XILINX®

XAPP713 (v1.1) April 18, 2007

Virtex-4 RocketIO Bit-Error Rate Tester

Author: Vinod Kumar Venkatavaradan

Summary

This application note describes the implementation of a Virtex™-4 RocketIO™ bit-error rate tester (XBERT) reference design. The XBERT reference design generates and verifies non-encoded or 8B/10B-encoded high-speed serial data on one or multiple point-to-point links between Virtex-4 RocketIO Multi-Gigabit Transceiver (MGT) ports embedded within a single Virtex-4 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 Dynamic Reconfiguration Port (DRP) on the Virtex-4 RocketIO™ MGT, which enables real-time control of PMA features, such as the TX output swing, TX pre-emphasis, and RX equalization. The software performs dynamic switching of the PMA clocking modes to support various data rates and brute-force scanning of the PMA parameters to find the best settings.

The embedded PPC405 processor transfers control and status to the XBERT module through the General-Purpose Input/Output (GPIO) interface. The UART interfaces to the processor through an OPB interface and enables a user interface through an external RS-232 serial port. XBERT finds application in performing either synchronous MGT tests through a backplane or asynchronous tests with an external device. The reference design is built using the Embedded Development Kit (EDK) [Ref 7], and it can be easily modified or extended.

Introduction

The hardware portion of the XBERT reference design includes a data plane and a control plane. Figure 1 shows the block diagram of the hardware 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 consists of two MGTs (MGTA and MGTB), which physically occupy one MGT tile in the Virtex-4 FPGA. Each MGT has its own pattern checker, but both MGTs in a channel share the same pattern generator. Each channel can load a different pattern. The MGT serial rate depends on the reference clock frequency and the internal PMA divider settings. The reference design can be scaled anywhere from one channel (two MGTs) to twelve channels (twenty-four MGTs).

The control plane of the reference design consists of the embedded PPC405 processor core, a DCR-to-OPB bridge, a data-side block RAM controller (DS-BRAM-CNTLR), 16 Kbyte data-side block RAMs (DS-BRAM), an instruction-side block RAM controller (IS-BRAM-CNTLR), and 64 Kbyte instruction-side block RAMs (IS-BRAM). The control plane also includes an OPB bus that connects the OPB UART Lite and the OPB GPIO cores. The processor reads the status and statistical values from the XBERT module and sends control to the XBERT module via the 32-bit GPIO. The UART provides an interactive user interface for the reference design. The ChipScope™ VIO is attached to the XBERT module to provide an alternative user interface.

The software portion of the reference design implements functions such as GPIO drivers, UART drivers, user menu display and input parsing, line rate calculation and timer function, XBERT control and statistics, MGT DRP data structure and read-modify-write routines, decision routine on PMA clocking mode settings, and brute-force scanning on PMA analog settings.

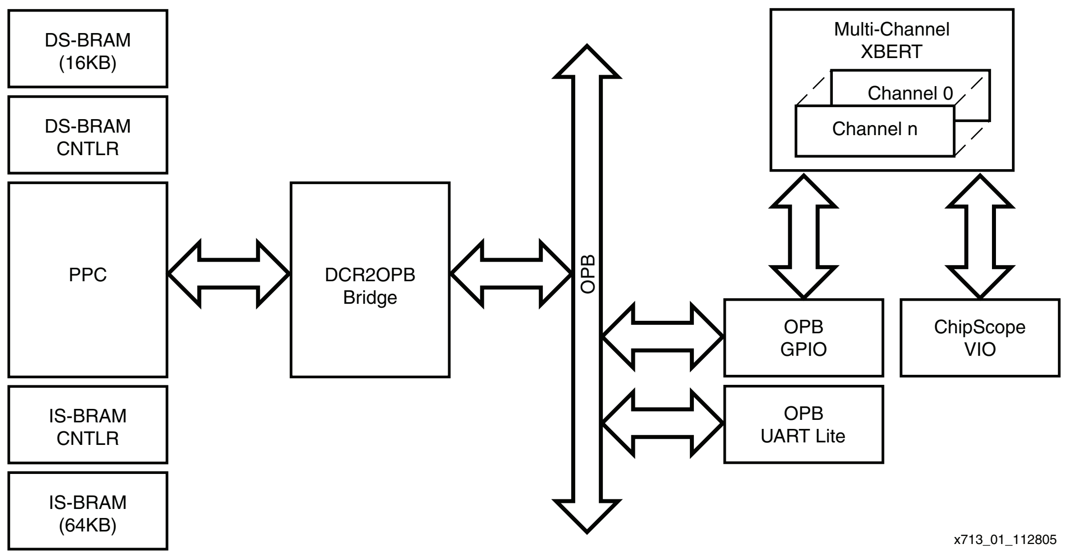


Figure 1: **Hardware Block Diagram of the XBERT Reference Design**

Features

The key features of the Virtex-4 XBERT reference design are summarized below:

- Modular design scales from 2 to 12 channels that enable up to 24 MGTs in Virtex-4 FPGA.
- Supports all devices except for XC4VFX12 in the Virtex-4 FX family.
- Supports any valid Virtex-4 MGT serial rate.
- Supports dynamic configuration of MGTs through the Dynamic Reconfiguration Port (DRP). Supports changing the serial rate and/or clocking mode on the fly.
- Supports 16, 20, 32, and 40-bit wide MGT fabric interface.
- Support streaming or framed transmission. Frame length and inter-frame gap is configurable.
- Supports 8B/10B encoded or non-encoded patterns, including eight ITU-T standard PRBS patterns (2^7-1, 2^9-1, $2^{11}-1$, $2^{15}-1$, $2^{20}-1$, $2^{23}-1$, $2^{29}-1$, $2^{31}-1$), a configurable user pattern, an IDLE pattern, and a counter pattern. Each channel can load a different pattern.
- Generates three clock patterns with frequency equal to $^1/_2$, $^1/_{10}$, or $^1/_{20}$ of the MGT serial rate.
- Supports BER test in both synchronous and asynchronous systems. Supports point-to-point BER test between any two MGTs on the same FPGA or on different FPGAs.
- MGTs in the left and right columns can operate fully asynchronously using different REFCLK sources.
- Supports MGT serial loopback mode. Implements a fabric loopback mode to reflect received data back to the transmission port through the FPGA fabric.
- Implements a self-synchronized pattern checker that automatically aligns and locks to an incoming pattern. Capable of testing with an external BER tester.
- Supports brute-force scanning for finding optimal PMA settings on each MGT.
- Supports 64-bit receive word and error counters for long test duration.
- Supports error injection on the transmit side to introduce a single bit error to the channel.
- Supports UART and ChipScope VIO user interfaces.
- Supports design resource reduction by removing unused patterns and/or features from the design.

Data Plane Description

MGT Use Model

To perform a successful bit-error rate (BER) test, some key features of the Virtex-4 MGT must be properly controlled. The XBERT reference design incorporates a subset of features provided by the Virtex-4 MGT, as listed in Table 1.

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

Layer	Virtex-4 MGT Features	Deployment in the Reference Design
PMA	SERDES	Yes
	Analog Clock and Data Recovery (CDR)	Yes and configurable
	Digital oversampled receiver	Yes and configurable
	Receiver signal detect, loss of signal indicator, and out-of-band (OOB) support at both receiver and transmitter	No
	Output swing adjustment and TX pre-emphasis	Yes and configurable
	RX continuous-time linear equalizer	Yes and configurable
	On-chip AC coupling	Yes and configurable
	Use of TXOUTCLK/RXRECCLK and internal clock dividers	Yes and configurable
	Serial loopback and repeater mode	Yes and configurable
PCS	TX FIFO and RX Elastic Buffer	Yes
	Comma Detection and Alignment	Yes and configurable
	8B/10B Encoder/Decoder	Yes and configurable
	64B/66B Encoder/Decoder, Scrambler, Gearbox, and Block Sync	No
	Clock Correction	No
	Channel Bonding	No
	Parallel loopback	No
	CRC generation and checking	No
	Dynamic Reconfiguration Port (DRP)	Yes

To avoid using clock correction schemes between local and remote ports, all Virtex-4 MGTs instantiated in the reference design use the recovered clock (RXRECCLK1) to clock in the FPGA fabric on the receive side. This makes the reference design capable of performing both asynchronous and synchronous BER tests, either between two Virtex-4 MGTs or between an MGT and external BER test equipment.

The reference design supports 16, 20, 32, and 40-bit MGT fabric interfacing, as well as 32-bit and 40-bit MGT internal data width. It can transmit and receive non-encoded data or 8B/10B encoded data, depending on the choice of the MGT fabric interface and internal data width. Table 2 lists all these options.

Table 2: **MGT Data Width and Encoding Options in XBERT Reference Design**

MGT Fabric Interface Width (XBERT Data Width)	MGT Internal Data Width	Encoding/ Decoding
16	32	None
16	40	8B/10B
20	40	None
32	32	None
32	40	8B/10B
40	40	None

The reference design produces standard PRBS patterns in the serial data stream. In order to transmit and receive raw PRBS patterns, the 8B/10B or 64B/66B coding on a Virtex-4 MGT should be bypassed. Such raw PRBS patterns are considered to be most stressful to the MGT on both run length and DC balance aspects. When 8B/10B is enabled, the run length of transmission pattern is limited to a maximum 5 bits. The 8B/10B encoder and decoder also require the use of comma detection and alignment in the MGT.

Depending on the target line rate, the reference design supports using the analog CDR or the digital oversampled receiver in the MGT.

MGT Clocking Scheme

Figure 2 shows the distribution of MGT reference clocks in Virtex-4 FPGA for a typical 4-channel XBERT implementation.

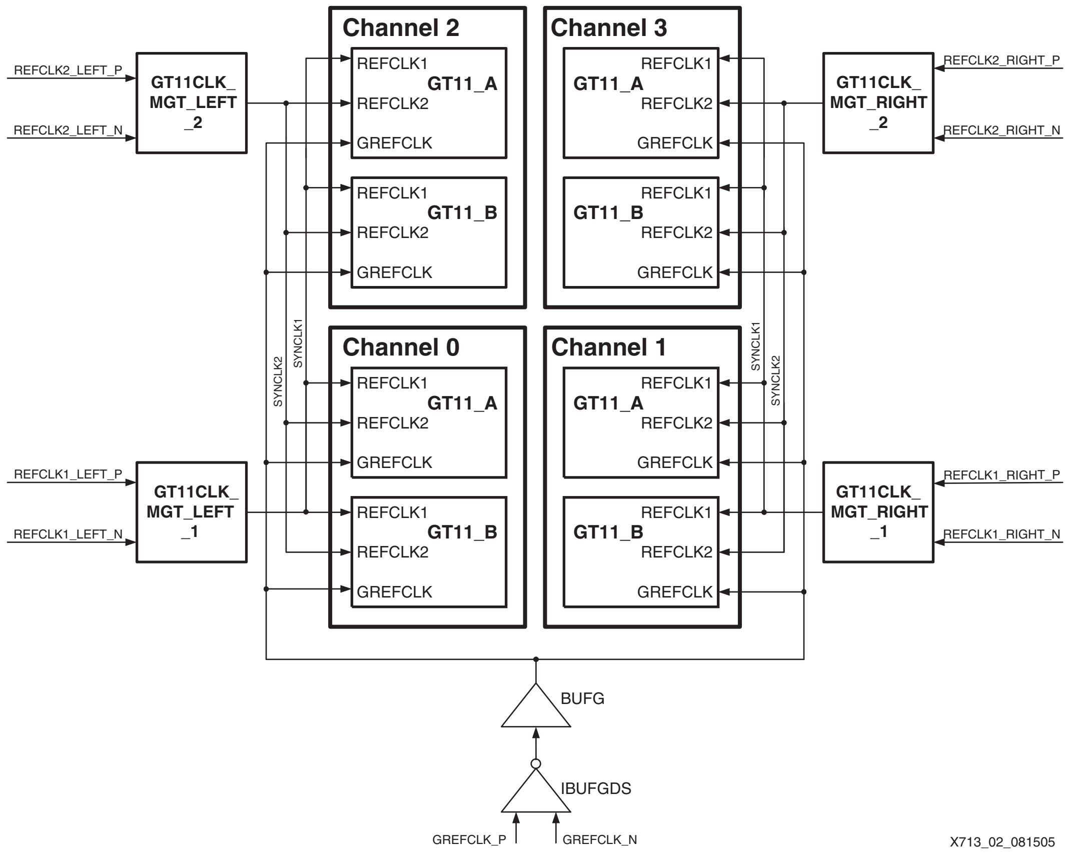


Figure 2: **MGT Reference Clock Distribution in XBERT Reference Design**

The MGT clock distribution in Virtex-4 supports the column-based structure. A column consists of multiple MGT tiles, which contain two MGTs each. Two low-jitter reference clock trees (SYNCLK1 and SYNCLK2) driven by the GT11CLK_MGT modules run the entire length of the column. In a tile, each PLL can select its own RX reference clock and a shared TX reference clock. There are three reference inputs to choose from: the two SYNCLKs in each column that drive the REFCLK1 and REFCLK2 inputs of the MGTs, and the global GREFCLK (for applications below 1 Gb/s).

There are two GT11CLK_MGT inputs bonded out per column. All four GT11CLK_MGT modules are instantiated in the XBERT reference design, named as GT11CLK_MGT_LEFT_1, GT11CLK_MGT_LEFT_2, GT11CLK_MGT_RIGHT_1, and GT11CLK_MGT_RIGHT_2. The following attributes are set on the GT11CLK_MGT_LEFT_1 and GT11CLK_MGT_RIGHT_1 modules so that they drive SYNCLK1 for the entire column:

```
SYNCLK1OUTEN = ENABLE
SYNCLK2OUTEN = DISABLE
```

Similarly, the following attributes are set on the GT11CLK_MGT_LEFT_2 and GT11CLK_MGT_RIGHT_2 modules so that they drive SYNCLK2 for the entire column:

```
SYNCLK1OUTEN = DISABLE
SYNCLK2OUTEN = ENABLE
```

The GREFCLK global clock is driven by a single IBUFGDS followed by a BUFG. The XBERT reference design provides a hardware configuration parameter for removing implementation of GREFCLK, thus saving the cost of this BUFG in the design.

