XAPP572 (v1.0) November 18, 2004

A 3/4/5/6X Oversampling Circuit for 200 Mb/s to 1000 Mb/s Serial Interfaces

Author: Jerry Chuang

Summary

High-speed Serializer/Deserializer (SERDES) devices (1 Gb/s and higher) are often analog-based and tuned for a particular frequency range. If a design requires using the SERDES for lower-rate applications, such as interfacing with legacy systems, an oversampling circuit must be attached to the "back end" of the SERDES to extend the SERDES to the lower frequency ranges. The circuit works by oversampling the incoming serial data stream, evaluating data transition locations, and extracting valid data bits from the oversampled data.

The oversampling module described in this application note performs 3/4/5/6X oversampling. The oversampling ratio is selectable during operation to facilitate multi-rate applications. It is designed to accept 20 bits of oversampled data and to output 10 bits of extracted data to the user interface. This module can be used with the Virtex-II Pro™ RocketIO™ Multi-Gigabit Transceiver (MGT) to achieve line rates of 200 Mb/s to 1000 Mb/s.

When using the Virtex-II Pro RocketIO MGT, sinusoidal input jitter tolerance exceeds 0.55 UI(1) for 3/4/5X oversampling when tested with the PRBS-23 data pattern, and over 0.38 UI(2) when tested with the SONET CID data pattern.

Characterization reports and descriptions of the oversampling techniques are available in the Xilinx SPICE Lounge (http://support.xilinx.com/support/software/spice/spice-request.htm) in a document titled "RocketIO 3X-Oversampling Logic Block Characterization: Test Report."

Module Overview

Figure 1 is a block diagram of the oversampling module.

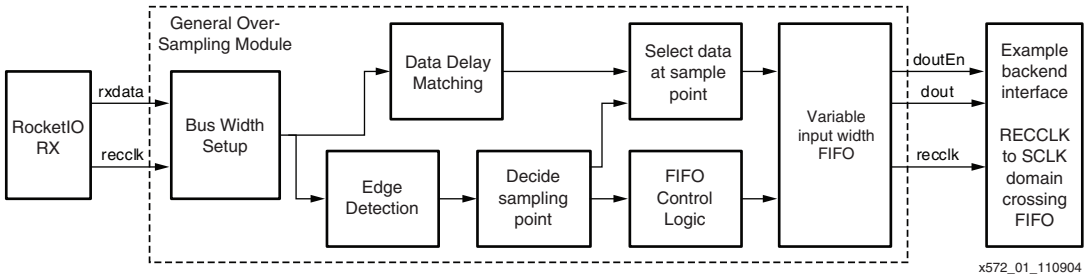

Figure 1: **Oversampling Circuit Architecture**

The RocketIO MGT transmits oversampled data (*rxdata*) to the oversampling module along with a recovered clock (*recclk*). The data is latched inside the module. Depending on the oversampling ratio (3X, 4X, 5X, or 6X), the data bus is set up accordingly for subsequent processing.

1. Sinusoidal jitter up to 5 MHz.
2. 3X oversampling still achieves 0.55 UI with SONET CID data pattern.

The oversampling module first performs edge detection to determine the location of the transition point. It then determines where the optimal sampling location is. Typically, the optimal sampling location is the point furthest from data transitions.

While the data is being edge-sampled and the sampling position determined, input data is delayed to match the decision logic. This allows the circuit to make an optimal decision for the specific incoming data sets. Once the sampling point decision is made, data bits are selected from the delayed raw data and passed on to the user.

Since the oversampling ratio might not match the data bus width of the SERDES, and since *recclk* is oftentimes not frequency-locked to the incoming data, a FIFO that allows for variable width is required. The output of the FIFO must be a fixed bus width with data enabled, clocked by *recclk*. The reference design in this application note uses a 10-bit output width.

Back-end logic can process the data based on *dout*, *doutEn*, and *recclk*. Alternatively, the back-end logic can transfer the data into a system clock domain before further processing is performed.

For a detailed description of the oversampling technique, please refer to "RocketIO 3X-Oversampling Logic Block Characterization: Test Report" available from the Xilinx SPICE Lounge (http://support.xilinx.com/support/software/spice/spice-request.htm).

Module Port Definition

Port Definitions

Figure 2 illustrates the SERDES and Fabric interface ports graphically, while Table 1, page 3, defines the characteristics of each port.

