XAPP581 (v1.0) October 6, 2006

Virtex-II Pro RocketIO Transceiver with 3X Oversampling for 1G Fibre Channel

Author: Vinod Kumar Venkatavaradan

Summary

This application note describes a 3X-oversampling reference design that provides a 200 Mb/s to 1000 Mb/s serial interface using the Virtex™-II Pro RocketIO™ multi-gigabit transceiver (MGT). The reference design implements a 3X-oversampling circuit at the back end of the MGT and is targeted for the Fibre Channel rate of 1.0625 Gb/s.

- Implements PCS features, such as comma detection/alignment, 8B/10B encoding/decoding, and clock correction.
- Provides single-port instantiation with a 2-byte user interface and clock correction to accommodate frequency differences between TX and RX user clocks.
- Supports far-end loopback mode to loop back data recovered at the user interface after comma alignment, 8B/10B decoding, and clock correction.

Design Description

Figure 1 shows a block diagram of the 3X-oversampling reference design. A single-port reference design consists of one Virtex-II Pro RocketIO multi-gigabit transceiver (MGT). The following sections describe selected sub-modules of the reference design in greater detail.

Clocking and Reset

The reference design consists of two DCMs, one for the RX and one for the TX side. The RX DCM divides the incoming RXRECCLK from the Virtex-II Pro RocketIO MGT by 3 and feeds it as the RX_USER_CLK. Similarly on the TX side, the TX DCM divides the incoming BREFCLK by 3 and feeds it as the TX_USER_CLK. The RXRECCLK and BREFCLK need to be set at 159.375 Mhz for a Fibre Channel rate of 1.0625 Gb/s and 3X oversampling. Target data rate = 159.375 * 20 / 3 = 1.0625 Gb/s. The reset module generates TX_USER_RESET whenever the TX DCM loses lock or SYS_RST_IN is asserted, and generates RX_USER_RESET whenever the RX DCM loses lock or SYS_RST_IN is asserted. RX_USER_RESET and TX_USER_RESET reset the appropriate blocks on the RX and TX sides respectively.

3X-Oversampling Module

The 3X-oversampling module receives the 20-bit raw data along with the RXRECCLK from the MGT, performs edge detection to determine the location of the transition point, then determines the optimal sampling location. When the sampling point decision is made, the data bits are extracted from the raw data. The oversampling module outputs 10 bits with a data enable signal. For a detailed description of the oversampling technique, refer to XAPP572, *A 3/4/5/6X Oversampling Circuit for 200 Mb/s to 1000 Mb/s Serial Interfaces*. [Ref 1]

Comma Alignment

The comma alignment module recognizes up to two 10-bit preprogrammed comma characters. It continuously monitors the 20-bit input for the presence of 10-bit comma characters at all possible 20-bit locations. The programmable option allows a user to align data on plus-comma,

minus-comma, both, or a unique user-defined and programmed sequence. If comma alignment is disabled, data is not aligned to any particular pattern.

