > xps
   ```

5. Select File->Open Project-> Load system.xmp project.
6. Select Options->Project Options-> HDL and Simulation -> Simulation Libraries Path -> Modify EDK Library and Xilinx Library to point to paths defined by $edk_nd_libs and $MTI_LIBS, where the EDK behavioral libraries, and the unisim and simprim simulation libraries are located. Also select "Behavioral" as the simulation model.
7. Select Tools->Start HDL Simulator-> Compile simulation libraries and launch ModelSim. If ModelSim is not automatically launched, run ModelSim manually.
8. In ModelSim, select File->Change Directory->Go to <XBERT_ROOT>/simulation/behavioral.
9. In ModelSim, run the following command to compile the reference design testbench:

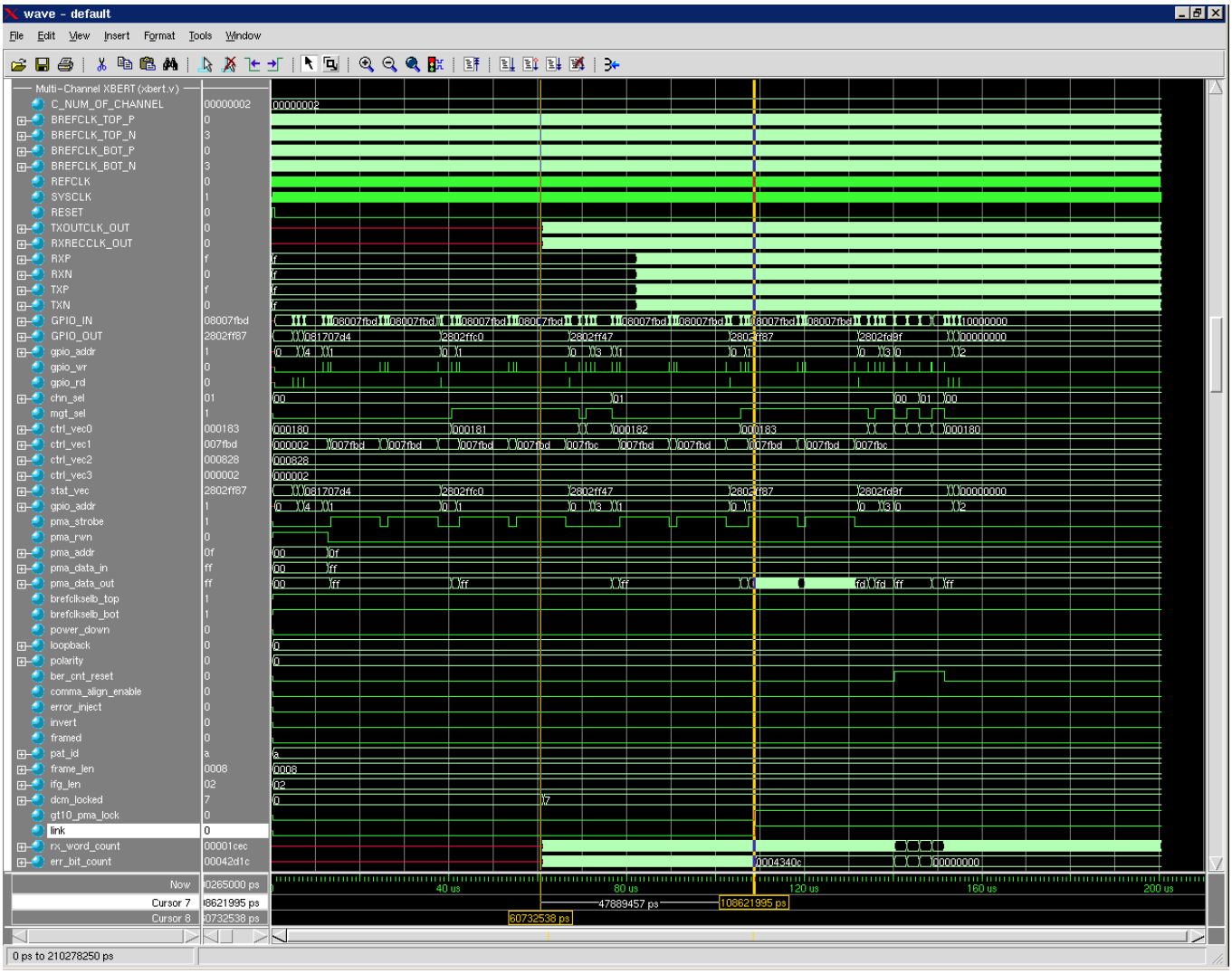
   ```
   > do ../../testbench/compile.do
   ```

10. In ModelSim, run the following command to run the behavioral simulation for 200 µs. The waveform window automatically pops up. The simulation takes about 15 minutes.

    ```
    > do ../../testbench/simulate.do
    ```

Figure 14 shows an example of the simulation result displayed on the waveform window. The user can expect to view the following behaviors (note that the MGT initialization time in future GT10 SmartModels may decrease, affecting the time of these events):

- At about 5 µs, the first GPIO transaction on the multi-channel XBERT module is observed on the bus.
- At about 14 µs, the first strobe on the PMA attribute programming bus is observed on the bus.
- At about 60 µs, TXOUTCLK and RXRECCLK on the MGT start toggling.
- At about 82 µs, TXP, TXN, RXP, and RXN on the MGT start toggling.