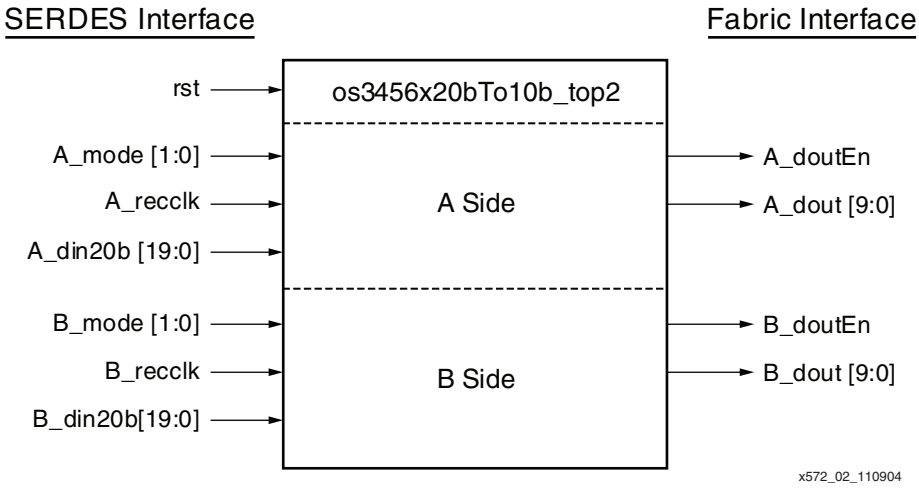

Figure 2: **Port Definition Diagram**

Table 1: **Port Definition Table**

Port	I/O	Width	Description
rst	Input	1	Logic reset. Asynchronous logic reset is used in this system.
A_mode B_mode	Input	2	Oversampling ratio selection. The oversampling ratio can be changed on the fly for multi-rate designs, or it can be a fixed value. The fixed value implementation is smaller, because it allows unused data path logic to be removed by the synthesis tool. The modes are: `00` - 3X oversampling `01` - 4X oversampling `10` - 5X oversampling `11` - 6X oversampling
A_recclk B_recclk	Input	1	The recovered clock (RXRECCLK) from the RocketIO MGT. The oversampling module uses the recovered clock from the RocketIO MGT. In addition, the RocketIO MGT's RXUSRCLK and RXUSRCLK2 must be connected to RXRECCLK. If the logic connected to RXRECCLK is small, use of a global clock (BUFG) is unnecessary. Local routing is sufficient.
A_din20b B_din20b	Input	20	Data from the RocketIO MGT. This is the oversampled data that is clocked out continuously by the RXRECCLK. The oversampling module assumes data is ordered MSB first. The MSB of *din20b* is the first bit received by the RocketIO MGT.
A_doutEn B_doutEn	Output	1	Output Data Enable. This signal indicates when output data (*A_dout* or *B_dout*) is valid.
A_dout B_dout	Output	10	Output Data. This is the recovered data after oversampling has been performed. The oversampling module assumes data is ordered MSB first. The MSB of *dout* is the first bit received by the RocketIO MGT.

Using Only One of the Two Channels

Since two channels share the same block memory for control logic state machines, the top-level wrapper includes two channels. If only Channel A is used, the input pins for Channel B must be tied to logic 0, and all Channel B outputs left unconnected. The synthesis tool or mapper tool removes the unused logic.

Usage Model

The usage of this module and the connections to the rest of the FPGA logic are described here using a 4X-oversampling example. This example assumes a line rate of 655 Mb/s and a fixed oversampling ratio of 4X. The connections are illustrated in Figure 3.