Figure 3 illustrates the MGT user clock distribution in a typical four-channel XBERT design.

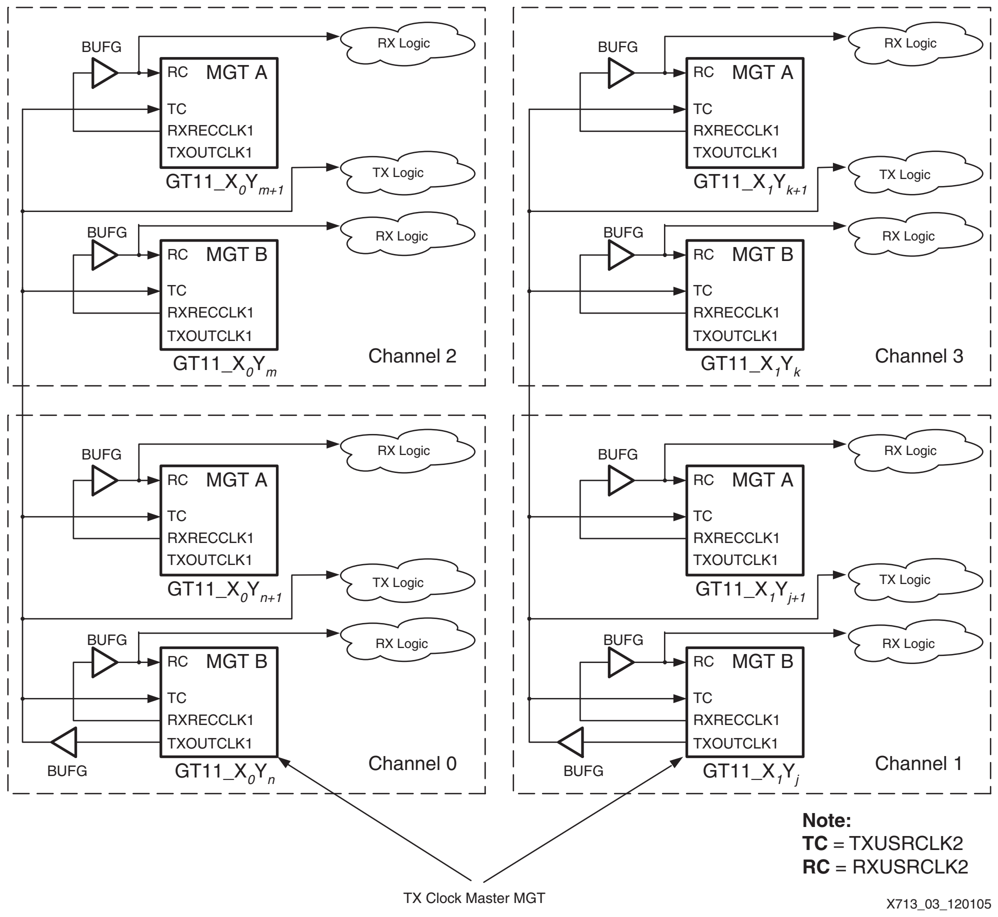


Figure 3: **MGT User Clock Distribution in XBERT Reference Design**

RXRECCLK1 from each MGT is fed to a global clock buffer (BUFG) and is used to clock the receiving-side logic and the RXUSRCLK2 port of the same MGT. This allows asynchronous operation between TX and RX on each MGT without the need of clock correction.

TXOUTCLK1 from one MGTB in each column (called the TX Clock Master MGT) is fed to a BUFG and is used to clock the transmitting-side logic and the TXUSRCLK2 port of all MGTs in the same column. This TX clocking scheme reduces the number of BUFGs required in the design. However, it also creates the following implications:

- The MGTs in one column must operate at a single line rate, since all TXUSRCLK must have the same frequency.
- Two columns of MGTs in the FPGA can operate at two different line rates fully independently.
- The two TX Clock Master MGTs must be activated and operate properly. In particular, the TX PLL of these MGTs must lock properly to the local reference clock. Otherwise, all MGTs in the entire column could be affected.

Hardware configuration parameters allow the XBERT reference design to support the following options:

- The location of the TX Clock Master MGT can be moved to an MGTB in any channel in the same column.
- A dedicated TXOUTCLK1 can be used for each channel. This results in the TX Clock Master MGT arrangement being discarded, and allows all the channels to operate independently of each other.
- The same clock used to generate TXUSRCLK2 can drive the RXUSRCLK2. This results in the use of RXRECCLK1 being discarded and allows TX and RX to operate fully synchronously.

To reduce the number of DCMs required in the system, the reference design does not use a DCM to generate RXUSRCLK and TXUSRCLK. Instead, it uses internal PCS clock dividers to attain the proper ratio between USRCLK and USRCLK2. Note that the MGT parallel loopback option becomes invalid when internal clock dividers are used in the design. Refer to UG076, *Virtex-4 RocketIO MGT User Guide* [Ref 2] for details regarding the use of internal PCS clock dividers.

PRBS Pattern Generation

Bit-error measurements are an 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.

For a given polynomial, there are two types of LFSR implementations that yields the equivalent result:

- Fibonacci (or Type-I) LFSR uses exclusive-OR (XOR) gates outside the shift register loop.
- Galois (or Type-II) LFSR uses XOR gates inside the shift register chain.

From a mathematical perspective, an implementation of multiple-stage LFSRs producing a PRBS pattern can be described by a polynomial. 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, with the result 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, but can switch to Galois LFSR using a compilation option.

Pattern Generator

The pattern generator in the XBERT reference design generates either 16, 20, 32, or 40-bit patterns to work with a 16, 20, 32, or 40-bit MGT fabric interface. Figure 4 shows the 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 16, 20, 32, or 40-bit data outputs (data_out).

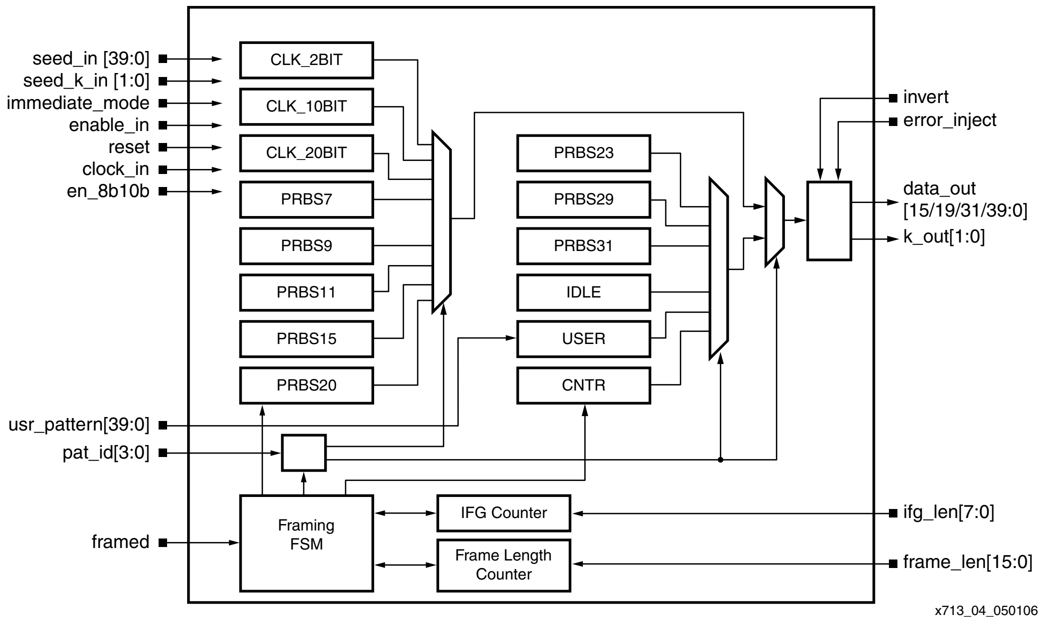


Figure 4: **Pattern Generator Block Diagram**

Table 3 summarizes all the patterns and their respective polynomials implemented in the XBERT reference design. Note that user can remove one or more patterns from the design for resource reduction.

Table 3 summaries the use model of each pattern supported in the XBERT reference design.

The pattern generator implements seven different polynomials (PRBS9,11,15, 20, 23, 29, and 31) as specified in ITU-T Recommendation O.150 for PRBS pattern generation [Ref 8]. 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 Virtex-4 RocketIO MGTs.

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

Pattern ID	Pattern Name	Pattern/Polynomial	Length of Sequence (bits)	Consecutive Zeros	Notes
0	1/2X Clock	`101010...`	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 [Ref 9].
1	1/10X or 1/8X Clock	5 ones, 5 zeros *or* 4 ones, 4 zeros	10 or 8	5 or 4	This pattern can be used as a clock pattern whose frequency is equal to $^1/_{10}$ or $^1/_8$ of the MGT serial speed, generating up to a 1.3 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 [Ref 9].
2	1/20X or 1/16X Clock	10 ones, 10 zeros *or* 8 ones, 8 zeros	20 or 16	10 or 8	This pattern can be used as a clock pattern whose frequency is equal to $^1/_{20}$ or $^1/_{16}$ of the MGT serial speed, generating up to a 645 MHz differential clock on the transceiver serial outputs.
3	$2^7 - 1$ PRBS	$x^7 + x^6 + 1$ (non-inverted signal)	$2^7 - 1$	7	Uses a proprietary polynomial that is not an ITU-T standard.
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 [Ref 8].
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 [Ref 8].
6	$2^{15} - 1$ PRBS	$x^{15} + x^{14} + 1$ (non-inverted signal)	$2^{15} - 1$	15	ITU-T Recommendation O.150, Section 5.3 [Ref 8]. 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 [Ref 8]. 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 [Ref 8]. 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 [Ref 8].
10	$2^{31} - 1$ PRBS	$x^{31} + x^{28} + 1$ (inverted signal)	$2^{31} - 1$	31	ITU-T Recommendation O.150, Section 5.8 [Ref 8]. This is a recommended PRBS test pattern for 10 Gigabit Ethernet. See IEEE Std 802.3ae-2002 [Ref 10].
11	Reserved				

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

Pattern ID	Pattern Name	Pattern/Polynomial	Length of Sequence (bits)	Consecutive Zeros	Notes
12	Idle Pattern	K28.5+, K28.5-, K28.5+, . . . (if 8B/10B bypassed and XBERT data width is 20 or 40 bits) *or* K28.5, D16.2, K28.5, D16.2, . . . (if 8B/10B enabled) *or* A1A1A1A1A2A21A2A2 (`0xF6F6F6F628282828`) ... (if 8B/10B bypassed and XBERT data width is 16 or 32 bits)	20 or 64	5	
13	User Pattern	Contains a repeated 16, 20, 32, or 40-bit configurable pattern. If 8B/10B is enabled, this pattern can be configured as a combination of a K-character and/or a data character. If 8B/10B is bypassed, the user can specify any 16, 20, 32, or 40-bit pattern, depending on the XBERT data width selected.	1 to 40	0 to 39	See Table 5 for a bit-mapping of the user pattern. *This pattern cannot be all zeros or all ones producing invalid data with an out-of-range run length.*
14	Reserved				

15	Counter Pattern	`0000000000,`
		`1111111111,`
		`. . .`
		(40-bit XBERT)
		or
		`00000000,`
		`11111111,`
		`. . .`
		(32-bit XBERT)
		or
		`00000,`
		`11111,`
		`. . .`
		(20-bit XBERT)
		or
		`0000,`
		`1111,`
		`. . .`
		(16-bit XBERT)
			Approximately from 320 (in 20-bit XBERT) to 640 (in 40-bit XBERT)	16 to 40	

Table 4: **Use Model of Supported Patterns in the XBERT Reference Design**

Pattern Type	Pattern ID	8B/10B	Framed Transmission	Comma Alignment
Clock patterns	0 to 2	Off	Off	Off
PRBS patterns	3 to 10	Off	Off	Off
		On	On	On
Idle pattern	12	Off	Off	On
		On		
User pattern	13	Off	Off	Off
		On	On	On
Counter pattern	15	Off	On	On
		On		

The idle pattern (IDLE) can be any one of the following patterns depending on the data width and encoding scheme selected. An idle pattern is composed of 2-byte or 8-byte idle word. The idle word is also used to fill-in inter-frame gap in framed transmission, which requires word alignment using the comma detection and alignment circuits inside the MGT.

- 8B/10B is bypassed. The XBERT data width is 20-bit or 40-bit. The idle pattern consists of interleaving K28.5+ (`0b0011111010`) and K28.5– (`0b1100000101`) symbols. The comma pattern is configured as the K28.5+ character in this case so that the MGT data output can be 2-byte aligned.
- 8B/10B is bypassed. The XBERT data width is 16-bit or 32-bit. The idle pattern consists of the A1A1A1A1A2A2A2A2 comma pattern (`0xF6F6F6F628282828`) typically used in SONET applications. The MGT data output can be 4-byte aligned.
- 8B/10B is enabled. The idle pattern consists of K28.5 and D16.2. The K28.5 character is configured as the MGT comma pattern. The MGT data output can be 2-byte aligned.

The counter pattern (CNTR) consists of 16, 20, 32, or 40-bit incremental counter words and idle words, providing a traceable and predictable test pattern that differs from the PRBS pattern. The design concatenates 4-bit counter values to build up the counter word so it repeats every 16 user clock cycles. 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 (USER) contains a repeated 16, 20, 32, or 40-bit configurable pattern. Table 5 lists the bit-mapping of the user pattern input (usr_pattern[39:0]) to the pattern generator.

Note: It is advisable to choose a user pattern different from the idle pattern if transmitting an 8B/10B-encoded user pattern. Otherwise, misalignment of the MGT data output could occur.