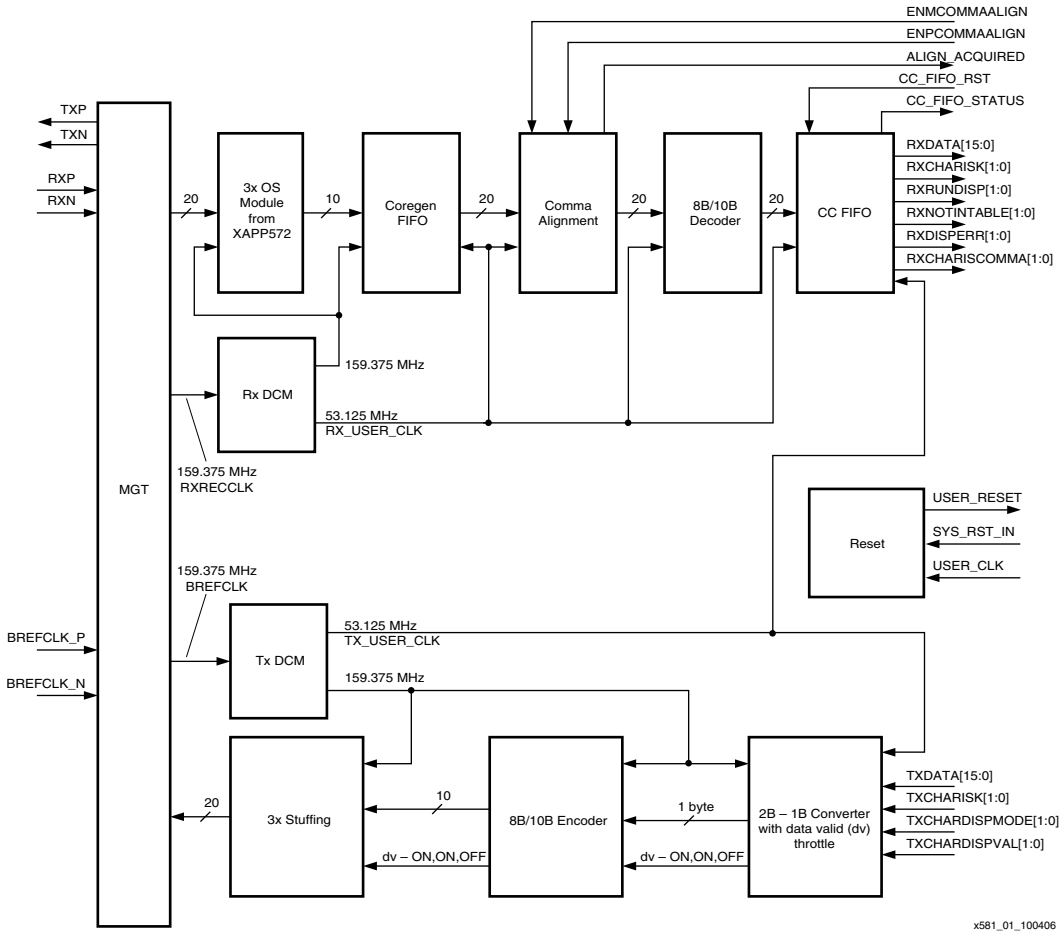

Figure 1: **RocketIO 3X-Oversampling Reference Design Block Diagram**

The comma alignment module attempts to realign the data to a proper byte boundary so that the comma appears on the most significant byte (MSB) location. The parameters MCOMMA_10B_VALUE, PCOMMA_10B_VALUE, and COMMA_10B_MASK are used by the reference design to indicate the defined comma characters to the comma alignment module. For example, MCOMMA_10B_VALUE[9:0] should be set to `1100000101` and PCOMMA_10B_VALUE[9:0] should be set to `0011111010` for the K28.5 character to accommodate transmission order in the reference design (most significant bit sent first).

Clock Correction FIFO

The decoded data, together with control signals such as RXCHARISK, are fed into the clock correction FIFO. The clock correction FIFO is a 16-address-deep asynchronous FIFO driven by RX_USER_CLK on the write side and TX_USER_CLK on the read side. The rate of the write data entering this FIFO reflects the incoming data rate; the data rate retrieved from the FIFO is determined by the rate at which the FPGA logic consumes the data.

The clock correction FIFO requires the incoming data to be properly byte-aligned so that the first byte of a CLK_COR_SEQ appears on the LSB, or byte 0, on the 20-bit internal data path. This can be accomplished by turning on the comma alignment function in the reference design and setting the comma character to match the second byte of CLK_COR_SEQ.

The following parameters must be set by the user so that the append/remove function can be used correctly:

CLK_CORRECT_USE

This parameter is used in the reference design to enable or disable the entire clock correction function. When clock correction is disabled (CLK_CORRECT_USE = FALSE), the FIFO acts like a pass-through buffer.

CLK_COR_SEQ_LEN

This parameter is used to define the length of each clock correction sequence. It can be either 2 or 4 bytes.

CLK_COR_SEQ_(1,2,3)_(1,2,3,4)

These parameters are used to set the clock correction sequences. The reference design supports three independent clock correction sequences, each defined by CLK_COR_SEQ_(1,2,3)_*.

Each clock correction sequence can be 2 or 4 bytes. CLK_COR_SEQ_*_1 and CLK_COR_SEQ_*_2 contain the first 2 bytes; CLK_COR_SEQ_*_3 and CLK_COR_SEQ_*_4 contain the last 2 bytes. CLK_COR_SEQ_*_1 holds the least significant byte transmitted/received, and CLK_COR_SEQ_*_2 holds the most significant byte transmitted/received by the MGT.

Each CLK_COR_SEQ_*_* parameter is 10-bit 8B/10B encoded or decoded data, as determined by BYPASS8B01B parameter defined in Table 1.

Table 1: **Clock Correction Sequence Bit Mapping**

Parameter	Bit Mapping of CLK_COR_SEQ_*_*
BYPASS8B10B = FALSE	Bit mapping is: [9] = 0 [8] = char is K [7:0] = 8-bit decoded data The lower 8 bits define the decoded 8-bit value after 8B/10B decoding. The ninth bit determines whether the lower 8-bit data is a K-character. The tenth bit is always 0.
BYPASS8B101B = TRUE	The entire 10-bit value specifies an 8B/10B encoded data.