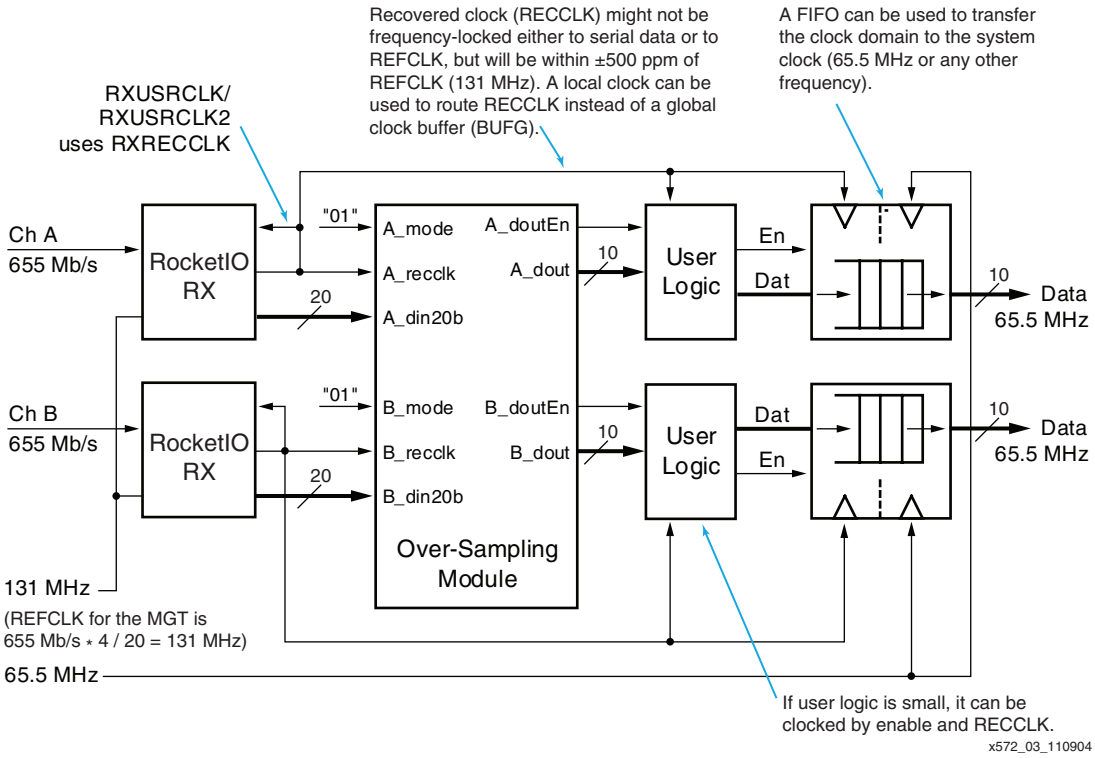

Figure 3: **Example 4X Usage Model at 655 Mb/s**

RocketIO MGT Receive Interface

To perform 4X oversampling for the 655 Mb/s data, a reference clock (REFCLK) of suitable frequency is required. This is obtained from the formula:

$$REFCLK = \frac{LineRate \times OversampleRatio}{PLLMultRatio}$$

In this design, the reference clock equates to 655 Mb/s x 4 / 20 = 131 MHz.

It is not a requirement to frequency-lock the 131 MHz REFCLK to the serial data. The oversampling module can operate properly, even if serial data, REFCLK, and RXRECCLK are not exactly frequency-locked.

It is a requirement, however, that the RocketIO MGT recovered clock RXRECCLK be used to clock the data out of the RocketIO MGT, as well as by the subsequent oversampling module. RXRECCLK should be connected to RXUSRCLK and RXUSRCLK2 using either a global clock (BUFG) or a local clock.

RXRECCLK Routing

Since the oversampling module consumes only a few hundred slices (see "Device Utilization and Performance" for details), it does not require global clock routing. Local clock routing is sufficient. The Xilinx routing tool supports local clock routing by using long lines and hex lines to create a proper clock tree. The timing tool also analyzes the relative minimum and maximum clock tree and data path delays to make sure timing violations do not occur.

For complex designs, an area group constraint is recommended to limit the size of the logic. This ensures that the logic stays within the optimal local clock tree (typically 12 x 12 CLB tiles).

If there is a large amount of user logic that also uses RXRECCLK, then global clock routing using a BUFG buffer should be considered.

Back-End Logic Interface and Clock Domain Management

The output of the oversampling module is 10-bit data with a data enable signal. Both these signals are in the RXRECCLK clock domain. Users have two options:

- Continue to use the RXRECCLK with the data enable signal to process the data
- Transfer the data to the system clock domain

It is also possible to do some processing in the RXRECCLK domain before transferring the data to the system clock domain.

When crossing clock domains from RXRECCLK to the system clock, a FIFO is typically required. The complexity of the FIFO control circuit depends on the clock frequency relationship.

Signal *doutEn* ensures that the throughput of the data is identical to the serial data. If the system clock and the serial data are frequency-locked (i.e., one is an integer multiple of the other), clock-domain crossing can be done with a simple FIFO. If the low-frequency jitter (also known as long-term frequency wander) is high, then a larger buffer might be necessary to ensure that the FIFO does not under- or overflow.

If the system clock is not frequency-locked to the serial data, then some kind of IDLE pattern insertion and removal scheme is required.

Protocols Using 8B/10B

Many protocols use 8B/10B encoding. The logic that supports 8B/10B encoding (byte alignment and 8B/10B decoder logic) can be added after the oversampling module. Figure 4 illustrates the basic concept.