Table 5: **User Pattern Bit Mapping**

Bit Location usr_pattern[39:0]	Data Width 16 Bits		Data Width 32 Bits		Data Width 20 Bits	Data Width 40 Bits
	8B/10B ON	8B/10B OFF	8B/10B ON	8B/10B OFF	8B/10B OFF	8B/10B OFF
[7:0]	8-bit 8B/10B character transmitted second	8-bit pattern transmitted second	8-bit 8B/10B character transmitted last	8-bit pattern transmitted last	10-bit pattern transmitted second	10-bit pattern transmitted last
[8]	`1`: bit[7:0] is a K-character `0`: bit[7:0] is a data character	(Not Used)	`1`: bit[7:0] is a K-character `0`: bit[7:0] is a data character	(Not Used)		
[9]	(Not Used)		(Not Used)			
[17:10]	8-bit 8B/10B character transmitted first	8-bit pattern transmitted first	8-bit 8B/10B character transmitted third	8-bit pattern transmitted third	10-bit pattern transmitted first	10-bit pattern transmitted third
[18]	`1`: bit[17:0] is a K-character `0`: bit[17:0] is a data character	(Not Used)	`1`: bit[17:0] is a K-character `0`: bit[17:0] is a data character	(Not Used)		
[19]	(Not Used)		(Not Used)			
[27:20]			8-bit 8B/10B character transmitted second	8-bit pattern transmitted second	(Not Used)	10-bit pattern transmitted second
[28]			`1`: bit[27:0] is a K-character `0`: bit[27:0] is a data character	(Not Used)		
[29]			(Not Used)			
[37:30]			8-bit 8B/10B character transmitted first	8-bit pattern transmitted first		10-bit pattern transmitted first
[38]			`1`: bit[37:0] is a K-character `0`: bit[37:0] is a data character	(Not Used)		
[39]			(Not Used)			

The pattern generator contains a *finite state machine* (FSM) to delimit the output pattern by periodically inserting idle words. A word is either a 16, 20, 32, or 40-bit vector determined by the data width of the MGT fabric interface. Idle words — K28.5 +/– symbols, K28.5 and D16.2 characters, or A1A1A1A1A2A2A2A2 comma patterns — are used to fill *inter-frame gaps* (IFGs). The frame length and the IFG are configurable through the *frame length input* (frame_len) and the *IFG length input* (ifg_len) ports.

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 a direct-computed result 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 a single error into the output pattern through the *error injection control port* (error_inject). When this port is asserted High, bit 0 on the output is flipped to introduce a single bit error into the transmission. This function is implemented to self-check the integrity of the 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 the patterns back in order to link up with a non-standard external BER tester.

The PRBS $2^{7}-1$ pattern implemented in the reference design uses a proprietary polynomial because ITU-T does not provide a recommendation for this type of PRBS. The XBERT cannot link up with an external BER tester using this PRBS $2^{7}-1$ pattern unless it uses the same polynomial.

The pattern generator contains heavy combinatorial logic (for example, a 40-bit carry chain) for parallel PRBS pattern generation that must meet the 163 MHz timing target in order to support a 6.5 Gb/s data rate.

Pattern Checker

The pattern checker in the XBERT reference design can verify either 16, 20, 32, or 40-bit incoming data from a 16, 20, 32, or 40-bit MGT fabric interface. Figure 5 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.

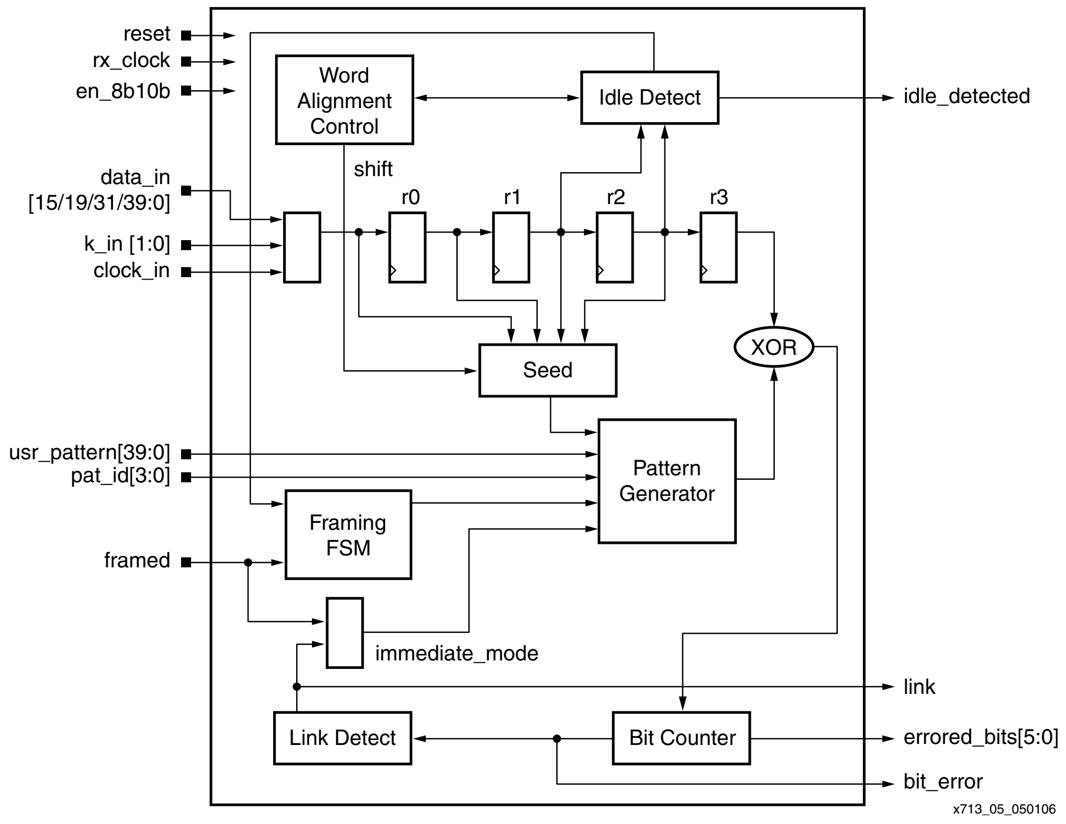


Figure 5: **Pattern Checker Block Diagram**

When the pattern checker receives framed data, it needs to recover the frame boundary by detecting the idle words within the incoming data IFG. Since the idle word is a 16, 20, 32, 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 framed pattern and holds the embedded pattern generator during 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 immediate mode when the link is down or reset is asserted. As illustrated in Figure 6, the pattern checker in immediate mode uses the incoming data to generate the seed value for the embedded pattern generator. The pattern generator uses the seed value to generate an output pattern in three clock cycles. This output pattern is expected to match the next incoming data.

In actual practice, several pipeline stages are introduced on the incoming data path to compensate for the three-cycle delays in the pattern generator and to meet an optimal timing target for 6.5 Gb/s operation. Therefore, the pattern checker can automatically adapt the internal state of the incoming data's polynomial 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. The process of watching for and declaring pattern match (or *link-up*) is performed by the link detection logic.

As soon as link-up is declared, the pattern checker switches to the self-advance mode. In this state, the embedded pattern generator starts to independently calculate and update the expected pattern on every clock cycle. 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 self-synchronization 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.

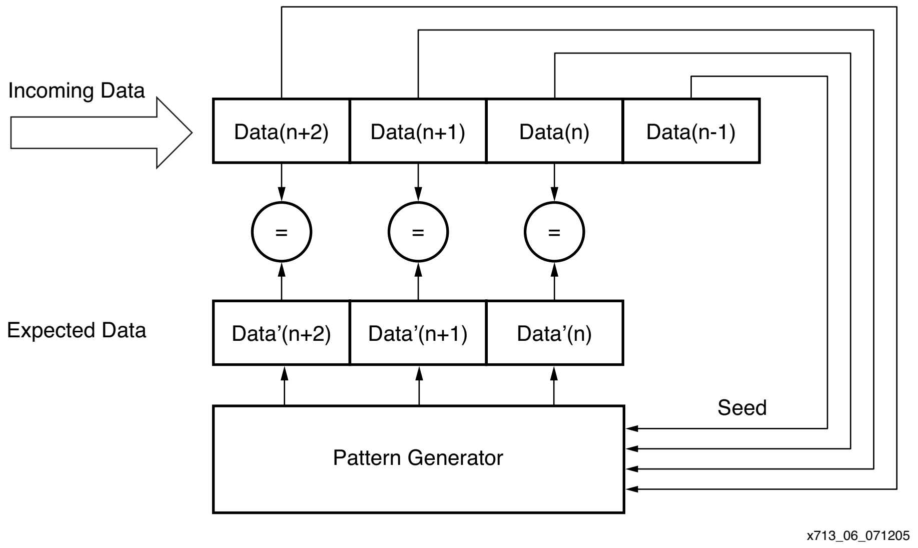


Figure 6: **Self Synchronized Pattern Checker**

Figure 7 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.

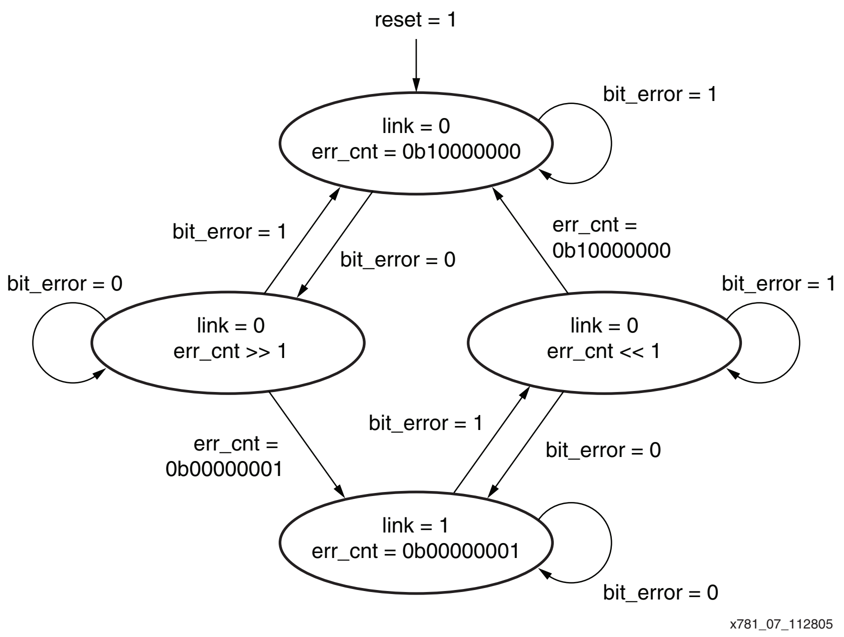


Figure 7: **Link Detection State Diagram**

Single-Channel XBERT Module

The single-channel XBERT is a self-contained module providing all necessary BER test functions, independent of the PPC405 subsystem. The single-channel XBERT module is a modular design that can be scaled to multiple channels in the FPGA fabric.

Figure 6 shows the 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:

- **An MGT tile containing two Virtex-4 RocketIO MGTs (MGTA and MGTB)**

  Both MGTs share the buffered TXUSRCLK2; therefore they must operate at the same TX data rate.

- **Two pattern checkers connecting to two MGTs**
- **One pattern generator shared by two MGTs**
- **One clock/reset module generating two independent RXUSRCLK2 for two MGTs**

  This module contains two global clock buffers (BUFGs) for buffering RXRECCLK1 from each MGT and generating RXUSRCLK2. It also generates clock status outputs (dcm_locked_rx) by detecting RXRECCLK1.

- **Two comma align control modules connecting to two MGTs**

  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 for two pattern checkers**

  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).

  The XBERT reference design uses the DSP48 block in the Virtex-4 FPGA as the 32-bit counter in order to improve timing performance and save fabric resources. The user can remove the use of DSP48 by asserting a compile option.

- **Two RX word counters for two MGTs**

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

- **Two shallow asynchronous FIFOs for two MGTs**

  These FIFOs provide the fabric loopback function that reflects the incoming data back to the transmission. Each FIFO 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. When fabric loopback is enabled, it requires that the incoming data be frequency-locked to the local reference clock, as in a synchronous operation.

- **Two Calibration Blocks for two MGTs**

  The calibration block is attached to each MGT to perform calibration and monitoring functions. The default calibration block used in this reference design is v1.4.1.

- **One ChipScope Pro™ ILA core**

  This core is clocked on the RXUSRCLK2 signal of either MGTA or MGTB. Data from both MGTs and associated internal logic are multiplexed by the mgt_sel input before being fed into the ILA core so that the selected MGT data can be displayed on ChipScope Pro Analyzer.

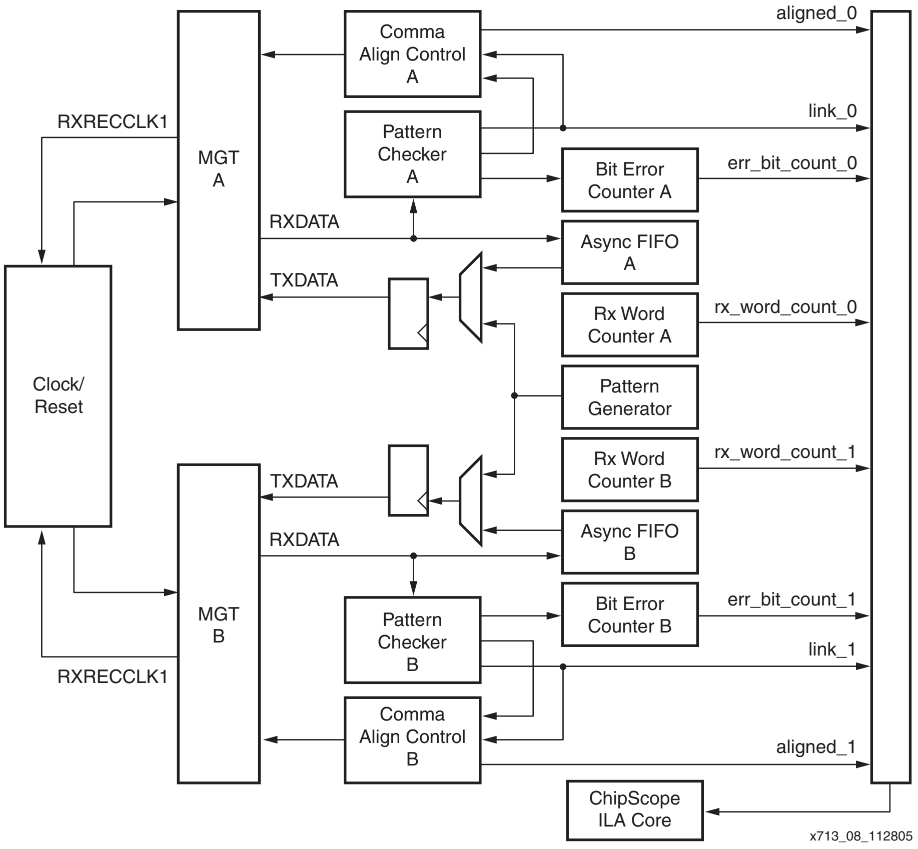


Figure 8: **Single-Channel XBERT Module**

Multi-Channel XBERT Module

The multi-channel XBERT module contains multiple instances (channels) of the single-channel XBERT module, grouped into left and right channels using a default placement configuration. Figure 9 is a block diagram of the multi-channel XBERT module. In this example, MGTs are placed at opposite sides of the FPGA, where even channels instantiate MGTs at the left side of the FPGA, and odd channels instantiate MGTs at the right side of the FPGA.