CLK_COR_SEQ_(1,2,3)_MASK

These parameters are used to mask the pattern-matching on the clock correction sequences. Bit 0 of each mask parameter corresponds to CLK_COR_SEQ_*_1, bit 1 corresponds to CLK_COR_SEQ_*_2, and so on. A bit set to 0 on each mask parameter means the corresponding CLK_COR_SEQ_*_* automatically matches.

CLK_COR_SEQ_(2,3)_USE

These parameters are used to enable the second and third clock correction sequences, which should be further defined using CLK_COR_SEQ_(2,3)_* parameters.

8B/10B Decoder and Encoder

The dual 8B/10B decoder on the receiving side receives the 20-bit aligned data from the comma alignment module and decodes the received data into 2 bytes of data in parallel. The 8B/10B encoder on the transmission side generates 10-bit encoded data for transmission. This encoder and decoder perform the same functions as those in the RocketIO MGT, except for the

function of TXCHARDISPMODE and TXCHARDISPVAL, as described in Table 2. 8B/10B decoding and encoding are bypassed when BYPASS8B10B is set to TRUE.

Table 2: **8B/10B Signal Significance**

Signal Name	Value (Binary)	Function, 8B/10B Enabled (BYPASS8B101B = FALSE)	Function, 8B/10B Bypassed (BYPASS8B101B = TRUE)
TXCHARDISPMODE[0], TXCHARDISPVAL[0] (or) TXCHARDISPMODE[1], TXCHARDISPVAL[1]	00	Maintain running disparity generated by the 8B/10B encoder.	Part of 10-bit encoded byte (see Figure 2).
	01		
	10	The transmitted byte is encoded with a NEGATIVE running disparity.	
	11	The transmitted byte is encoded with a POSITIVE running disparity.	
TXCHARISK[0] (or) TXCHARISK[1]	1	Transmitted byte is a K-character	Unused.
	0	Transmitted byte is a data character	
RXCHARISK[0] (or) RXCHARISK[1]	1	Received byte is a K-character.	Part of 10-bit encoded byte (see Figure 3).
	0	Received byte is a data character.	
RXRUNDISP[0] (or) RXRUNDISP[1]	0	Indicates running disparity is NEGATIVE.	
	1	Indicates running disparity is POSITIVE.	
RXDISPERR[0] (or) RXDISPERR[1]	1	Disparity error occurred on current byte.	Unused.
	0	No disparity error on current byte.	
RXNOTINTABLE[0] (or) RXNOTINTABLE[1]	1	Indicates received byte is an invalid 8B/10B character. In this case, RXCHARISK and RXRUNDISP output the first two received bits, and RXDATA outputs the rest of the received bits (see Figure 3).	
	0	Received byte is a valid 8B/10B character.	
RXCHARISCOMMA[0] (or) RXCHARISCOMMA[1]	1	Received byte is a comma.	
	0	Received byte is not a comma.	

Figure 2 and Figure 3 illustrate the 10-bit data mapping and bit transmission order when the 8B/10B encoder and decoder are bypassed.

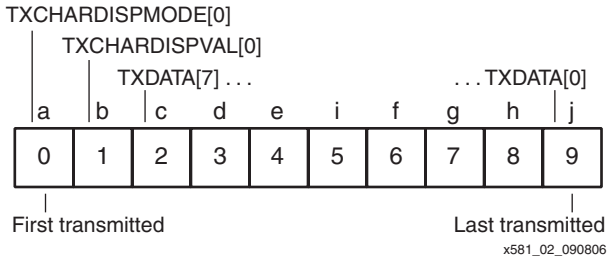

Figure 2: **10-bit TX Data Map with 8B/10B Bypassed**

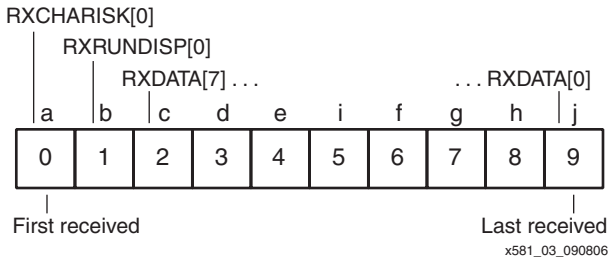

Figure 3: **10-bit RX Data Map with 8B/10B Bypassed**

3X Data Stuffing

The 3X data stuffing module in the reference design takes the 10-bit input from the 8B/10B encoder and replicates each bit three times, producing a 20-bit output to be fed to the MGT, resulting in an effective data rate of 1.0625 Gb/s (159.375 x 20 / 3).

Far-End Loopback

The reference design implements an optional far-end loopback that creates a data path from the receiving side to the transmission side. This loopback occurs at the user interface after the received data is decoded and outputted from the clock correction FIFO. The clock correction FIFO accommodates the phase differences between the RX clock domain and the TX clock domain. As a result of this clock correction FIFO, far-end loopback can be used in asynchronous as well as synchronous test setups.