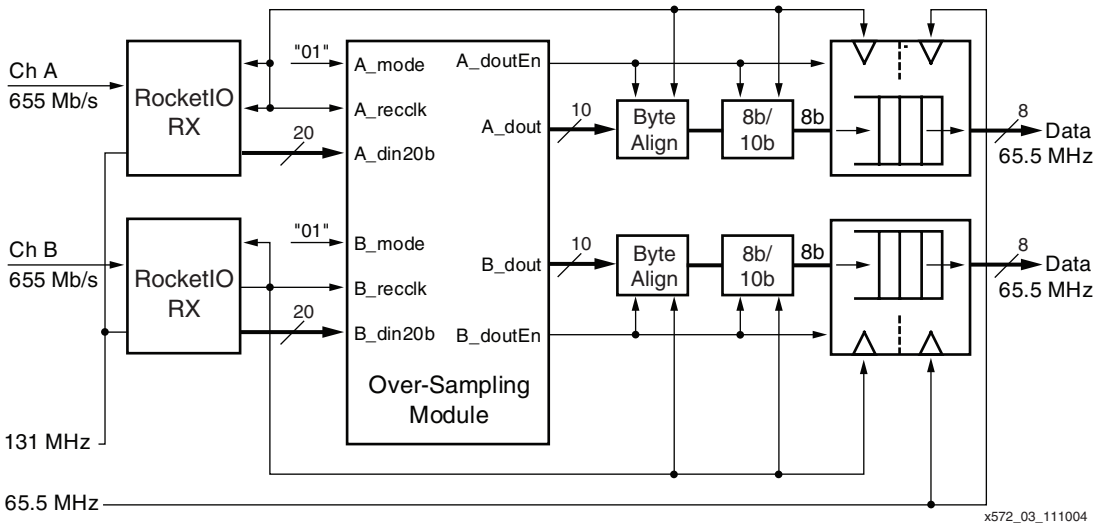

Figure 4: **Adding an 8B/10B Decoder Circuit to the Oversampling Module**

The byte alignment circuit and 8B/10B decoder run on the RECCLK domain with *doutEn*. If clock correction is necessary, that logic must be included in the elastic buffer design.

Optimization Techniques

When implemented in gates, the byte alignment logic for a 10-bit bus is essentially ten 10-to-1 muxes. Each 10-to-1 mux requires five LUTs, resulting in a total requirement for 50 LUTs.

If further reduction of logic is needed, the embedded 18 x 18 signed multiplier can be used to implement the data shifting. For example, 2 bits of shifting is achieved by multiplying the input by 4. A 19 x 10 unsigned multiplier is ideal, but the 18 x 18 signed multiplier can also be used with some user logic.

An 8B/10B encoder and decoder are provided free of charge in the XIlinx Logic Core offering. Users have the choice of implementing the encoder and decoder in LUT-based logic or memory-mapped lookup in BRAMs. When the BRAM implementation is used, two channels can share the same encoder/decoder BRAM, as they are dual-port memory.

Transition Run Length Requirement for the Oversampling Circuit

In typical analog clock recovery transceivers, there is often a maximum run length requirement that specifies the maximum number of bit cycles between two edge transitions on the serial interface. For the Virtex-II Pro RocketIO MGT, this requirement is 75 bit cycles.

When the maximum run length is violated, the recovered clock from the RocketIO MGT is not guaranteed to be frequency-locked to the serial data. The design of the RocketIO MGT, however, guarantees that the recovered clock's frequency is within ±500 ppm of the reference clock.

The oversampling module does not work in the same manner as analog transceivers. Because it does not require the RXRECCLK to be frequency locked, the 75 bit-cycle run length can be violated. Figure 5 illustrates the timing of this scenario.

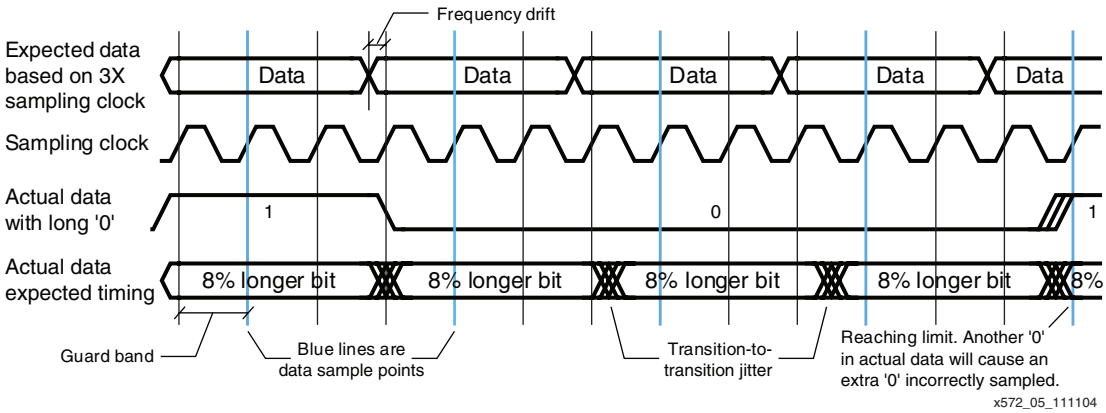

Figure 5: **Transition Run Length Requirement**

The transition run length for the oversampling circuit is a function of the following:

- [*Jitter_tt*] Transition-to-transition jitter. This is the very short-term jitter that occurs between two transitions. Based on input jitter tolerance analysis, this is typically a fraction of high-frequency (5 MHz and above) jitter.
- [*FreqDrift*] Per-cycle frequency drift caused by frequency offset between serial data and RXRECCLK. Worst case with the RocketIO MGT: 1000 ppm, or 0.001 UI.
- [*Guardband*] Minimum distance from a data transition to the sampling location. For 3X oversampling, this is 0.33 UI; for 4X oversampling, 0.25 UI; for 5X oversampling, 0.40 UI.

The equation (in UI of actual data rate) is:

$$MaxRunLength(UI) = \frac{Guardband - Jitter\_tt}{FreqDrift}$$

For 4X oversampling, the maximum run length is around (0.25 – 0.05) / 0.001 = 200 UI. *Jitter_tt* is assumed to be 0.05 UI in this example.