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.

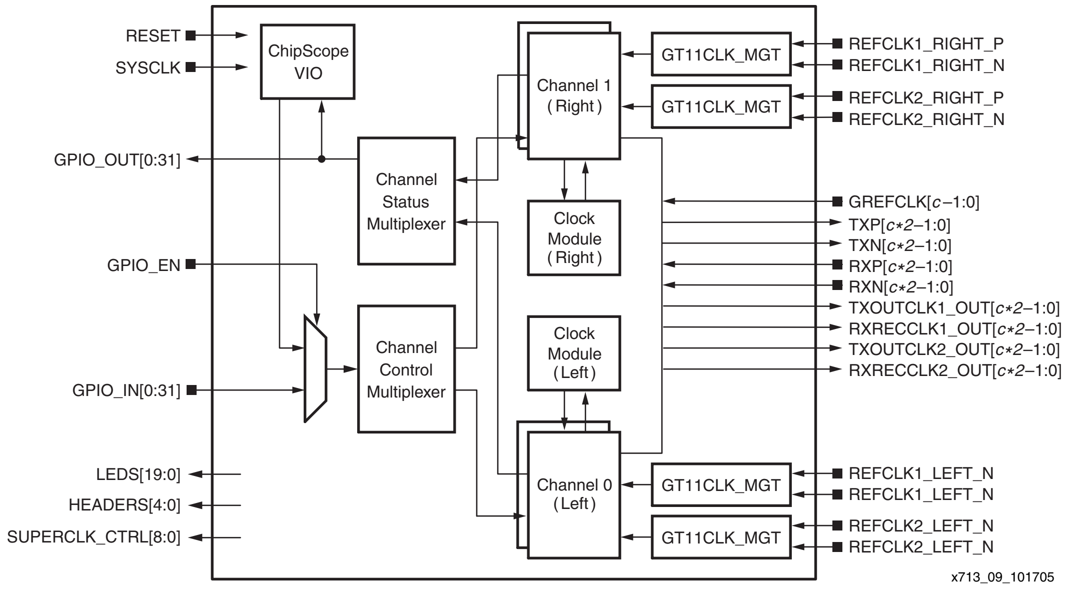


Figure 9: **Multi-Channel XBERT Module**

The user-accessible ports on the multi-channel XBERT module are defined in Table 6. 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 6: **Interface Signals of the Multi-Channel XBERT Module**

Name	Direction	Description
SYSCLK	Input	System Clock. Typically 50Mhz.
RESET	Input	System Reset.
GPIO_IN[0:31]	Input	32-bit GPIO input from the microprocessor, synchronous to SYSCLK.
GPIO_OUT[0:31]	Ouput	32-bit GPIO output to the microprocessor, synchronous to SYSCLK.
GPIO_EN	Input	GPIO enable signal.
REFCLK1_RIGHT_P, REFCLK1_RIGHT_N	Input	High-speed dedicated reference clock to the MGTs in the right column.
REFCLK2_RIGHT_P, REFCLK2_RIGHT_N	Input	Alternative high-speed dedicated reference clock to the MGTs in the right column.
REFCLK1_LEFT_P, REFCLK1_LEFT_N	Input	High-speed dedicated reference clock to the MGTs in the left column.
REFCLK2_LEFT_P, REFCLK2_LEFT_N	Input	Alternative high-speed dedicated reference clock to the MGTs in the left column.
GREFCLK[$c-1$:0][(1)]	Input	Global reference clock to the MGT tiles used for low speed applications(<1 Gb/s).
TXP[$c*2-1$:0], TXN[$c*2-1$:0]	Output	MGT differential ports that transmit serial data. Each Channel outpututs two bits as follows: Bit 0 is the output of MGTA in the channel. Bit 1 is the output of MGTB in the channel.

Table 6: **Interface Signals of the Multi-Channel XBERT Module *(Continued)***

Name	Direction	Description
RXP[$c*2-1$:0], RXN[$c*2-1$:0]	Input	MGT differential ports that receive serial data. Each Channel inputs two bits as follows: Bit 0 is the input of MGTA in the channel. Bit 1 is the input of MGTB in the channel.
TXOUTCLK1_OUT[$c*2-1$:0]	Output	Non-buffered TXOUTCLK1 outputs from every MGTs implemented in the design. Each channel outputs two bits as follows: Bit 0 is the output of MGTA in the channel, and Bit 1 is the output of MGTB in the channel.
RXRECCLK1_OUT[$c*2-1$:0]	Output	Non-buffered RXRECCLK1 outputs from every MGT implemented in the design. Each channel outputs two bits as follows: Bit 0 is the output of MGTA in the channel, and Bit 1 is the output of MGTB in the channel.
TXOUTCLK2_OUT[$c*2-1$:0]	Output	Non-buffered TXOUTCLK2 outputs from every MGTs implemented in the design. Each channel outputs two bits as follows: Bit 0 is the output of MGTA in the channel, and Bit 1 is the output of MGTB in the channel.
RXRECCLK2_OUT[$c*2-1$:0]	Output	Non-buffered RXRECCLK2 outputs from every MGT implemented in the design. Each channel outputs two bits as follows: Bit 0 is the output of MGTA in the channel, and Bit 1 is the output of MGTB in the channel.
LEDS[19:0]	Output	Bits 9 to 0 indicate the link status of all MGTs implemented on the left side. Bits 1 and 0 correspond to the MGTB and MGTA within channel 0. Bits 9 and 8 correspond to the MGTB and MGTA within channel 8. Bits 19 to 10 indicate the link status of all MGTs implemented on the right side. Bits 11 and 10 correspond to the MGTB and MGTA within channel 1. Bits 19 and 18 correspond to the MGTB and MGTA within channel 9.
HEADERS[4:0]	Output	Bit 0 is the SYSCLK. Bit 1 is the GREFCLK[0]. Bits 2 to 4 are unused.
SUPERCLK_CTRL[8:0]	Output	Optional control outputs to the SuperClock module used in Xilinx ML42x platform.

Notes:

1. *c* is the number of channels implemented in the XBERT reference design.

Table 7 and Table 8 list the bit definitions of the 32-bit GPIO input (GPIO_IN) and GPIO output (GPIO_OUT) registers, respectively.

Table 7: **Bit Definitions of the 32-bit GPIO Input (GPIO_IN[0:31])**

Bit	Vector 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]	N/A	Reserved.
[6:31]	CTRL	26-bit control vector transferred from the microprocessor to the data plane (the XBERT module). See Table 9 for bit definitions of this vector.

Table 8: **Bit Definitions of the 32-bit GPIO Output (GPIO_OUT[0:31])**

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

Table 9 defines the bits of the 26-bit control vector (CTRL) at various addresses. Table 10 defines the bits of the 32-bit status vector (STAT) at various addresses. Note that some of control and status vectors are dedicated to each MGT in a channel, while others are shared by both MGTs (MGTA and MGTB) in a channel.

Table 9: **Definitions of the 26-bit Control Vector (GPIO_IN[6:31])**

Address	Bit	Name	Description
0	[6:13]	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 a single bit error being injected on both MGTs in the selected channel.
	[17:18]	POLARITY	Sets the TX and/or RX polarity of the selected MGT within 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. `0b000`: Normal operation (no loopback) `0b001`: Reserved `0b010`: Reserved `0b011`: Activates serial loopback mode `0b100`: Activates the fabric loopback mode
	[22]	POWER_DOWN	When asserted High, deactivates fabric logic associated with the selected MGT, reducing overall power consumption.
	[23]	PMA_RESET	Resets the TX and RX PMA of the selected MGT within the selected channel.
	[24:25]	N/A	Reserved
	[26:30]	CHN_SEL	Selects the channel implemented in the multi-channel XBERT module, allowing reads/writes of the status/control vectors dedicated to this channel.
	[31]	MGT_SEL	Selects one of the MGTs (MGTA and MGTB) in the selected channel, allowing reads/writes of the status/control vectors dedicated to this MGT. `0`: Selects MGTA `1`: Selects MGTB

Table 9: **Definitions of the 26-bit Control Vector (GPIO_IN[6:31])** ***(Continued)***

Address	Bit	Name	Description
1	[6:21]	DI	Data to write to the DRP bus on the selected MGT in the selected channel.
	[22:29]	DADDR	Sets the address for DRP read or write on the selected MGT in the selected channel.
	30	DWE	Selects DRP write or read operation on the selected MGT in the selected channel. `0`: DRP read `1`: DRP write
	31	DEN	Rising edge of this signal enables DRP bus to perform a single read/write operation on the selected MGT in the selected channel.
2	[6:7]	N/A	Reserved
	[8:23]	FRAME_LEN	Sets the frame length of a delimited pattern in a framed transmission on both MGTs in the selected channel. The length of a frame begins from the first word following an inter-frame-gap (IFG) and ends at the last word prior to the next IFG. The length of a frame can be from one word to 65,535 words.
	24	N/A	Reserved
	25	EN_8B10B	Enable 8B/10B encoding/decoding on both MGTs in the selected channel.
	[26:29]	PAT_ID	Selects a pattern to generate on TX and verify on RX of both MGTs in the selected channel. Refer to Table 3, page 8 for a list of supported patterns
	30	FRAMED	Enable the framed transmission on both MGTs in the selected channel. This function delimits the output data with idle pattern.
	31	COMMA_ALIGN_ENABLE	Enable the comma alignment on both MGTs in the selected channel. This function requires the presence of idle pattern in the received data.

Table 9: **Definitions of the 26-bit Control Vector (GPIO_IN[6:31])** ***(Continued)***

Address	Bit	Name	Description
3	[6:15]	SUPERCLK_CTRL	Control signals to the SuperClock module attached to Xilinx ML42x platform. Bit 6: Three-state control: `0`: All signals to SuperClock module are high-Z `1`: All signals to SuperClock module are enabled Bit 7: MR Bit 8, 9: SEL1, SEL0 Bit 10, 11, 12: M2, M1, M0 Bit 13, 14, 15: N2, N1, N0
	[16:19]	N/A	Reserved
	[20]	CS_MGT_SEL	Selects one of the MGTs (MGTA and MGTB) to connect to ChipScope ILA. `0`: Select MGTA `1`: Select MGTB
	[21:23]	CB_CTRL	Control signals to the calibration block on both MGTs in a selected channel. Bit 23: Disable the calibration block Bit 22: Reserved Bit 21: Reset he calibration block
	[24:31]	IFG_LEN	Sets the length of the inter-frame gap (IFG) in a framed transmission on 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	[6:11]	N/A	Reserved
	[12:31]	USR_PATTERN_LOWER	Specify the lower 20 bits of the 40-bit user pattern on both MGTs in the selected channel
5	[6:7]	N/A	Reserved
	[8:31]	CB_CTRL_EXT	Extended calibration block control signals.
6	[6:11]	N/A	Reserved
	[12:31]	USR_PATTERN_UPPER	Specify the upper 20 bits of the 40-bit user pattern on both MGTs in the selected channel

Table 10: **Bit Definitions of the 32-bit Status Vector (GPIO_OUT(0:31])**

Address	Bit	Name	Description
0	[0:7]	BITS_PER_WORD	Indicates the XBERT data width as the number of bits in a word that transmits or receives on the multi-channel XBERT module. Each word can be 16, 20, 32, or 40 bits corresponding to the data width of the MGT fabric interface.
	[8:10]	N/A	Reserved
	11	GPIO_EN	Indicates the level of GPIO_EN signal on the multi-channel XBERT module. `1`: The GPIO interface is enabled `0`: The GPIO interface is disabled If ChipScope VIO core is implemented in the design, it can only be enabled when the GPIO interface is disabled.
	[12:15]	NUM_OF_CHN	Indicates the total number of channels implemented in the multi- channel XBERT module.
	[16:24]	N/A	Reserved
	25	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
	26	LINK	Indicates the link status on 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
	27	PMA_LOCK_TX	Indicates the PMA TX lock status on the selected MGT in the selected channel. `1`: The PMA TX has achieved lock `0`: The PMA TX has not achieved lock
	28	PMA_LOCK_RX	Indicates the PMA RX lock status on the selected MGT in the selected channel. `1`: The PMA RX has achieved lock `0`: The PMA RX has not achieved lock
	29	DCM_LOCKED_TX	Indicates the status of the MGT TX user clocks in the selected channel. `1`: The clock is detected and stable `0`: The clock is not detected or unstable
	[30:31]	DCM_LOCKED_RX	Indicates the status of the MGT RX user clocks in the selected channel. Bit 30 is for MGTB. Bit 31 is for MGTA. Each bit can be: `1`: The clock is detected and stable `0`: The clock is not detected or unstable

Table 10: **Bit Definitions of the 32-bit Status Vector (GPIO_OUT(0:31))** ***(Continued)***

Address	Bit	Name	Description
1	[0:31]	RX_WORD_COUNT	Indicates the total number of received words on the selected MGT in the selected channel. Each word is either 16,20,32 or 40 bits. This counter starts at the completion of a system reset or a counter reset. The output is a 32-bit value that 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 in the selected channel. This counter starts at the completion of a system reset or a counter reset. The output is a 32-bit value that wraps around when it exceeds 4,294,967,295. This output is only valid when the link is up on the associated MGT.
3	[0:7]	DEVICE_TYPE	Indicates the type of target FPGA device used in the implementation. The decimal value on this output corresponds to the device type. For example, 8'd60 indicates the V4FX60 device, 8'd100 indicates the V4FX100 device.
	[8:15]	VERSION1	Provides the first digit of the XBERT reference design version number.
	[16:23]	VERSION2	Provides the second digit of the XBERT reference design version number.
	[24:31]	CALBLK_VER	Provides the calibration block version number.
4	[0:7]	BUILT_MON	Provides the month when implements this design and generates the bitstream.
	[8:15]	BUILT_DAY	Provides the day when implements this design and generates the bitstream.
	[16:31]	BUILT_YEAR	Provides the year when implements this design and generates the bitstream.
5	[0:15]	DOUT	This is the DRP data output from the selected MGT in the selected channel.
	[16:23]	MGT_B_LOC	Indicates the placement location of MGTB in the selected channel. The decimal number on this output corresponds to the CHN_*_MGT_B value defined in the design configuration file (`config.v`).
	[24:31]	MGT_A_LOC	Indicates the placement location of MGTA in the selected channel. The decimal number on this output corresponds to the CHN_*_MGT_A value defined in the design configuration file (`config.v`).

