XAPP875 (v1.1) January 13, 2010

Dynamically Programmable DRU for High-Speed Serial I/O

Author: Paolo Novellini and Giovanni Guasti

Summary

Multi-service optical networks today require the availability of transceivers that can operate over a wide range of input data rates. High-speed serial I/O has a native lower limit for operating data rates, preventing easy interfacing to low-speed client signals. The non-integer data recovery unit (NI-DRU) presented in this application note is specifically designed for RocketIO™ GTP and GTX transceivers in Virtex®-5 LXT, SXT, TXT, and FXT platforms and consists of look-up tables (LUTs) and flip-flops. The NI-DRU extends the lower data rate limit to 0 Mb/s and the upper limit to 1,250 Mb/s, making embedded high-speed transceivers the ideal solution for true multi-rate serial interfaces.

The NI-DRU's operational settings (data rate, jitter bandwidth, input ppm range, and jitter peaking) are dynamically programmable, thus avoiding the need for bitstream reload or partial reconfiguration. Operating on a synchronous external reference clock, the NI-DRU supports fractional oversampling ratios. Thus, only one BUFG is needed, independent of the number of channels being set up, even if all channels are operating at different data rates.

Given the absence of a relationship between the reference clock and incoming data rate, two optional barrel shifters ease the interfacing of the NI-DRU with an external FIFO and/or with any required decoder. The first barrel shifter has a 10-bit output that can be easily coupled to an 8B/10B or 4B/5B decoder (both not included in the reference design). The second barrel shifter has a 16-bit output and is specifically designed for 8-bit protocols, such as SONET/SDH. Other barrel shifters can be designed by the user.

Introduction

This application note is divided into three main parts:

- NI-DRU usage model
- Simulating the NI-DRU
- Testing the NI-DRU on the ML523 RocketIO transceiver characterization platform, revision C or higher [Ref 1]

The NI-DRU usage model section describes a block diagram of the NI-DRU in detail, and the overall transfer function of the DRU is calculated as a function of all the hardware settings.

Configuring the NI-DRU, page 4 guides the user algorithmically in sizing the hardware parameters ($G1$, G_2, G_{1_P}, CENTER_F), starting from the application requirements. Simulating the DRU, page 8 describes the testbench developed to simulate the DRU, while Testing the DRU on the ML523 Characterization Platform, page 9 describes the testbench that shows the DRU at work on the ML523 characterization platform.

The Fast Ethernet case (125 Mb/s ± 100 ppm) and OC3/STM1 (155.520 Mb/s ± 20 ppm) are used as a practical example throughout the application note.

NI-DRU Usage Model

This section describes how to translate the application requirements into hardware settings. It provides introductory material useful to understanding Configuring the NI-DRU, page 4.

NI-DRU Block Diagram

Figure 1 shows a block diagram of the NI-DRU.