- At about 108 µs, PLL lock is achieved on the MGT.
- At about 110 µs, links are established on both channels.
- At about 140 µs, a strobe on the BER_CNT_RESET is observed and all counter values are reset. The ERR_BIT_COUNT value remains at 0 to indicate error-free links.



X762_14_092804

Figure 14: **ModelSim Behavioral Simulation**

Implementation in Xilinx Platform Studio (XPS)

The following steps provide the procedure for implementing the reference design using Xilinx Platform Studio (XPS):

1. Go to the root directory of the reference design.
2. Call the `config.bash` or `config.csh` script to set up the environment on Windows or UNIX.
3. Select and copy one of the MHS files (`system_2ch_1cpu.mhs` and `system_4ch_2cpu.mhs`) to `system.mhs`, depending on the number of processor cores available in the target FPGA device.
4. Modify `data/config.v` or copy one of the Verilog configuration files (`data/config_mk32x_*bit.v`) to `config.v`.
5. Modify `data/system.ucf` or copy one of the MK32x UCF files (`data/mk32x_*bit_*.ucf`) to `system.ucf`.
6. Modify other hardware and software configuration parameters listed in Table 12 and Table 13.
7. Launch Xilinx Platform Studio (XPS) GUI.

8. Select `File->Open Project-> Load system.xmp` project.
9. Select `Options->Project Options` and set the target device (architecture, device size, package and speed grade).
10. Select `Tools -> Update Bitstream`. This command compiles the software program, synthesizes the reference design, implements the design, generates the bitstream, and updates the bitstream with software instruction. The generated bitstream is located at `./implementation/download.bit`.

Demonstration on the MK32x Platform

The XBERT reference design includes demonstration bitstreams for Xilinx MK322 and MK325 platforms (referred to as the MK32x platforms). Refer to UG137: *RocketIO X BERT Reference Design User Guide* for instructions to set up and operate the demo on the MK32x platform.

Download Reference Design

The XBERT reference design can be downloaded from:
http://www.xilinx.com/bvdocs/appnotes/xapp762.zip.

Resource Utilization and Performance

Table 15 provides the resource utilization and speed performance of the XBERT reference design, measured using Xilinx ISE Tool 6.3i. Resource utilization includes the cost of the ChipScope Pro ILA core and all other optional features. Speed performance is measured on the longest data path in the multi-channel XBERT module, excluding paths in the ChipScope ILA core.

Table 15: **Resource Utilization and Speed Performance**

Design Configuration and Target Device	LUTs	Flip-Flops	Slices	BRAMs	BUFGs	DCMs	MGTs	Speed Performance at –6 Speed Grade (MHz)
2 channels, 40BIT, XC2VPX20	8799 (44%)	6768 (34%)	5809 (59%)	38 (43%)	10 (62%)	1 (12%)	4 (50%)	252.97
2 channels, 20BIT, XC2VPX20,	5904 (30%)	4819 (24%)	4147 (42%)					202.43
4 channels, 40BIT, XC2VPX70	17313 (26%)	13,091 (19%)	11572 (34%)	52 (16%)	14 (87%)		8 (40%)	252.65
4 channels, 20BIT, XC2VPX70	11525 (17%)	9201 (13%)	8084 (24%)					202.10

Conclusion

This application note describes a RocketIO X BERT reference design implemented in a Virtex-II Pro X FPGA. The reference design generates and verifies non-encoded, high-speed serial data on one or multiple point-to-point links (2.5 Gb/s to 10 Gb/s) between RocketIO X MGT ports. Such high-speed serial data are constructed using a pseudo-random bit sequence (PRBS) pattern, a clock pattern, or a user-defined pattern. The reference design builds a simple PPC system using the Xilinx EDK and Xilinx UltraController solution that can be easily modified or extended.

References

The following documents provide supplementary material useful with this application note:

1. Xilinx, Inc., UG137: *RocketIO X BERT Reference Design User Guide*, http://www.xilinx.com/bvdocs/userguides/ug137.pdf
2. Xilinx, Inc., XAPP661: "RocketIO Transceiver Bit-Error Rate Tester", http://www.xilinx.com/bvdocs/appnotes/xapp661.pdf
3. Xilinx, Inc., XAPP672: "The UltraController Solution: A Lightweight PowerPC Microcontroller", http://www.xilinx.com/bvdocs/appnotes/xapp672.pdf
4. Xilinx, Inc., UG035: *RocketIO X Transceiver User Guide*, http://www.xilinx.com/bvdocs/userguides/ug035.pdf
5. Xilinx, Inc., DS257: Linear Feedback Shift Register v3.0, http://www.xilinx.com/ipcenter/catalog/logicore/docs/lfsr.pdf
6. Xilinx, Inc., UG012: *Virtex-II Pro Platform FPGA User Guide*, http://www.xilinx.com/bvdocs/userguides/ug012.pdf
7. Xilinx, Inc., Embedded Development Kit, http://www.xilinx.com/edk
8. ITU-T Recommendation O.150, General Requirements for Instrumentation for Performance Measurements on Digital Transmission Equipment
9. IEEE Standard 802.3-2002, Part 3: Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications
10. IEEE Standard 802.3ae-2002, Amendment: Media Access Control (MAC) Parameters, Physical Layer, and Management Parameters for 10 Gb/s Operation

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
09/30/04	1.0	Initial Xilinx release.