Table 10: **Bit Definitions of the 32-bit Status Vector (GPIO_OUT(0:31))** ***(Continued)***

Address	Bit	Name	Description
6	[0]	DRDY	This is the DRP data ready output from the selected MGT in the selected channel. This signal is generated as a "sticky bit" for each DRP access. It can only be cleared by the DEN signal on the DRP port asserted for the subsequent DRP access.
	[1:4]	N/A	Reserved
	[5:8]	TXOUTCLK_ MASTER_RIGHT	Indicates the channel number of the TX master channel in the right column.
	[9:12]	TXOUTCLK_ MASTER_LEFT	Indicates the channel number of the TX master channel in the left column.
	[13:15]	OTHER_CONFIG	Bit 13 indicates the GREFCLK configuration option. `1`: Use of GREFCLK is implemented `0`: GREFCLK option is removed Bit 12 indicates the TXOUTCLK configuration option. `1`: Each channel is using its own TXOUTCLK `0`: All channels in the same column share the TXOUTCLK from the TX clock master MGT Bit 11 indicates the fabric loopback configuration option. `1`: Fabric loopback is implemented `0`: Fabric loopback is not implemented
	[16:31]	PATTERN_CONFIG	Indicates the pattern configuration option. Bit 31 corresponds to 1st pattern shown in Table 3, bit 30 corresponds to 2nd pattern in Table 3, etc. High on each bit indicates the corresponding pattern is implemented in the design. Low indicates the corresponding pattern is removed from the design.

GPIO Operation

As shown in Figure 10, 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 on the GPIO_IN bus. The GPIO_IN data is then held for at 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 can be updated only when address 0 is selected during a GPIO write.

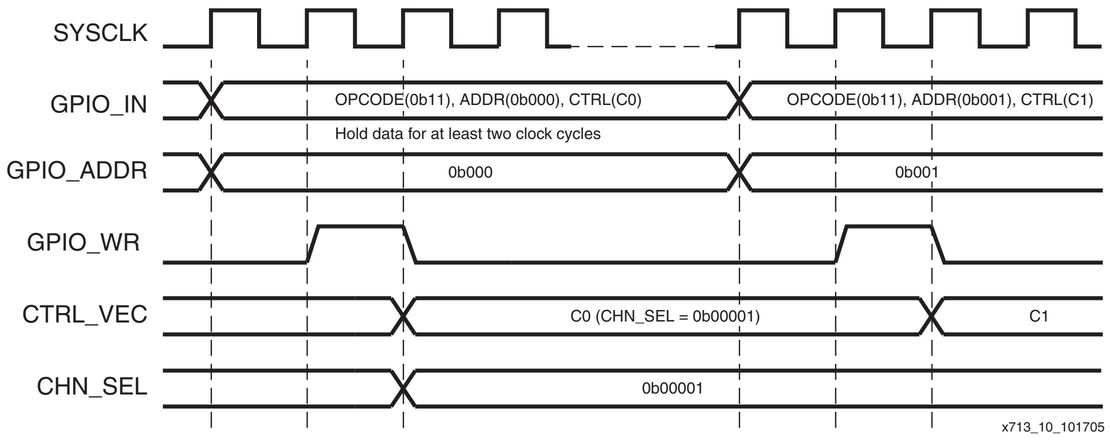
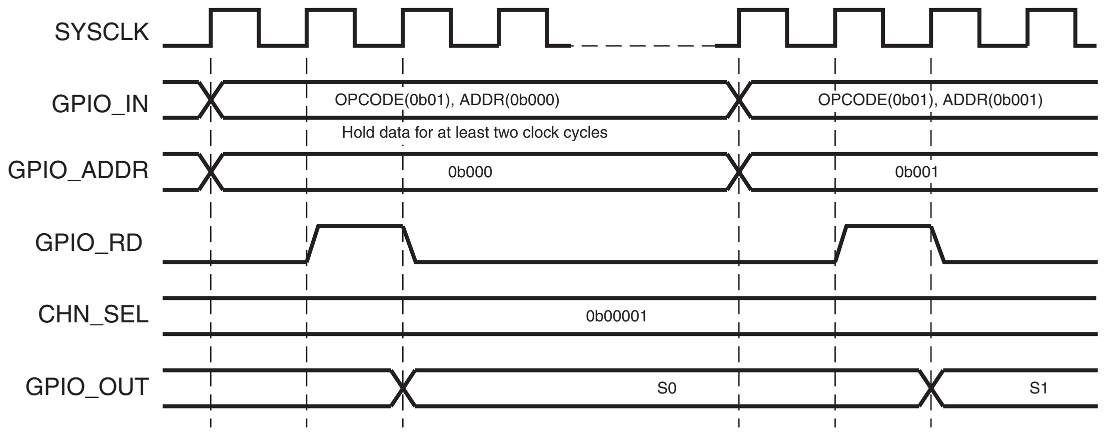

SYSCLK
GPIO_IN
OPCODE(0b11), ADDR(0b000), CTRL(C0)
OPCODE(0b11), ADDR(0b001), CTRL(C1)
Hold data for at least two clock cycles
GPIO_ADDR
0b000
0b001
GPIO_WR
CTRL_VEC
C0 (CHN_SEL = 0b00001)
C1
CHN_SEL
0b00001
x713_10_101705

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

As shown in Figure 11, to read a status vector from the multi-channel XBERT module, the user sets the operation code (OPCODE = `0b01`) and the address (ADDR) fields on the GPIO_IN bus. The GPIO_IN data is then held for at least two clock cycles. The status vector at the 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 at address 0, then initiates a GPIO read transaction. Data on the GPIO_OUT bus stays until the next GPIO read occurs.


SYSCLK
GPIO_IN
OPCODE(0b01), ADDR(0b000)
OPCODE(0b01), ADDR(0b001)
Hold data for at least two clock cycles
GPIO_ADDR
0b000
0b001
GPIO_RD
CHN_SEL
0b00001
GPIO_OUT
S0
S1

Note: CHN_SEL or MGT_SEL is set by the latest GPIO write at address 0.

x713_11_101705

Figure 11: **Timing Diagram of GPIO Read on the Multi-Channel XBERT Module**

Control Plane Description

PowerPC OCM and BRAM

The on-chip memory (OCM) interface on the PowerPC 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 control plane in the XBERT reference design contains a data-side block RAM controller (DS-BRAM-CNTLR) and an instruction-side block RAM controller (IS-BRAM-CNTLR). Each BRAM controller serves as a dedicated interface between the block RAMs (BRAMs) 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 (IS-BRAM), which is configured into 64 Kbyte memory in the reference design. The D-Side controller provides an interface to the 32-bit data-side block RAM (DS-BRAM), which is configured into 16 KByte memory. Although the reference design supports memory depth expansion, the maximum amount of memory addressable by the instruction-side BRAM controller and data-side BRAM controller is 128 KBytes and 64 KBytes, respectively.

The instruction-side and data-side OCM interfaces operate at a 1:1 ratio of the processor clock. This clock is generated from the FX output of the DCM module, as shown in Figure 12. To enable the 1:1 clock ratio on OCM interfaces, the user must set ISCNTLVALUE and DSCNTLVALUE to `0x81`.

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 12. The PPC405 core and the other modules in the control plane (such as the I-Side controller, the D-Side controller, and the GPIO interface) run at the same frequency, twice the frequency, or one-half 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, which is triggered by an external active high reset input (sys_rst). Both DCM_MODULE and PROC_SYS_RESET blocks are standard components provided in the EDK.

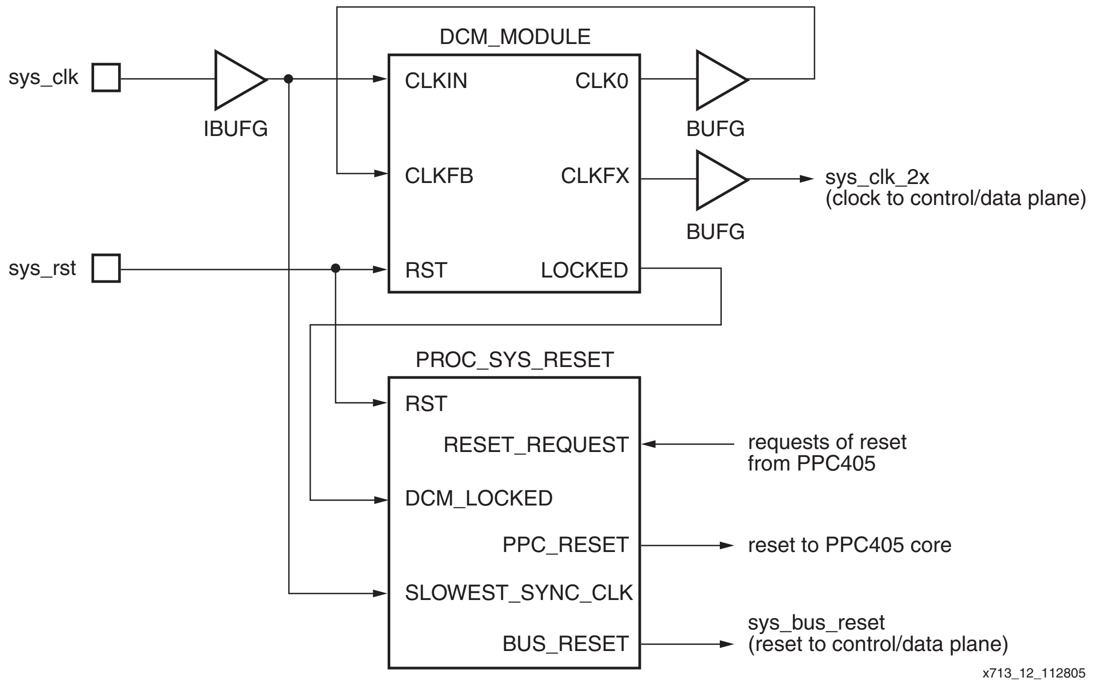


Figure 12: **Clock and Reset Distribution in the Control Plane**

DCR2OPB Bridge

The DCR-to-OPB bridge (DCR2OPB) translates DCR transactions to OPB transactions. It functions as a slave on the DCR side and a master on the OPB side. This allows utilizing OPB devices, such us OPB UART Lite and OPB GPIO, with DRP-type access. The DCR2OPB bridge eliminates the PLB and PLB-to-OPB bridge, thereby freeing the high-speed PLB bus for high-performance devices.

The DCR base address (C_DCR_BASEADDR) and high address C_DCR_HIGHADDR) specify the address range on the DCR that translate to OPB transactions. The OPB offset address (C_OFFSET) specifies the start address on the OPB that corresponds to the DCR base address. Since the OPB address is a byte address and the DCR address is a 4-byte word address, one DCR data word maps to four OPB data bytes, and the OPB address range should be four times the DCR address range.

The DCR2OPB bridge provided in the XBERT reference design is a pcore that comes with source codes in both Verilog and VHDL. Refer to Table 15 for a list of configuration parameters for this core.

OPB GPIO

The OPB general purpose input/output (OPB GPIO) is a 32-bit peripheral that attaches to the OPB bus. The GPIO input/output ports connect to the multi-channel XBERT module. This OPB GPIO core is a standard EDK core. Refer to the *Processor IP Reference Guide* in EDK documentation [Ref 7] for details on this core. Refer to Table 15 for a list of configuration parameters for this core.

OPB UART Lite

The OPB UART Lite core is a standard EDK core that attaches to the OPB bus. 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 15 for a list of configuration parameters for this core.

Software Description

Address Map

The software instruction and data are loaded into the instruction-side BRAM and data-side BRAM connected to the instruction-side and data-side OCM interfaces of the PPC405 processor. The UART and GPIO are memory mapped to the OPB and bridged to the DCR on the processor. The multi-channel XBERT module is connected to the GPIO interface, which in turn is connected to the processor through the DCR2OPB bridge. Table 11 lists the software address map of the memory and DCR/OPB devices in the XBERT reference design.

These addresses are defined in the `system.mhs` file and used in some C codes (`sw/gpio.c, sw/uart.c`).