Interface Timing

The interface for the oversampling module is straightforward. On the SERDES interface, data is continuously streamed into the oversampling module. On the output, oversampled data is output with a data enable. Interface timing diagram for typical-case 4X oversampling is shown in Figure 6.

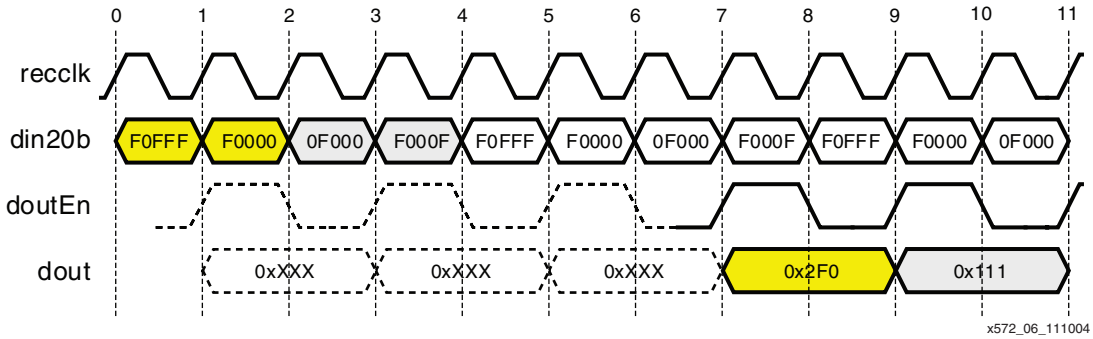

Figure 6: **Timing for 4X Oversampling, Typical Case**

Data Path Latency

The data path of the oversampling module has six RECCLK cycles of latency. However, due to the parallel-to-parallel conversion, additional delay is added.

In the case of 4X oversampling, 5 bits of actual data are typically extracted from 20 bits of sampled data from the SERDES. In order to form the 10-bit output, an additional cycle is required to gather another 5 bits. Hence, the typical-case latency is seven cycles. In cases where only 4 bits of data are extracted, eight cycles are necessary.

Table 2 lists the typical and worst-case data path delays, as well as the number of possible gaps between *doutEn* pulses, for various oversampling ratios.

Table 2: **Latency and *doutEn* Gap for Different Oversampling Ratios**

Oversampling Ratio	Recovered Bits per RECCLK	Typical Case (RECCLK cycles)	Worst Case (RECCLK cycles)	Gap Between *doutEn* Pulses
3X	6, **7**(1) bits	7	7	0, 1(2)
4X	4, **5**, 6 bits	7	8	0, 1,2
5X	3, **4**, 5 bits	8	8	1, 2
6X	**3**, 4 bits	8	9	1, 2, 3

Notes:

1. Boldface numbers are most typical.
2. See "doutEn Frequency and Timing Constraints." A gap of 1 means a `10101010` pattern.

doutEn Frequency and Timing Constraints

Over a long period of time, the frequency of the data-enable signal *doutEn* is exactly the same as that of the input serial data stream. To match the rate of the serial data, *doutEn* must make adjustments periodically. Each adjustment varies the gap between the *doutEn* pulses by ± 1 RECCLK cycle.

In the case of 4X oversampling, *doutEn* is typically asserted once every two cycles (with a gap of 1). When adjustment is necessary, back-to-back *doutEn* assertion is possible, as well as two disables between the enable assertions. This is shown in Figure 7.

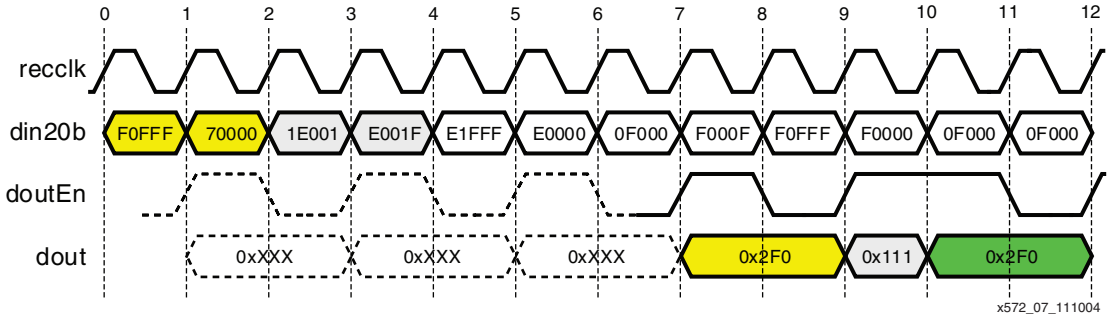

Figure 7: **Timing for 4X Oversampling, Shortened *doutEn* Case**

Implementation

Design Files

All the files necessary for the design are included in the design ZIP file xapp572.zip. The files used for synthesis and simulation are listed in the hierarchical tree below.