Interface Description

Table 3 lists all the user interface signals that are synchronous to USER_CLK. Table 4 lists other asynchronous signals on the user interface.

Table 3: **Synchronous Signals on the User Interface**

Name	Width(1)	I/O	Description
USER_CLK	[1:0]	Output	Clock signal to the user logic. Bit [1]: TX_USER_CLK Bit [0]: RX_USER_CLK
USER_RST	[1:0]	Output	Synchronous reset signal to the user logic. Bit [1]: TX_USER_RESET Bit [0]: RX_USER_RESET
RXDATA	P x 16	Output	Received 8B/10B decoded data to the user logic. MSB byte is received first.
RXNOTINTABLE	P x 2	Output	Indicates an invalid 8B/10B character within one byte or two bytes.

Table 3: **Synchronous Signals on the User Interface *(Continued)***

Name	Width(1)	I/O	Description
RXDISPERR	P x 2	Output	Indicates a disparity error within one byte or two bytes.
RXRUNDISP	P x 2	Output	Indicates running disparity on the received bytes.
RXCHARISK	P x 2	Output	Indicates a K-character on the received bytes.
RXCHARISCOMMA	P x 2	Output	Indicates a comma character on the received bytes.
TXDATA	P x 16	Input	User data to transmit. MSB byte is transmitted first.
TXCHARISK	P x 2	Input	Indicates one or more K-character is presented in TXDATA.
TXCHARDISPMODE	P x 2	Input	Determines whether to override the 8B/10B encoder's internal running disparity. When asserted, TXDATA input is encoded based on a running disparity set by TXCHARDISPVAL port when BYPASS8B10B = FALSE. Part of the 10-bit data when BYPASS8B10B = TRUE.
TXCHARDISPVAL	P x 2	Input	Controls the running disparity against which the current input is to be encoded when BYPASS8B10B = FALSE. Part of the 10-bit data when BYPASS8B10B = TRUE.
CC_FIFO_RST	P	Input	Dedicated reset signal to the clock correction FIFO. Use this reset to clear any FIFO overflow/underflow error reported in CC_FIFO_STATUS. Assertion of this reset signal interrupts data reception.
CC_FIFO_STATUS	P x 5	Output	Indicates the following status of the clock correction FIFO using a 5-bit bus: Bit [4]: FIFO overflow error. Bit [3]: FIFO underflow error. Bit [2]: Insertion of a clock correction sequence occurred. Bit [1]: Removal of a clock correction sequence occurred. Bit [0]: A clock correction sequence is matched.

Notes:

1. P = NUM_PORTS = 1 indicates a single-port reference design.

Table 4: **Asynchronous Signals on the User Interface**

Name	Width(1)	In/Out	Description
TXP and TXN	P	Output	High-speed differential serial output from the RocketIO MGT.
RXP and RXN	P	Input	High-speed differential serial input to the RocketIO MGT.
BERFCLK_P and BREFCLK_N	[1:0]	Input	High-speed differential clock inputs. Bit [1]: OSC input Bit [0]: SMA input
REFCLKSEL	1	Input	Selects OSC input when REFCLKSEL = 1 or SMA input when REFCLKSEL = 0.

Table 4: **Asynchronous Signals on the User Interface *(Continued)***

Name	Width(1)	In/Out	Description
SYS_RST_IN	1	Input	Asynchronous system reset.
FAR_END_LOOPBACK	1	Input	Enables far-end loopback mode. In this mode, the RXDATA after decoding and clock correction is sent to the TXDATA at the user interface.
LOOPBACK	P x 2	Input	Selects the loopback mode (serial or parallel) of the MGT. Bit [1]: Serial loopback if this bit set to 1. Bit [0]: Parallel loopback if this bit set to 1.
TXOUTCLK	P	Output	Non-buffered TXOUTCLK outputs from the MGT.
RXRECCLK	P	Output	Non-buffered RXRECCLK output from the MGT.
POWERDOWN	P	Input	Powers down the receiver and transmitted of the MGT when asserted High.
TXINHIBIT	P	Input	If asserted High, the TX differential pairs are forced to a constant 1/0 (TXN = 1, TXP = 0).
TXPOLARITY	P	Input	Inverts the polarity on TXP and TXN when asserted High and sets regular polarity when deasserted.
RXPOLARITY	P	Input	Inverts the polarity on RXP and TXN when asserted High and sets regular polarity when deasserted.