Table 11: **Address Map of Memory and DCR/OPB Devices in XBERT Reference Design**

Device	Register	Address Type	Address Boundaries		Size
			Upper	Lower (Base Addr)	
Instr-Side BRAM		Memory	`0xFFFFFFFF`	`0xFFFF0000`	64 Kbytes
Data-Side BRAM		Memory	`0xFE003FFF`	`0xFE000000`	16 Kbytes
OPB GPIO		OPB	`0xA00003FF`	`0xA0000200`	512 bytes
	Data Register	OPB	`0xA0000203`	`0xA0000200`	4 bytes
	3-state Register	OPB	`0xA0000207`	`0xA0000204`	4 bytes
OPB UART Lite		OPB	`0xA00000FF`	`0xA0000000`	256 bytes
	Receive FIFO Reg.	OPB	`0xA0000003`	`0xA0000000`	4 bytes
	Transmit FIFO Reg.	OPB	`0xA0000007`	`0xA0000004`	4 bytes
	Status Register	OPB	`0xA000000B`	`0xA0000008`	4 bytes
	Control Register	OPB	`0xA000000F`	`0xA000000C`	4 bytes
DCR2OPB Bridge		DCR	`0xFF`	`0x00`	1024 bytes
		OPB	`0xA00003FF`	`0xA0000000`	1024 bytes

Data Structures

There are two main data structures defined in the header file (`sw/xbert.h`):

- **XBERT_ENTRY:** Stores general XBERT configuration data, such as the number of channels, device type, XBERT version number, etc. The software only needs one instance of this data structure.
- **CHN_ENTRY:** Stores channel-dependent XBERT configuration and status data, such as loopback and power-down control, link status, bit error counts, etc. The software instantiates multiple instances of this data structure for each XBERT channel.

There are several DRP data arrays instantiated in the main application code (`sw/xbert.c`):

- **clk_attr_str** and **clk_attr_offset:** Contain data-rate-dependent PMA settings such as PMA clock divider settings and PMA analog settings. These settings are given as DRP addresses, masks, offsets, and DRP values. Whenever the user changes the MGT clock source or changes the target line rate, these settings should be modified properly through the MGT DRP. Among these settings, some are dedicated to each MGT and can be accessed via the DRP of each MGT. Others have effect on both MGTs and can be accessed only through the DRP of one MGT (A or B) in each MGT tile. There is a cross-reference array (clk_attr_cross_index) that links the names of each setting with the associated DRP entries. Table 12 lists these rate dependent PMA settings with their DRP addresses.

Table 12: **Data Rate Dependent PMA Settings and DRP Addresses**

PMA Attribute/ Setting Scope	PMA Attribute/Setting Name	DRP Address(1)	DRP Bit Mask(2)
• Dedicated to each MGT. • Accessible through the DRP of each MGT.	TXOUTCLK1_USE_SYNC	A/B: 0x5B	0x4000
	RXRECCLK1_USE_SYNC	A/B: 0x5B	0x8000
	TXCLK0_FORCE_PMACLK	A/B: 0x5B	0x2000
	RXCLK0_FORCE_PMACLK	A/B: 0x5B	0x1000
	TX_CLOCK_DIVIDER[1:0]	A/B: 0x5B	0x0C00
	RX_CLOCK_DIVIDER[1:0]	A/B: 0x5B	0x0300
	ENABLE_DCDR	A/B: 0x53	0x0200
	SAMPLE_8X	A/B: 0x53	0x0100
	RXBY_32	A/B: 0x53	0x0800
	DIGRX_FWDCLK[1:0]	A/B: 0x7C	0x0003
	DIGRX_SYNC_MODE	A/B: 0x7C	0x0004
	RXUSRDIVISOR[4:0]	A/B: 0x53	0x001F
	RXFDET_HYS_SEL[2:0]	A/B: 0x60	0x01C0
	RXFDET_LCK_SEL[2:0]	A/B: 0x60	0x0038
	TXFDET_HYS_SEL[2:0]	A/B: 0x70	0x01C0
	TXFDET_LCK_SEL[2:0]	A/B: 0x70	0x0038
	RXFDET_HYS_CAL[2:0]	A/B: 0x60	0x7000
	RXFDET_LCK_CAL[2:0]	A/B: 0x60	0x0E00
	TXFDET_HYS_CAL[2:0]	A/B: 0x70	0x7000
	TXFDET_LCK_CAL[2:0]	A/B: 0x70	0x0E00
	DCDR_FILTER[2:0]	A/B: 0x53	0x0060
	RXVCODAC_INIT[9:0]	A/B: 0x68	0x7FE0
	VCODAC_INIT[9:0]	A/B: 0x78	0x7FE0
	TXCLKMODE[3:0]	A:0x5C B:0x5E	0x000F
	RXCLKMODE[5:0]	A:0x5E B:0x7A	0x007E
	RXASYNCDIVIDE[1:0]	A:0x5E B:0x7A	0x0180
	RXDIGRX	A:0x7D B:0x59	0x0002
	RXDIGRESET	A:0x46 B:0x62	0x8000
	RXPLLNDIVSEL[3:0]	A:0x7D B:0x59	0x0F00
	RXOUTDIV2SEL_0[3:0]	A:0x7D B:0x59	0xF000
	RXOUTDIV2SEL_1[3:0]	A:0x6D B:0x49	0x7800
	TXASYNCDIVIDE[1]	A:0x54 B:0x56	0x0020
	TXASYNCDIVIDE[0]	A:0x4C B:0x4E	0x2000
	RXCPSEL	A:0x75 B:0x51	0x0001
	RXLOOPFILT[3:0]	A:0x7D B:0x59	0x003C
	RXRCPADJ[2:0]	A:0x56 B:0x72	0x3800
	TXSLEWRATE	A:0x54 B:0x56	0x0010
	RXCMFPD	A:0x65 B:0x41	0x8000
	RXRCPPD	A:0x46 B:0x62	0x0008
	RXCMADJ	A:0x4E B:0x6A	0xC000

Table 12: **Data Rate Dependent PMA Settings and DRP Addresses *(Continued)***

PMA Attribute/ Setting Scope	PMA Attribute/Setting Name	DRP Address(1)	DRP Bit Mask(2)
• Common to both MGTs in an MGT tile. • Accessible through the DRP of one MGT in an MGT tile.	TXPLLNDIVSEL[3:0]	`A:0x7A`	`0x0F00`
	TXOUTDIV2SEL_0[3:0]	`A:0x7A`	`0xF000`
	TXOUTDIV2SEL_1[3:0]	`A:0x6A`	`0x7800`
	TXCPSEL	`A:0x72`	`0x0001`
	TXLOOPFILT[3:0]	`A:0x7A`	`0x003C`
	TXABPMACLKSEL[1:0]	`B:0x5D`	`0x0300`
	RXAPMACLKSEL[1:0]	`B:0x5D`	`0x3000`
	RXBPMACLKSEL[1:0]	`B:0x5D`	`0x0C00`

Notes:

1. **A** and **B** represents MGTA and MGTB.
2. **Bit Mask** represents the valid bits within the 16-bit DRP register for the given attribute/setting.

- **pma_attr_str** and **pma_attr_offset:** Contain configurable PMA settings such as RX equalization, TX pre-emphasis and CDR sampling phase. These settings are given as DRP addresses, masks, offsets and DRP values. The software can perform brute force scanning of all possible values on these settings. The setting that results in a lowest BER on the tested MGT is deemed to be the optimal setting for the given test condition. Multiple entries of this array can be combined together in the brute force scan so their combinatorial effect on the MGT can be evaluated. There is a cross-reference array (pma_attr_cross_index) that links the names of each setting with the associated DRP entries. Table 13 lists these PMA settings with their DRP addresses.

Table 13: **Configurable/Scannable PMA Settings and DRP Addresses**

PMA Attribute/Setting Name	DRP Address(1)	DRP Bit Mask(2)
RXLKADJ[4:0]	`A:0x4E B:0x6A`	`0x001F`
RXSELDACFIX[4]	`A:0x56 B:0x72`	`0x4000`
RXSELDACFIX[3:0]	`A:0x46 B:0x62`	`0x7800`
RXSELDACTRAN[4:0]	`A:0x46 B:0x62`	`0x07C0`
RXAFEEQ[2:0]	`A:0x56 B:0x72`	`0x0007`
RXEQ[62] (DFE_POWER_DOWN)(3)	`A:0x5F B:0x5C`	`0x4000`
RXEQ[61:56] (DFE_CTRL)(3)	`A:0x5F B:0x5C`	`0x3F00`
RXEQ[55:48] (DFE_TAP_C1_MSB)(3)	`A:0x5F B:0x5C`	`0x00FF`
RXEQ[47:40] (DFE_TAP_C1_LSB)(3)	`A:0x57 B:0x54`	`0xFF00`
RXEQ[39:32] (DFE_TAP_C2)(3)	`A:0x57 B:0x54`	`0x00FF`
RXEQ[31:24] (DFE_TAP_C3)(3)	`A:0x4F B:0x4C`	`0xFF00`
RXEQ[23:16] (DFE_TAP_C4)(3)	`A:0x4F B:0x4C`	`0x00FF`
RXEQ[15:8] (DFE_TAP_C5)(3)	`A:0x47 B:0x44`	`0xFF00`
RXEQ[7:0] (DFE_TAP_C6)(3)	`A:0x47 B:0x44`	`0x00FF`
RXDCCOUPLE	`A:0x4E B:0x6A`	`0x0040`
TXDAT_PRDRV_DAC[2]	`A:0x54 B:0x56`	`0x0001`
TXDAT_PRDRV_DAC[1:0]	`A:0x4C B:0x4E`	`0xC000`
TXDAT_TAP_DAC[4:0]	`A:0x4C B:0x4E`	`0x001F`
TXPRE_TAP_PD	`A:0x5C B:0x5E`	`0x0080`
TXPRE_PRDRV_DAC[2:0]	`A:0x5C B:0x5E`	`0x0700`

Table 13: **Configurable/Scannable PMA Settings and DRP Addresses**

PMA Attribute/Setting Name	DRP Address(1)	DRP Bit Mask(2)
TXPRE_TAP_DAC[4:3]	`A:0x5C B:0x5E`	`0x0060`
TXPRE_TAP_DAC[2:0]	`A:0x54 B:0x56`	`0xE000`
TXPOST_TAP_PD	`A:0x4C B:0x4E`	`0x1000`
TXPOST_PRDRV_DAC[2:0]	`A:0x54 B:0x56`	`0x000E`
TXPOST_TAP_DAC[4:0]	`A:0x4C B:0x4E`	`0x0F80`

Notes:

1. **A** and **B** represents MGTA and MGTB.
2. **Bit Mask** represents the valid bits within the 16-bit DRP register for the given attribute/setting.
3. Although DFE controls are provided, the DFE feature is not supported in the Virtex-4 MGT.

- **drp_debug_attr:** Contains a list of DRP entries that are loaded during DRP batch writes at XBERT initialization. This array is used to overwrite MGT default settings from the software, and mainly used for MGT debug purpose. This array can also be used to load customized MGT settings tailored for user applications.
- **pd_attr_offset:** Contains a list of DRP entries each powering down some portions of the MGT circuit. This array is used to generate a complete power-down function for the MGT. Among these settings, some are dedicated to each MGT and can be accessed via the DRP of each MGT. Others have effect on both MGTs and can be accessed only through the DRP of one MGT (A or B) in each MGT tile.
- **comma_attr_offset:** Contains all COMMA-related PCS settings. The software needs to change these settings when switching between 10-bit comma alignment and 32-bit SONET A1/A2 alignment. Table 15 lists these COMMA-related PCS settings and their DRP addresses.

Table 14: **COMMA Settings and DRP Addresses**

PMA Attribute/Setting Name	DRP Address(1)	DRP Bit Mask(2)
COMMA32	`A/B:0x42`	`0x0080`
ALIGN_COMMA_WORD[1:0]	`A/B:0x42`	`0x0003`
COMMA_10B_MASK[9:0]	`A/B:0x52`	`0x03FF`
MCOMMA_32B_VALUE[15:0]	`A/B:0x61`	`0xFFFF`
MCOMMA_32B_VALUE[31:16]	`A/B:0x69`	`0xFFFF`
PCOMMA_32B_VALUE[15:0]	`A/B:0x71`	`0xFFFF`
PCOMMA_32B_VALUE[31:16]	`A/B:0x79`	`0xFFFF`

Notes:

1. **A** and **B** represents MGTA and MGTB.
2. **Bit Mask** represents the valid bits within the 16-bit DRP register for the given attribute/setting.

GPIO Driver

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

- **Level 0:** provides reads and writes to the 32-bit GPIO interface through DCR mtdcr and mfdcr functions.
- **Level 1:** generates the OPCODE and ADDR, transfers the control and status from software data structures (XBERT_ENTRY and CHN_ENTRY) to the GPIO interface.
- **Level 2:** provides higher-level functions such as channel setup, channel query, channel control, and PMA/DRP query.

Since the RX_WORD_COUNT and ERR_BIT_COUNT are only 32-bit outputs from the multi-channel XBERT module, these counters might roll over during a long BER test. The software resolves this issue by detecting counter carries and aggregating them into 64-bit wide counters.

UART Driver

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

Level 0: provides UART initialization and read/write functions through DCR mtdcr and mfdcr functions.

Level 1: provides higher level functions such as formatted printing, HEX number printing and number input. It also provide cursor control functions using ANSI/VT100 terminal control sequences, such as clear entire screen, reset the cursor position and clear a line on the screen.

Main Application

The main application code (`sw/xbert.c` and `sw/xbert.h`) implements the navigation of the XBERT software menu and the interaction with user inputs. Figure 13 is the flow diagram of the main application code. Refer to UG242, *Virtex-4 RocketIO Bit-Error Rate Tester User Guide* [Ref 1] for details regarding the usage of the XBERT software menu.