`os3456x20bto10b_top2.vhd` is the top-level file for the oversampling module. `tb_3456x20bTo10b_top2.vhd` is a simple simulation test bench.

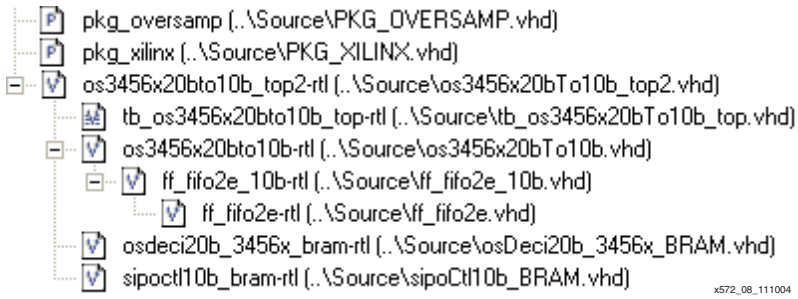

Figure 8: **Design File Hierarchy**

Additional files included in the design ZIP file are:

- `do_osTop.do` — Modelsim compile script that compiles the design and test bench files, starts simulation, and loads a simple wave format
- `wv_os3456x.do` — Simple Modelsim wave format containing the inputs and outputs
- `os3456x_test.vhd` — A wrapper file that enables users to compile for utilization and performance numbers
- `os3456x_test.ucf` — Constraint file for `os3456x_test.vhd`
- `mgtPro_20b.vhd` — Example file that shows how to instantiate the RocketIO MGT for oversampling applications

Device Utilization and Performance

The oversampling design allows users to change the oversampling ratio on the fly. However, many applications could be expected to use only one or two oversampling ratios. Utilization and performance data for both multi-ratio and 4X-only designs are shown in Table 3. These data are created using `os3456x_test.vhd` and `os3456x_test.ucf`.

Table 3: **Device Utilization and Performance**

Configuration	Slice Util	LUT Util	FF Util	BRAM Util	Frequency(1)
3/4/5/6X 2 channels	288 slices	366 LUTs	212 FFs	3 BRAMs	185MHz in -5 XC2VP20
4X only 2 channels	192 slices	156 LUTs	206 FFs	3 BRAMs	185MHz in -5 XC2VP20

Notes:

1. Top frequency required for Virtex-II Pro devices is 156.25 MHz (3.125 Gb/s line rate at 20-bit interface).

For fixed-ratio designs, unnecessary data muxes are removed to reduce LUT count. FF and BRAM utilization have little or no impact. BRAMs are used in this design to allow flexible control logic.

Figure 9 illustrates the footprint of the compiled 4X-only design. BRAMs are placed in the middle because they are shared by two channels. One channel is placed to the left and the other channel to the right of the BRAMs.