Notes:

1. P = NUM_PORTS = 1 indicates a single-port reference design.

Endianness, Transmission Order, and Byte Mapping

The most significant byte (MSB) of the user interface data bus (TXDATA/RXDATA) is transmitted and received first. Most of the status and control buses (RXNOTINTABLE and TXCHARISK) correlate to a specific byte of RXDATA or TXDATA. Bit 0 of these status and control buses corresponds to byte 0 of RXDATA and TXDATA: that is, RXDATA[7:0] and TXDATA[7:0]. This scheme is shown in Table 5. These bit definitions are similar to the RocketIO MGT byte mappings as specified in [Ref 3].

Table 5: **Byte Mapping on the User Interface**

Status/Control Bus Bit (TXCHARISK, RXNOTINTABLE, etc.)	Data Bus Bits (TXDATA,RXDATA)
0	[7:0]
1	[15:8]

Reference Design

Design Configuration

The following implementation options are available for the reference design and are specified as VHDL generics, as shown in Table 6.

Table 6: **Configuration Parameters**

Parameter	Value	Description	Default Value
NUM_PORTS	1	Specify the number of ports instantiated in the design. This reference design is targeted for a single port design only.	1
PCOMMA_10B_VALUE	10-bit binary	Defines plus-comma for comma alignment. Note that the most significant bit is received first.	0011111010
MCOMMA_10B_VALUE	10-bit binary	Defines minus-comma for comma alignment. Note that the most significant bit is received first.	1100000101
COMMA_10B_MASK	10-bit binary	Defines the mask that is ANDed with the incoming data before comparison against PCOMMA_10B_VALUE and MCOMMA_10B_VALUE	1111111000
BYPASS8B10B	Boolean: FALSE/TRUE	Specifies whether to bypass the 8B/10B encoding and decoding	FALSE
CLK_COR_SEQ_1_(1,2,3,4)	10-bit binary	These define the first clock correction sequence.	BYPASS8B10B=FALSE: 0011111111 0110111100 0000000000 0000000000 BYPASS8B10B=TRUE: 1010110001 0011111010 0000000000 0000000000
CLK_COR_SEQ_2_(1,2,3,4)	10-bit binary	These define the second clock correction sequence.	0000000000
CLK_COR_SEQ_3_(1,2,3,4)	10-bit binary	These define the third clock correction sequence.	0000000000
CLK_CORRECT_USE	Boolean: TRUE/FALSE	Enable or Disable clock correction.	TRUE
CLK_COR_SEQ_2_USE	Boolean: FALSE/TRUE	Enable use of the second clock correction sequence.	FALSE
CLK_COR_SEQ_3_USE	Boolean: FALSE/TRUE	Enable use of the third clock correction sequence.	FALSE
CLK_COR_SEQ_LEN	Integer: 2 or 4	Specify the length of clock correction sequence. It affects all three clock correction sequences.	2
CLK_COR_SEQ_1_MASK	4-bit binary	Defines the mask of CLK_COR_SEQ_1_*. Bit 0 corresponds to CLK_COR_SEQ_1_1, bit 1 corresponds to CLK_COR_SEQ_1_2, and so on. A bit set to 0 automatically matches the corresponding byte (CLK_COR_SEQ_1_*)	1111
CLK_COR_SEQ_2_MASK	4-bit binary	Defines the mask of CLK_COR_SEQ_2_*. Bit 0 corresponds to CLK_COR_SEQ_2_1, bit 1 corresponds to CLK_COR_SEQ_2_2, and so on. A bit set to 0 automatically matches the corresponding byte (CLK_COR_SEQ_2_*)	1111

Table 6: **Configuration Parameters *(Continued)***

Parameter	Value	Description	Default Value
CLK_COR_SEQ_3_MASK	4-bit binary	Defines the mask of CLK_COR_SEQ_3_*. Bit 0 corresponds to CLK_COR_SEQ_3_1, bit 1 corresponds to CLK_COR_SEQ_3_2, and so on. A bit set to 0 automatically matches the corresponding byte (CLK_COR_SEQ_3_*)	1111
BYPASS_MGT	Boolean: FALSE/TRUE	If TRUE, bypasses the RocketIO MGT and replaces it with a parallel loopback path at the MGT parallel interface, connecting the transmit (GT_DATA_OUT) data to the receive data (GT_DATA_IN). This can implement a test design that contains no MGT.	FALSE

Design Hierarchy

The directory structure of the reference design is shown below. The reference design is provided in VHDL only.

```
<ROOT>                                      Root directory of the
   |                                         reference design
   |
   |------- src                             Source code directory
   |          |
   |          |------- vhdl                 VHDL top-level files
   |
   |------- build
   |          |
   |          |------- vhdl                 Sample UCF file for the
   |                     |                   reference design
   |                     |
   |                     |------- ISE_Proj   ISE project directory
   |                     |                   and project files
   |                     |
   |                     |------- cs         ChipScope™ project files
   |
   |------- test
   |          |
   |          |------- func_sim             ModelSim (*.ini files)
   |          |          |                   and config scripts
   |          |          |
   |          |          |------- vhdl       Simulation scripts (*.do)
   |          |                              and simulation working
   |          |                              directory
   |          |
   |          |------- test_vector          Test vector file (*.vec)
   |          |
   |          |------- testbench
   |                     |
   |                     |------- verilog    Tester verilog files
   |                     |
   |                     |------- vhdl       Testbench VHDL source
```

Critical Files

Table 7 lists all the critical files provided in the reference design.