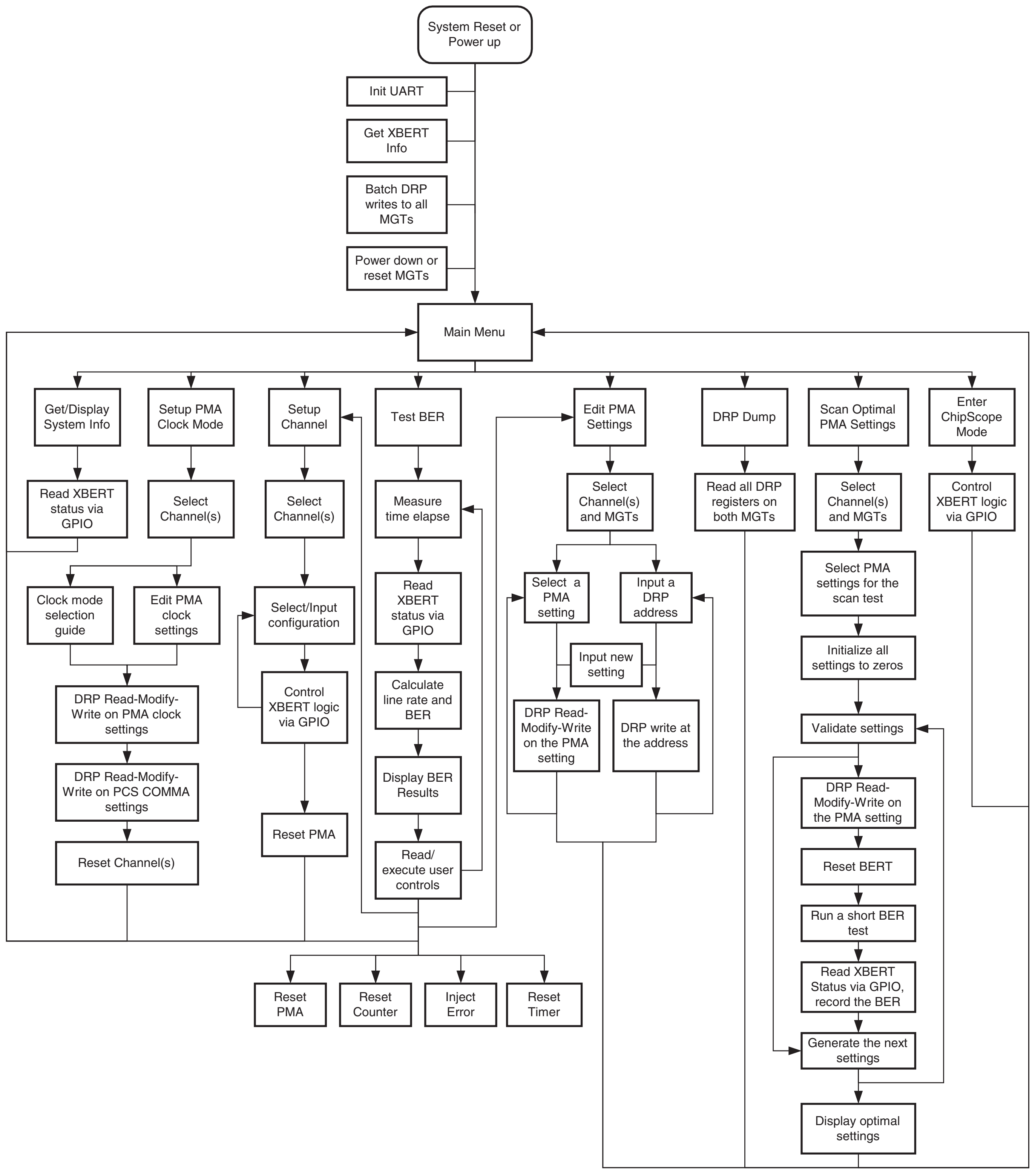

System Reset or Power up
Init UART
Get XBERT Info
Batch DRP writes to all MGTs
Power down or reset MGTs
Main Menu
Get/Display System Info
Setup PMA Clock Mode
Setup Channel
Test BER
Edit PMA Settings
DRP Dump
Scan Optimal PMA Settings
Enter ChipScope Mode
Read XBERT status via GPIO
Select Channel(s)
Select Channel(s)
Measure time elapse
Select Channel(s) and MGTs
Read all DRP registers on both MGTs
Select Channel(s) and MGTs
Control XBERT logic via GPIO
Clock mode selection guide
Edit PMA clock settings
Select/Input configuration
Read XBERT status via GPIO
Select a PMA setting
Input a DRP address
Select PMA settings for the scan test
Initialize all settings to zeros
Input new setting
DRP Read-Modify-Write on PMA clock settings
Control XBERT logic via GPIO
Calculate line rate and BER
DRP Read-Modify-Write on the PMA setting
DRP write at the address
Validate settings
DRP Read-Modify-Write on PCS COMMA settings
Display BER Results
DRP Read-Modify-Write on the PMA setting
Reset PMA
Reset Channel(s)
Read/ execute user controls
Reset BERT
Run a short BER test
Reset PMA
Reset Counter
Inject Error
Reset Timer
Read XBERT Status via GPIO, record the BER
Generate the next settings
Display optimal settings
x713_13_101705

Figure 13: **Flow Diagram of Main Application Code**

Some important C functions implemented in this main application code are described below.

- **Read-Modify-Write on the DRP (pma_attr_rmw):** This function has the inputs of DRP address, bit offset, bit mask, target MGT number, target channel number, and the value to write to the DRP register. It first reads the 16-bit value on the DRP register, modifies only the masked bits on this register, and then writes back to the register. Furthermore, it reads back the DRP register and checks if the masked bits are written properly.

- **PMA reset (mgt_pma_reset):** This function resets the MGT TX and RX PLL, pattern generator, and checker in the BERT module on the selected MGT in a selected channel.
- **BERT reset (bert_reset):** This function first resets the PMA, and then resets the pattern checker and BER counters on the selected MGT in a selected channel.
- **Channel reset (channel_reset):** This function executes the BERT reset on a selected channel. It also resets and configures the MGT calibration blocks.
- **BER counter reset (reset_counter):** This function only resets the pattern check and BER counters on the selected MGT in a selected channel. It does not affect the MGT PLL.
- **Batch DRP write (drp_batch_write):** This function loads the default DRP settings from the drp_debug_attr array to all MGTs. This process creates customized MGT settings by overriding the default settings in the ISE software.
- **PMA clock settings selection (pma_clk_attr_select):** This function guides the user to select proper PMA clock settings based on target line rate, reference clock preference, VCO rate, coding scheme, etc. The outputs of this function are stored in the clk_attr_offset array.
- **Get XBERT info (get_xbert_info):** This function reads basic XBERT hardware information through the GPIO. This includes the number of MGTs available in the device, the number of enabled MGTs, the number of channels implemented in the design, the placement of these channels in the design, etc.
- **ChipScope setup (chipscope_setup):** This function sets the selection between MGTA and B in each channel, so that the data multiplexer on the ChipScope ILA core in the channel selects signals from one of these MGTs.
- **Channel setup (channel_setup):** This function sets up channel parameters such as power down, loopback, polarity, pattern inversion, pattern selection, etc.
- **Scan optimal PMA settings (scan_optimal_pma_attr):** In this function, the user combines up to 64 bits of different PMA settings into a test vector. A brute-force scan of all possible test vector values is then performed, and a short BER test is run for each test value. The settings that yield the best BER result are recorded as the optimal settings. The brute-force scan can be accelerated by skipping a subsequent test value if the current test value does not generate a link on the BERT, and as a result, the BER test finishes more quickly. Optionally, the duration of each short BER test can be user-programmed.
- **DRP edit (pma_edit):** This function provides a list of PMA settings and allows read-modify-write on the selected setting. It can also takes user inputs (such as DRP address, target MGT, and value) and write them directly to the DRP.
- **DRP dump (drp_dump):** This function reads and displays all DRP register values on both MGTs in the selected channel.
- **BER test (bert_test):** This function performs the BER testing on all channels implemented in the design. It continuously polls the XBERT status, calculates the line rate and BER results, and displays the BERT test console.

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 five-minute BER test at 10 Gb/s receives three trillion bits. Such a test ensures the BER is less than 10^{-12} with 95% accuracy, if no errors are observed. A longer thirty-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* [Ref 3] 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 Word Count and Bit Error Count) using the formula below. The software assumes that the next received bit contains an error. Therefore, the BER can never be equal to 0 but should decrease the longer BER test lasts (that is, when RX Word Count 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.

$$\text{Bit Error Rate (BER)} = \frac{\text{Number of Bit Errors} + 1}{(\text{RX Word Count} \times \text{Number of Bits per Word}) + 1}$$

Time Function

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 50 MHz):

```
BEGIN PROCESSOR
  PARAMETER DRIVER_NAME = cpu_ppc405
  PARAMETER DRIVER_VER = 1.00.a
  PARAMETER HW_INSTANCE = ppc405_1
  PARAMETER CORE_CLOCK_FREQ_HZ = 50000000
  PARAMETER COMPILER = powerpc-eabi-gcc
  PARAMETER ARCHIVER = powerpc-eabi-ar
END
```

Hence, If the processor frequency changes then appropriate changes should be made in the CORE_CLOCK_FREQ_HZ parameter to reflect these changes.

DRP Access

The XBERT reference design provides access to the Dynamic Reconfiguration Port (DRP) on the MGT so that the user can edit any PMA and/or PCS setting of the MGT. *This feature is for advanced users only.* Any edit should be done only after a thorough understanding of the DRP register being edited. Refer to Appendix C in UG076, *Virtex-4 Rocket IO Transceiver User Guide* [Ref 2] for a description of the DRP registers.

Design Configuration

The XBERT reference design provides hardware and software configuration parameters that allow the user to customize the design for any of the following purposes:

- Tailor feature sets based on the demand in the target system
- Reduce the resource cost by removing some of the features
- Port the design from one V4FX device/system to another
- Extend the design by adding additional FPGA logic

The XBERT reference design provides a PERL script, design_config.pl, to help the user makes changes to these configuration parameters if the target board is one of Xilinx ML42x platforms. This script modified all necessary design files to make appropriate configurations.

Hardware Configuration Parameters

Changing the hardware configuration parameters involves editing one or more of the following files:

- The XMP file: `system.xmp`
- 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 15 lists the hardware configuration parameters and the location of these parameters.

Table 15: **Hardware Configuration Parameters**

Module	Parameter Description	Acceptable or Typical Settings	Where and What to Change
XBERT	Target FPGA device	xc4vfx20, xc4vfx40, xc4vfx60, xc4vfx100, xc4vfx140	1. Change this parameter in the XMP file or through the XPS GUI. 2. Change C_DEVICE_TYPE parameter on the XBERT module in the MHS file.
	Number of channels implemented in the design	`2 to 12`	1. Change C_NUM_OF_CHANNEL parameter on the XBERT module in the MHS file. 2. Change the port width of TXP, TXN, RXP, RXN, TXOUTCLK1_OUT, TXOUTCLK2_OUT, RXRECCLK1_OUT, and RXRECCLK2_OUT to match the number of channels. 3. Modify the UCF file to match the number of channels.
	MGT placement	GT11_X0Y0 to GT11_X0Y11 GT11_X1Y0 to GT11_X1Y11	1. Modify the UCF file in the MGT placement section. 2. Modify the MGT placement numbers defined by CHN_*_MGT_* parameters in the Verilog configuration file to match the UCF file. These parameters are required to properly distribute clocks between left and right columns, and provide the MGT placement data to the software.
	XBERT hardware version number	`8'd0` to `8'd255`	Change C_VERSION_1 and C_VERSION_2 parameters on the XBERT module in the MHS file. These make the first two digits of the XBERT version number as the hardware version.
	XBERT data width	`16, 20, 32, or 40`	Change C_DATA_WIDTH on the XBERT module in the MHS file to specify the XBERT data width and MGT fabric interface width.
	Clock frequency of SYSCLK input	8 to 100	Change the C_SYSCLK_PERIOD_NS parameter on the XBERT module in the MHS file to define the SYSCLK clock period in nanoseconds. SYSCLK is used to clock the GPIO interface and MGT DRP. The maximum frequency of this clock is 125 MHz.
	Source of TXUSRCLK2	MASTER, DEDICATED, or NONE	Change the C_TXOUTCLK_MODE parameter on the XBERT module in the MHS file to select one of the following TXUSRCLK2 clocking options: • MASTER: Use TXOUTCLK1 from the TX clock master MGT to generate per-column TXUSRCLK2. • DEDICATED: Use TXOUTCLK1 from each channel to generate per-channel TXUSRCLK2. • NONE: Use MGT reference clock (MGTCLK) to generate per-column TXUSRCLK2.
	The MGT location of the TX clock master	0 to C_NUM_OF_CHANNEL – 1	Change one of the following parameters on the XBERT module in the MHS file to set the location of TX clock master in the left or right column, respectively: • C_TXOUTCLK_MASTER_LEFT • C_TXOUTCLK_MASTER_RIGHT
	The use of ChipScope ILA and VIO core	0 or 1	Changing C_USE_ILA_VIO parameter on the XBERT module in the MHS file inserts or remove the ChipScope ILA and VIO cores in the reference design. Furthermore, use C_ILA_CHN_A and C_ILA_CHN_B to define two channels that attach to ChipScope ILA.
	Removal of fabric loopback	0 or 1	Change C_USE_FABRIC_LOOPBACK parameter on the XBERT module in the MHS file to implement or remove the fabric loopback function and its associated FPGA logic in order to reduce resource cost.