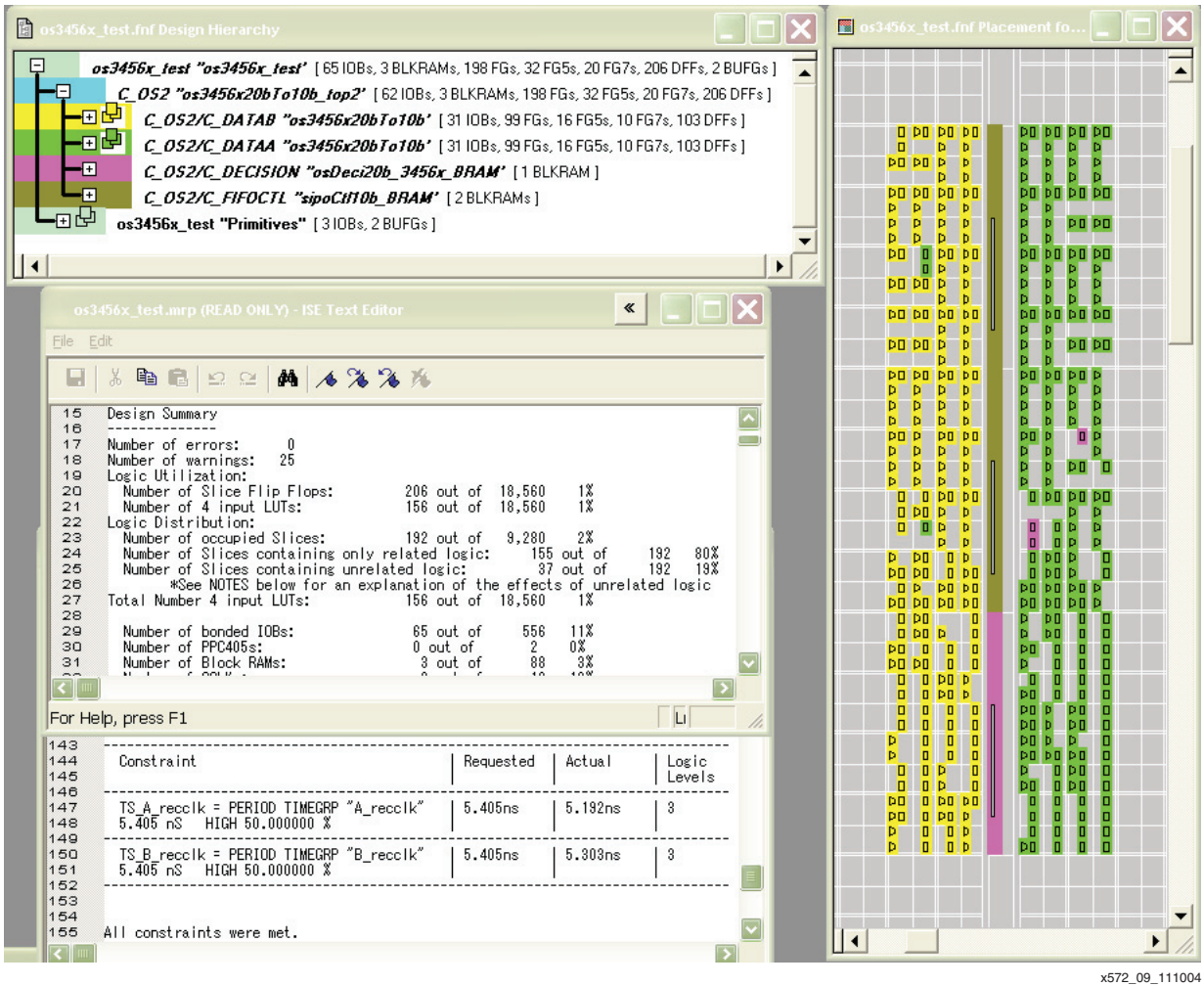

Figure 9: **4X-Only Design Compilation Results**

RocketIO MGT Instantiation Example

When using the oversampling module, most of the features in the RocketIO MGT are bypassed. An example showing RocketIO MGT instantiation is provided in `mgtPro_20b.vhd`. Common settings are listed in Table 4.

Table 4: **RocketIO MGT Attributes for Oversampling Applications**

Attribute	Value
TX_DIFF_CTRL	User Defined
TX_PREEMPHASIS	User Defined
REF_CLK_V_SEL	User Defined. '0' selects REFCLK, '1' selects BREFCLK
SERDES_10B	FALSE
TERMINATION_IMP	50
CHAN_BOND_MODE	OFF
CHAN_BOND_ONE_SHOT	FALSE
CHAN_BOND_WAIT	8
CHAN_BOND_LIMIT	16
CHAN_BOND_OFFSET	8
CHAN_BOND_SEQ_2_USE	FALSE
CHAN_BOND_SEQ_LEN	1
CHAN_BOND_SEQ_2_2	00000000000
CHAN_BOND_SEQ_1_4	00000000000
CHAN_BOND_SEQ_2_3	00000000000
CHAN_BOND_SEQ_2_1	00000000000
CHAN_BOND_SEQ_1_2	00000000000
CHAN_BOND_SEQ_1_1	00000000000
CHAN_BOND_SEQ_1_3	00000000000
CHAN_BOND_SEQ_2_4	00000000000
RX_DECODE_USE	FALSE
ALIGN_COMMA_MSB	FALSE
DEC_VALID_COMMA_ONLY	FALSE
MCOMMA_DETECT	FALSE
PCOMMA_DETECT	FALSE
DEC_PCOMMA_DETECT	FALSE
DEC_MCOMMA_DETECT	FALSE
PCOMMA_10B_VALUE	0011111000
MCOMMA_10B_VALUE	1100000000
COMMA_10B_MASK	1111111000

Table 4: **RocketIO MGT Attributes for Oversampling Applications**

Attribute	Value
RX_LOS_INVALID_INCR	1
RX_LOSS_OF_SYNC_FSM	FALSE
CLK_CORRECT_USE	FALSE
CLK_COR_INSERT_IDLE_FLAG	FALSE
CLK_COR_KEEP_IDLE	FALSE
CLK_COR_SEQ_2_USE	FALSE
CLK_COR_SEQ_LEN	1
CLK_COR_REPEAT_WAIT	1
CLK_COR_SEQ_1_1	00000000000
CLK_COR_SEQ_1_2	00000000000
CLK_COR_SEQ_1_3	00000000000
CLK_COR_SEQ_1_4	00000000000
CLK_COR_SEQ_2_1	00000000000
CLK_COR_SEQ_2_2	00000000000
CLK_COR_SEQ_2_3	00000000000
CLK_COR_SEQ_2_4	00000000000
TX_DATA_WIDTH	2
RX_DATA_WIDTH	2
RX_BUFFER_USE	TRUE
TX_BUFFER_USE	TRUE
RX_LOS_THRESHOLD	4
CRC_FORMAT	USER_MODE
RX_CRC_USE	FALSE
TX_CRC_USE	FALSE
CRC_START_OF_PKT	K27_7
CRC_END_OF_PKT	K29_7
TX_CRC_FORCE_VALUE	11010110

Hardware Characterization Design

Hardware characterization of the oversampling module using the ML321 RocketIO MGT Evaluation Board is not included in the design ZIP file, but is available. Please contact a Xilinx Field Application Engineer to obtain the design.