Table 7: **List of Critical Files**

File Name	Description
<ROOT>/readme.txt	Readme file that contains version number, revision history, and EDA tool version of the reference design.
<ROOT>/src/vhdl/mros_top.vhd	Top level VHDL file of the reference design.
<ROOT>/build/vhdl/mros_top_ml321_2vp7.ucf	A sample UCF file for the reference design.
<ROOT>/build/vhdl/ISE_Proj/mros.ise	The Xilinx ISE™ project file for the reference design.
<ROOT>/test/func_sim/config.*	Scripts to setup simulation environment on UNIX, Linux and Windows systems.
<ROOT>/test/func_sim/modelsim_*.ini	ModelSim configuration file for UNIX, Linux and Windows environments.
<ROOT>/test/func_sim/vhdl/mros_tb_2B.do	ModelSim macro file (DO file) to run simulation of the reference design using 2-byte interface.
<ROOT>/test/test_vector/mros_test_2B.vec	2-byte Test Vector file for simulation.
<ROOT>/test/testbench/vhdl/mros_tb_2B.vhd	Testbench reference design VHDL files using 2-byte interface.

Implementation of the Design Using Xilinx ISE Project Navigator

1. Install Xilinx ISE software. <ROOT>/readme.txt file in the reference design specifies the recommended ISE version.
2. Open Project Navigator in ISE and load the project file <ROOT>/build/vhdl/ISE_Proj/mros.ise. This loads the reference design source code (top level being mros_top.vhd) and the UCF file mros_top_ml321_2vp7.ucf.
3. Modify the design to set proper system configuration parameters.
4. Modify the UCF file to set up proper location constraints according to the system being used.
5. Modify the target device in the project navigator to specify device, package and speed grade.
6. In the Source For window, select **Synthesis/Implementation** and click **MROS_TOP - rtl** to highlight the top-level VHDL file.
7. In the Processes window, right-click **Generate Programming File**, then select **Run** from the drop-down list to build the bitstream. If this succeeds, then the bitstream (MROS_TOP.bit) is saved in <ROOT>/build/vhdl/ISE_Proj.

Design Simulation

The reference design provides a behavioral simulation suite under the <ROOT>/test directory, which contains a series of testbenches, test vectors, and scripts dedicated to the user interface configuration. As shown in Figure 4, a simulation testbench contains a tester module that is connected to the user interface on the design block. The MGT high-speed I/O (HSIO) ports are connected in loopback mode on the design interface. The tester module reads a test

vector file and generates data and control signals to the TX interface on the design. The tester module also reads data from the RX interface on the design, compares it to the expected data, and generates the test result.