Table 15: **Hardware Configuration Parameters *(Continued)***

Module	Parameter Description	Acceptable or Typical Settings	Where and What to Change
XBERT (cont'd)	Use of GREFCLK	ENABLE or DISABLE	Changing the C_GREFCLK_MODE parameter on the XBERT module in the MHS file can implement or remove the use of GREFCLK as alternative MGT reference clock. Removal of GREFCLK can save BUFG resource.
	Source of RXUSRCLK2	DEDICATED or NONE	Change the C_RXRECCLK_MODE parameter on the XBERT module in the MHS file to select one of the following RXUSRCLK2 clocking options: • DEDICATED: Use RXRECCLK1 from each MGT to generate per MGT RXUSRCLK2. • NONE: Use TXUSRCLK2 to generate RXUSRCLK2.
	Removal of individual pattern	`0b1011010101001100`	Change the C_PATTERN_CONFIG parameter on the XBERT module in the MHS file to implement or remove specified patterns from the design to reduce resource cost and improve performance. Each bit of this parameter corresponds to a pattern. The LSB bit indicates the 1st pattern listed in Table 3, page 8.
PPC405	Processor frequency in Hz	`50000000`	1. Change CORE_CLOCK_FREQ_HZ on the processor module in the MSS file. 2. Modify the UCF file to change the timing constraints on the processor clock (sys_clk_2x).
	Stack size of the processor	`1024`	Change this parameter in the XMP file or through the XPS GUI.
	Heap size of the processor	`4`	Change this parameter in the XMP file or through the XPS GUI.
OCM BRAMs	The base and high addresses of the instruction-side BRAM	`0xFFFF0000`, `0xFFFFFFFF`	1. Change C_BASEADDR and C_HIGHADDR on the isbram_if_cntlr module in the MHS file. 2. Change the origin and length of isocm defined in the linker script to match the base address and the high address defined in the MHS file.
	The base and high addresses of the data-side BRAM	`0xFE000000`, `0xFE003FFF`	1. Change C_BASEADDR and C_HIGHADDR on the dsbram_if_cntlr module in the MHS file. 2. Change the origin and length of dsocm defined in the linker script to match the base address and the high address defined in the MHS file.
DCR2OPB	The base and high addresses of the DCR2OPB bridge	`0x000`, `0x0FF`	Change C_DCR_BASEADDR, C_DCR_HIGHADDR on the dcr2opb_bridge module in the MHS file.
	The offset of OPB bus addresses mapped to DCR	`0xA0000000`	Change C_OFFSET on the dcr2opb_bridge module in the MHS file.
OPB GPIO	The base and high addresses of the OPB GPIO module	`0xA0000200`, `0xA00003FF`	Change C_BASEADDR, C_HIGHADDR on the opb_gpio module in the MHS file.

Table 15: **Hardware Configuration Parameters *(Continued)***

Module	Parameter Description	Acceptable or Typical Settings	Where and What to Change
OPB UART Lite	The base and high addresses of OPB UART Lite	`0xA0000000, 0xA00000FF`	Change C_BASEADDR and C_HIGHADDR on the opb_uartlite module in the MHS file.
	The number of data bits in the serial frame of UART	`8`	Change C_DATA_BITS on the opb_uartlite module in the MHS file.
	Clock frequency of the OPB bus driving the OPB UART Lite module in Hz.	`50000000`	Change C_CLK_FREQ on the opb_uartlite module in the MHS file.
	Baud rate for the OPB UART Lite in bits per second	`38400`	Change C_BAUDRATE on the opb_uartlite module in the MHS file.
	The use of data parity of the OPB UART Lite module	`0 or 1`	Change C_USE_PARITY on the opb_uartlite module in the MHS file to enable or disable parity for the OPB UART Lite module.
	Parity type of the OPB UART Lite module	`0 or 1`	Change C_ODD_PARITY on the opb_uartlite module in the MHS file.

Software Configuration Parameters

Table 16 below lists the software configuration parameters and the location of these parameters.

Table 16: **Software Configuration Parameters**

Module	Parameter Description	Acceptable or Typical settings	Where and What to change
XBERT	The software version	`1 to 10`	Change SW_VERSION in `sw/xbert.c`. This makes the third digit of the XBERT version number the software version.
OPB GPIO	The DCR address of the 32-bit GPIO data register.	`128`	Change GPIO_DATA_REG in `sw/gpio.c`.
	The DCR address of the 32-bit GPIO tri-state register.	`129`	Change GPIO_TRI_REG in `sw/gpio.c`.
OPB UART Lite	The DCR address of the receive FIFO register of the OPB UART Lite.	`0`	Change RFIFO in `sw/uart.c`.
	The DCR address of the transmit FIFO register of the OPB UART Lite.	`1`	Change TFIFO in `sw/uart.c`.
	The DCR address of the status register of the OPB UART Lite.	`2`	Change STA in `sw/uart.c`.
	The DCR address of the cotrol register of the OPB UART Lite.	`3`	Change CTR in `sw/uart.c`

Reference Design

The XBERT reference design can be downloaded from http://www.xilinx.com/bvdocs/appnotes/xapp713.zip.

Directory Structure

Figure 14 below shows the directory structure of the XBERT reference design.

```
<XBERT_ROOT>                Readme file, Scripts, EDK files(XMP, MHS, MSS,
  |                         etc.)
  |
  |--- __xps                Option files for EDK tools (platgen, libgen,
  |                         compiler, etc)
  |
  |--- chipscope_vio        ChipScope project and netlist files.
  |
  |--- data                 UCF and Verilog configuration files
  |
  |--- demo                 Demo bitstream/ACE files for Xilinx ML42x
  |                         platforms
  |
  |--- etc                  Option files for Xilinx implementation tools
  |
  |--- pcores
  |       |
  |       |--- dcr2opb_bridge_v2_00_a     The DCR2OPB bridge core
  |       |
  |       |--- xbert_v1_00_a              The multi-channel XBERT core
  |
  |--- ppc_testbench        ModelSim behavioral simulation files for the
  |                         entire system (including the PPC405/software)
  |
  |--- sw                   Software codes
  |
  |--- xbert_testbench      ModelSim behavioral simulation files for the
  |                         multi-channel XBERT module (excluding the
  |                         PPC405/software)
  |
  |--- xbert_vio_projnav  The ISE project files for the VIO based XBERT
  |                         design
```

Figure 14: **XBERT Reference Design Directory Structure**

Table 17 lists some of the important source files being used in the XBERT reference design.

Table 17: **Important Source Files of the XBERT Reference Design**

File Name	Description
`./system.xmp`	Xilinx Microprocessor Project (XMP) file for the reference design to be used by Xilinx Platform Studio (XPS).
`./system.mhs`	Microprocessor Hardware Specification (MHS) file for the reference design. Defines the EDK components of the design.
`./system.mss`	Microprocessor Software Specification (MSS) file for the reference design. Defines the drivers for the various EDK components of the design.
`./config.csh`	Unix environment setup file for the reference design.
`./config.linux`	Linux environment setup file for the reference design.
`./design_config.pl`	The PERL script that customizes configuration parameters for selected ML42x platform.

Table 17: **Important Source Files of the XBERT Reference Design *(Continued)***

File Name	Description
`./data/config.v`	Verilog configuration file that defines various hardware configuration parameters for the reference design.
`./data/system.ucf`	User Constraints File that define the placement of input, output ports and specify timing constraints for the design.
`,/sw/linker_script`	Linker script that defines memory size and base address of BRAM (instruction-side and data-side) blocks.
`./sw/xbert.c`	Main application code of the design.
`./sw/xbert_sim.c`	Simplified version of the main application code for simulation purposes only.
`./ppc_testbench/compile.do`	Script to compile the entire reference design for behavioral simulation
`./ppc_testbench/simulate.do`	Script to simulate the entire reference design
`./ppc_testbench/system_tb.v`	Simulation test bench for the entire reference design.
`./xbert_testbench/simulate.do`	Script to compile and simulate the multi-channel XBERT module
`./xbert_testbench/system_tb.v`	Simulation test bench for the multi-channel XBERT module
`./xbert_vio_projnav/ xbert_vio_projnav.ise`	The ISE Project Navigator project file for the VIO-based XBERT design.

Installation of the Reference Design and Tools

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

1. Extract `xapp713.zip` into the root directory `<XBERT_ROOT>`.
2. Install the Xilinx ISE software, ModelSim SE, and EDK tools, and set up the environment variables, such as XILINX, XILINX_EDK, etc. If running Xilinx tools on Unix or Linux, the XBERT reference design provides two environment setup scripts, `config.csh` for Unix and `config.linux` for Linux under `<XBERT_ROOT>`. The user can update all the paths of tools and libraries as specified in these files, then run the following commands to set up the environment:

   ```
   >source config.unix
   ```

   or

   ```
   >source config.linux
   ```

3. If behavioral simulation is required, 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 SmartModel. Also set up the environment variable MODELSIM to point to the ModelSim initialization file provided in the reference design:

   `setenv MODELSIM <XBERT_ROOT>/ppc_testbench/modelsim_linux.ini` (on Linux)

   `setenv MODELSIM <XBERT_ROOT>/ppc_testbench/modelsim_unix.ini` (on Unix)

   `MODELSIM = <XBERT_ROOT>\ppc_testbench\modelsim_win.ini` (on Windows)

4. If behavioral simulation is required, compile the Unisim, Simprim, and EDK IP core libraries files within the Xilinx Platform Studio (XPS) GUI:

   Open the project `system.xmp` in XPS, then navigate to **Options** → **Project Options** → **HDL and Simulation** → **Simulation Libraries Path**

Set EDK and Xilinx library paths, then click **Compile**.

5. If ChipScope ILA or VIO is required, install ChipScope Pro 7.1i on a PC.

Implementation of the Reference Design

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

1. Go to `<XBERT_ROOT>` and run the PERL script `design_config.pl` to configure the XBERT reference design. The following command line configures the design in interactive mode based on user input:

   ```
   >xilperl design_config.pl -i
   ```

Alternatively, manually modify any configuration files (e.g., `system.mhs`, `system.xmp`, `data/config.v`, `data/system.ucf`). See "Design Configuration," page 35 for details.

2. Launch XPS and open the **system.xmp** project file (**File** → **Open Project** → **Load** **`system.xmp`**).
3. Select **Options** → **Project Options** and set the target device (architecture, device size, package, and speed grade).
4. Select **Tools** → **Update Bitstream**. This command compiles the software code, and synthesizes, implements, and generates the bitstream with software code updated on it. The bitstream is placed at `<XBERT_ROOT>/implementation/download.bit`.

Behavioral Simulation of the Entire Reference Design

The following steps provide the procedure for simulating the entire reference design including the PPC405 core and the C software. Note that the main application code for the simulation (`sw/xbert_sim.c`) is simplified from the real application code (`sw/xbert.c`) in order to reduce the simulation time.

1. Go to `<XBERT_ROOT>` and run the PERL script `design_config.pl` to configure the XBERT reference design. The following command line configures the design in interactive mode based on user input:

   ```
   >xilperl design_config.pl -i
   ```

2. If the MTI_LIBS, edk_nd_libs, and DUT_ROOT environment variables are not set, open the `<XBERT_ROOT>/ppc_testbench/compile.do` file, then modify the paths on MTI_LIBS, EDK_ND_LIBS, and DUT_ROOT. They should point to the Unisim/Simprim simulation library, EDK IP simulation library, and the root of the XBERT reference design, respectively.
3. Launch XPS. Open the project file `system.xmp` in XPS.
4. Change to use the simulation application code to initialize the BRAM.
5. Compile the software and generate the simulation library for the software.
6. Launch ModelSim.
7. In ModelSim, change the working directory to `<XBERT_ROOT>/simulation/behavioral`.
8. Run the following command to compile the XBERT reference design:

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

9. Run the following command to start the simulation:

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

Behavioral Simulation of the Multi-Channel XBERT Module

The following steps provide the procedure for simulating the multi-channel XBERT module, which excludes the rest of the PPC405 system and the software.

1. Go to `<XBERT_ROOT>` and run the PERL script `design_config.pl` to configure the XBERT reference design. The following command line configures the design in interactive mode based on user input:

   ```
   >xilperl design_config.pl -i
   ```
2. If MTI_LIBS, and DUT_ROOT environment variables are not set, open the `<XBERT_ROOT>/xbert_testbench/simulate.do` file, then modify the paths on MTI_LIBS and DUT_ROOT. They should point to the Unisim/Simprim simulation library and the root of the XBERT reference design, respectively.
3. Go to `<XBERT_ROOT>/xbert_testbench` and invoke ModelSim in this directory.
4. Run the following command to compile the design and start the simulation:

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

Demonstration on the Xilinx ML42x Platform

The XBERT reference design includes the demonstration bitstream on ML42x platforms. Refer to UG242, *Virtex-4 RocketIO Bit-Error Rate Tester User Guide* [Ref 1] for instructions on setting up and operating the demo on ML42x platforms.

Resource Utilization

Table 18 below lists the resource utilization of the XBERT reference design evaluated using Xilinx ISE 7.1i tools.

Table 18: **Resource Utilization of the XBERT Reference Design**

Design Options	Flip-Flops	LUTs	BRAMs	BUFGs	DCMs	GT11 CLKs	GT11s	PPCs	DSPs
PPC405 and peripherals only	378	401	40	2	1	0	0	1	0
2-channel XBERT (PPC405+VIO+ILA)	10,157	9,202	66	14	1	4	4	1	8
6-channel XBERT (PPC405+VIO+ILA)	22,639	23,636	66	20	1	4	12	1	24
6-channel VIO-based XBERT (no PPC405)	19,691	20,831	26	19	0	4	12	0	24

Notes:

1. Assume typical configurations on the XBERT reference design. Numbers change if design configuration differs.

References

1. Xilinx, Inc., UG242, *Virtex-4 RocketIO Bit-Error Rate Tester User Guide*, http://www.xilinx.com/bvdocs/userguides/ug242.pdf
2. Xilinx, Inc., UG076, *Virtex-4 RocketIO Multi-Gigabit Transceiver User Guide*, http://www.xilinx.com/bvdocs/userguides/ug076.pdf
3. Xilinx, Inc., XAPP661, *RocketIO Transceiver Bit-Error Rate Tester*, http://www.xilinx.com/bvdocs/appnotes/xapp661.pdf
4. Xilinx, Inc., XAPP762, *RocketIO X Bit-Error Rate Tester Reference Design*, http://www.xilinx.com/bvdocs/appnotes/xapp762.pdf
5. Xilinx, Inc., DS257, *Linear Feedback Shift Register v3.0*, http://www.xilinx.com/ipcenter/catalog/logicore/docs/lfsr.pdf

6. Xilinx, Inc., UG070, *Virtex-4 User Guide*, http://www.xilinx.com/bvdocs/userguides/ug070.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
06/22/06	1.0	Initial Xilinx release.
01/16/07	1.0.1	Corrected broken link to reference design archive.
04/18/07	1.1	• Table 1: Corrected RX real-time linear equalizer feature description • Table 3, User Pattern: Added references to 32- and 40-bit configurable patterns. • Table 5: Replaced with new, more comprehensive table. • Table 9: ♦ Corrected description of Address 0, POLARITY. ♦ Corrected configuration and descriptions of Addresses 4, 5, and 6 (end of table). • Table 12: Added RXCMADJ. • Table 13: Added Note (3) regarding DFE support.