The design allows:

- Cable loopback testing
- FR4 trace length testing
- Bit Error Rate (BER) testing

- Varying PRBS sequence testing
- Data eye observation.

The architecture of the design is illustrated in Figure 10.

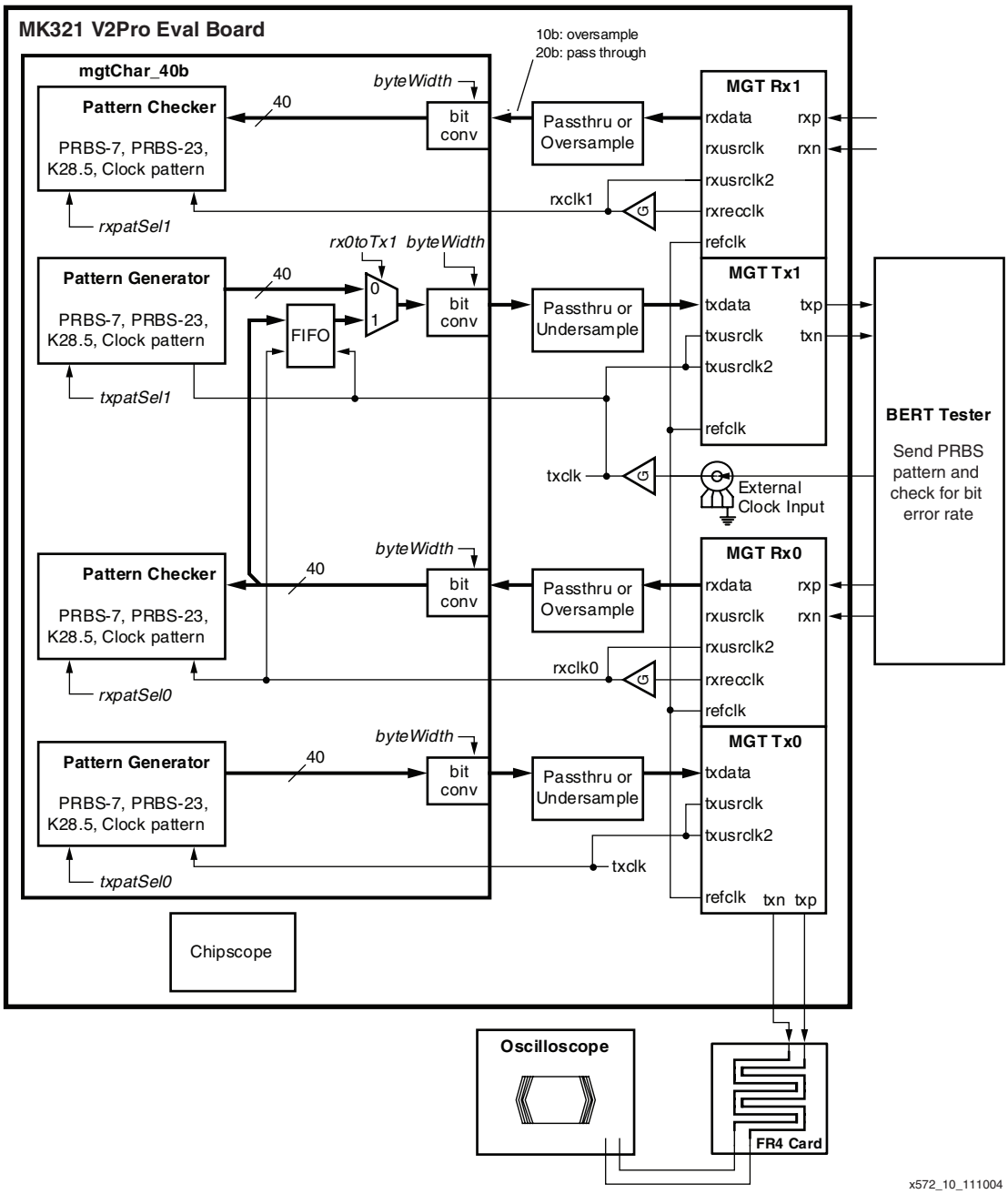

Figure 10: **Hardware Characterization Design for ML321 Evaluation Board**

Additional Information

Xilinx is able to provide further assistance with special-case designs. For applications implementing SD-SDI functionality, please refer to Xilinx Application Note XAPP684, "Multi-Rate HD/SD-SDI Receiver Using Virtex-II Pro RocketIO Multi-Gigabit Transceivers." For all applications requiring line rates less than 200 Mb/s, or for additional help with other special-case design requirements, please contact the Xilinx Support Hotline.

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
11/18/04	1.0	Initial Xilinx release.