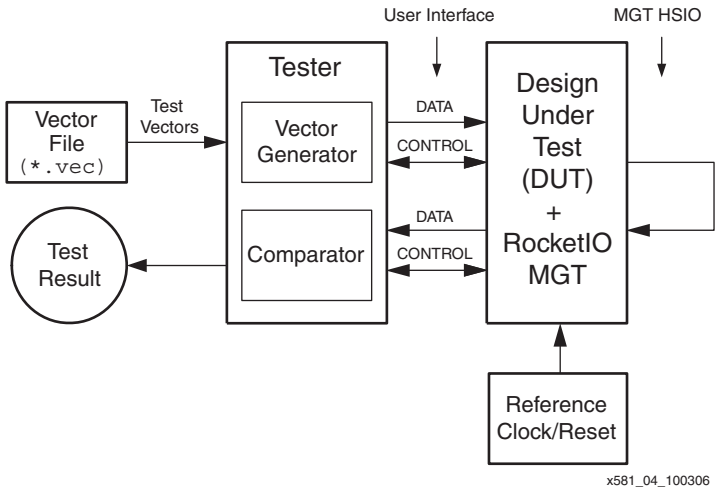

Figure 4: **Simulation Testbench**

Running Behavioral Simulation on the Design Using ModelSim

1. Install ModelSim. `<ROOT>/readme.txt` file in the reference design specifies the recommended ModelSim version.
2. Install Xilinx ISE software. `<ROOT>/readme.txt` file in the reference design specifies the recommended ISE version.
3. Make sure that SmartModel/Swift Interface is set up properly for use with ModelSim. Xilinx Answer Record #14019 explains how to use ModelSim with SmartModel/Swift Interface.
4. Depending on the operating platform, modify the Windows `config.bash`, UNIX `config.csh`, or Linux `config.linux` file under `<ROOT>/test/func_sim` to update all paths of the tools and libraries as specified in these files. Most importantly, set DUT_PATH, MODELSIM, and MTI_LIBS variables to point to valid paths.
5. Call the `config.*` script to set up the environment on Windows, UNIX, or Linux:
   a. On the Windows platform, launch the XYGWIN shell (supplied in ISE), go to `<ROOT>/test/func_sim`, then enter:

      ```
      > source config.bash
      ```

   b. On the UNIX platform, go to `<ROOT>/test/func_sim`, then enter:

      ```
      > source config.csh
      ```

   c. On the Linux platform, go to `<ROOT>/test/func_sim`, then enter:

      ```
      > source config.linux
      ```

6. Run the compxlib tool to compile simulation libraries (UNISIM, etc.) of the Virtex-II Pro family for ModelSim. The library should be compiled into the `$MTI_LIBS` directory defined in the `config.*` file. The following is an example of running this tool:

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

7. Launch ModelSim.
8. Change directory to `<ROOT>/test/func_sim/vhdl` directory. Run `mros_tb_2B.do` to simulate the design for a 2-byte interface. The waveform window should pop up automatically.

Test Setups

This section describes the common test setups with which the Design Under Test (DUT) can be used. These test setups are parallel, serial, and far-end loopback as shown in Figure 5, Figure 6, and Figure 7 respectively.

Note: For simplicity, not all blocks in the DUT are shown in these figures.

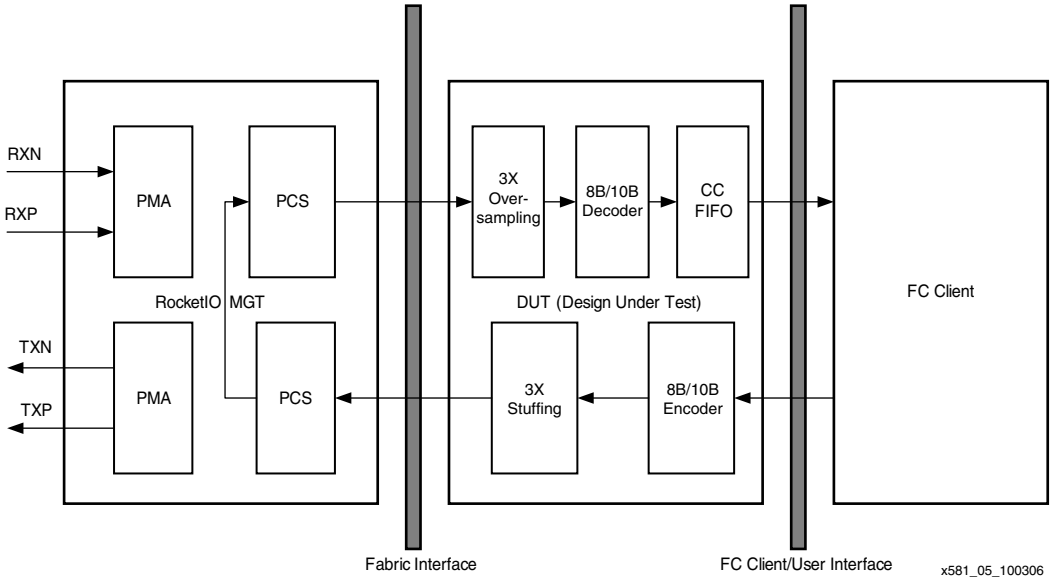

Figure 5: **Parallel Loopback Test Setup**

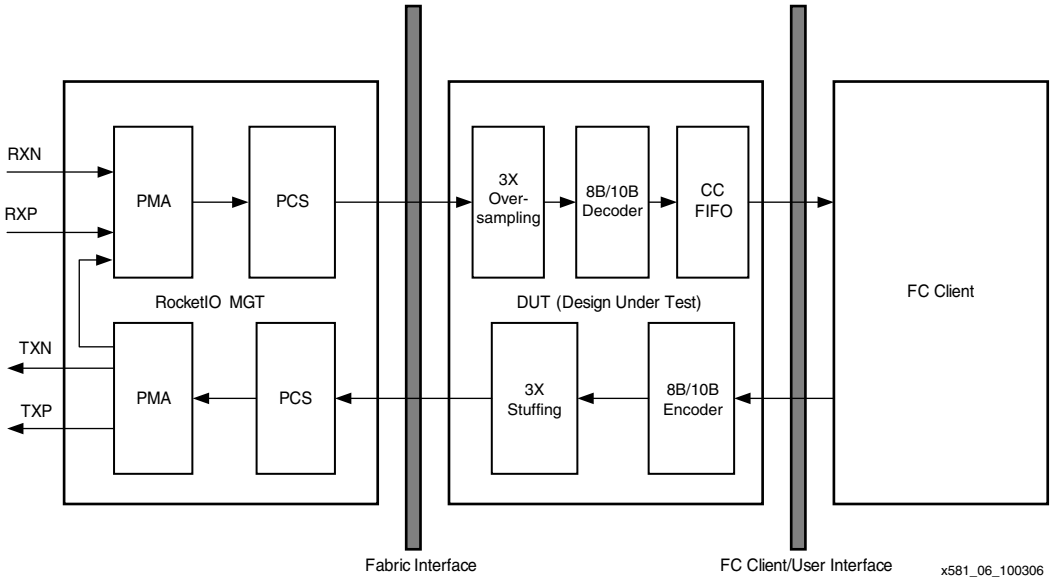

Figure 6: **Serial Loopback Test Setup**

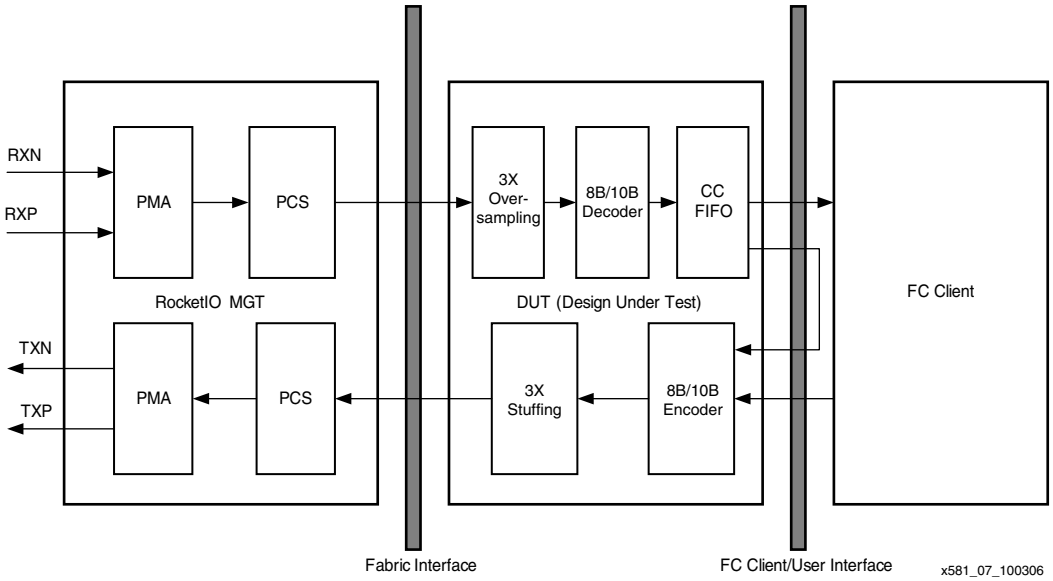

Figure 7: **Far-End Loopback Test Setup**

Resource Utilization

Table 8 shows the resource utilization of the reference design evaluated using Xilinx ISE™ 8.2i tools on an XC2VP7-FF672-6 device.

Table 8: **Resource Utilization of the Reference Design**

Design Options	Flip-Flops	LUTs	Block RAMs	PPC-405s	GCLKs	DCMs	GTs
Single-port design with all features for a 2-byte interface	1,489	1,299	39	0	7	2	1

Design Files

The reference design can be downloaded at:

http://www.xilinx.com/bvdocs/appnotes/xapp581.zip

References

The following documents provide supplementary material related to this application note:

1. Xilinx, Inc., XAPP572: *A 3/4/5/6X Oversampling Circuit for 200 Mb/s to 1000 Mb/s Serial Interfaces*, http://www.xilinx.com/bvdocs/appnotes/xapp572.pdf
2. Xilinx, Inc., UG035: *RocketIO X Transceiver User Guide*, http://www.xilinx.com/bvdocs/userguides/ug035.pdf
3. Xilinx, Inc., UG024: *RocketIO Transceiver User Guide*, http://www.xilinx.com/bvdocs/userguides/ug024.pdf
4. Xilinx, Inc., UG033: *Virtex-II Pro ML320, ML321, ML323 Platform User Guide*, http://www.xilinx.com/bvdocs/userguides/ug033.pdf

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
10/06/06	1.0	Initial Xilinx release.