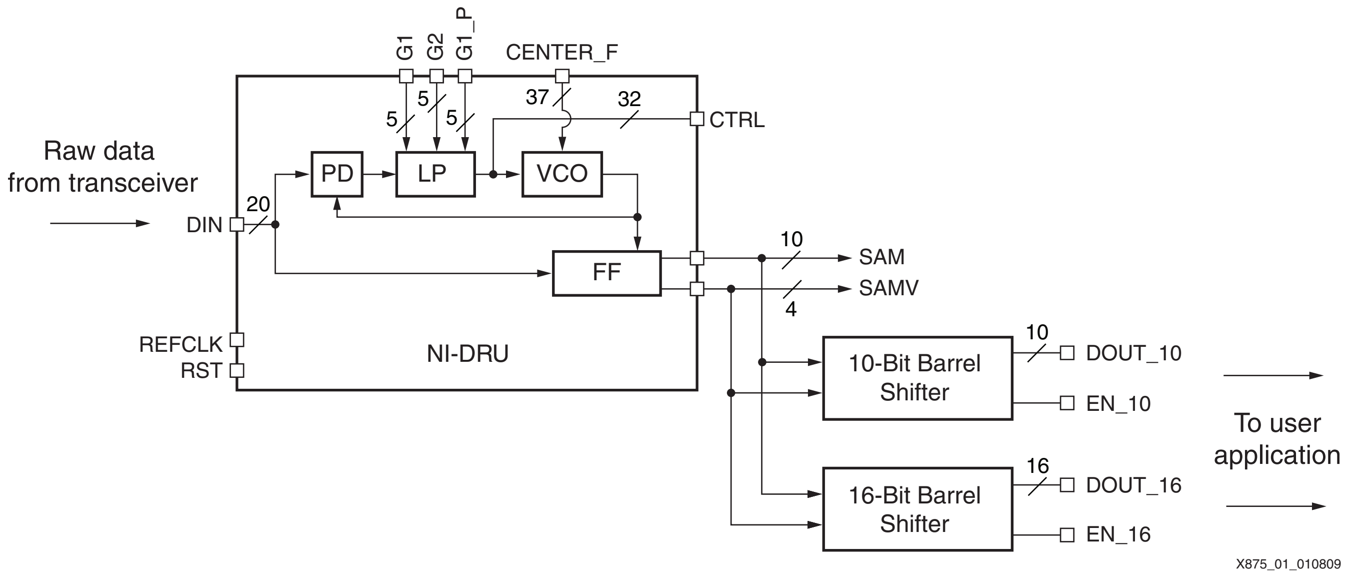


Figure 1: **NI-DRU Block Diagram with Optional Barrel Shifters**

Data In (DIN) receives oversampled data 20 bits wide (with the LSB being the oldest bit[1]) from the transceiver set in lock-to-reference mode. Lock-to-reference mode is described in the GTP and GTX transceiver user guides [Ref 2] [Ref 3]. The phase detector (PD) looks for transitions in the incoming data, continuously comparing the phase of the incoming data with the phase of the internal digital voltage-controlled oscillator (VCO). The error signal generated by the PD and filtered out by the low-pass filter (LP) corrects the VCO frequency to minimize the phase error, thus realizing the phase-locked loop (PLL) functionality of the NI-DRU [Ref 4] [Ref 5] [Ref 6]. Based on the VCO output, the flip-flop block (FF) selects the samples that are positioned in the middle of the eye. There can be up to 10 valid samples in each REFCLK cycle, which are placed by FF on the SAM output. SAMV indicates the number of valid samples on SAM. The NI-DRU and the barrel shifters are synchronized to REFCLK.

The PLL operates in parallel over the incoming data, producing more than one bit output for each clock cycle. The relationship between the operating frequency and the incoming data rate dictates the maximum number of bits processed per clock cycle, N_{MAX}, according to Equation 1.

$$N_{MAX} = truncate\left[\left(\frac{f_{DIN}}{f_{REFCLK}}\right)_{MAX}\right] + 1 \qquad \textit{Equation 1}$$

For example, in the case of Fast Ethernet with REFCLK = 125 MHz and $N_{MAX} = 2$, if N is greater than 1, the optional barrel shifters ease the interfacing of the DRU to a fixed-width FIFO.

1. This endianness is consistent with that used in Virtex-4 and Virtex-5 FPGA transceivers. The MGTs in Virtex-II Pro FPGAs serialize and deserialize MSB first.

Table 1 describes the pinout of the NI-DRU.

Table 1: **Pinout Description of the NI-DRU**

Pin Name	Type	Description	Comment
CENTER_F[36:0]	In	Center frequency of the NI-DRU.	Setting CENTER_F, page 6 describes the method to set this vector. This setting can be changed dynamically to support true multi-rate applications.
CTRL[31:0]	Out	The ppm frequency difference between REFCLK and the incoming data rate.	This debug output allows the user to read the ppm difference between REFCLK and the incoming data rate. Refer to Setting CENTER_F, page 6 for details on how to interpret this output.
DIN[19:0]	In	Data enters the NI-DRU on this pin in a parallel fashion. This pin connects directly to the transceiver interface.	The frequency and its associated ppm tolerance can be programmed dynamically via CENTER_F and G_1 respectively. Bit 0 is the oldest.
DOUT_10[9:0]	Out	10-bit barrel shifter data output.	DOUT_10 is synchronous to REFCLK. Data on DOUT_10 is valid when EN_10 = 1.
DOUT_16[15:0]	Out	16-bit barrel shifter data output.	DOUT_16 is synchronous to REFCLK. Data on DOUT_16 is valid when EN_16 = 1.
EN_10	Out	10-bit barrel shifter enable output.	EN_10 is synchronous to REFCLK. When EN = 1, DOUT_10 carries 10 valid bits.
EN_16	Out	16-bit barrel shifter enable output.	EN_16 is synchronous to REFCLK. When EN = 1, DOUT_16 carries 16 valid bits.
G1[4:0]	In	Post gain of the integral path.	G_1, G_2, and G_{1_P} configure the DRU bandwidth and the DRU jitter peaking. Refer to Selecting G1 and G1_P, page 7.
G1_P[4:0]	In	Pre-gain of the integral path.	G_1, G_2, and G_{1_P} configure the DRU bandwidth and the DRU jitter peaking. Refer to Selecting G1 and G1_P, page 7.
G2[4:0]	In	Gain of the direct path.	G_1, G_2, and G_{1_P} configure the DRU bandwidth and the DRU jitter peaking. Refer to Selecting G2, page 6.
REFCLK	In	This reference clock for the NI-DRU is shared with the GTX transceiver.	Selecting REFCLK Frequency, page 5 contains an algorithmic guide to select the frequency of this clock.
RST	In	Asynchronous reset.	This active-Low input resets the whole DRU. The DRU has no unknown/unrecoverable states, so the user is not required to reset the machine. The RST pin is provided for debug and simulation purposes. The Virtex-5 FPGA GTP/GTX transceivers need to be reset according to the description in the GTP and GTX transceiver user guides [Ref 2] [Ref 3].
SAM[9:0]	Out	Output data.	Valid bits are in the least significant part of the vector.
SAMV[3:0]	Out	Number of valid bits.	SAMV is synchronous to REFCLK and equals the number of valid bits. This output is coded.

NI-DRU Transfer Function

The DRU transfer function can be customized by the input pins, G_1, G_2, and G_{1_P}. This section calculates the most general transfer function of the NI-DRU and shows its dependency on G_1, G_2, and G_{1_P}. Then, these parameters are sized algorithmically. The impact of the designer's choices on loop stability, loop bandwidth, residual phase error and damping factor is shown.

The detailed math behind the calculation of the transfer function is beyond the scope of this application note and requires a detailed look into the architectural structure of the PLL. Equation 1 gives the final result for the transfer function *H(s)*.

$$H(s) = \frac{K_d K_v 2^{-G_2} s + K_f K_d K_v 2^{-(G_1 + G_{1\_P} + 1)}}{s^2 + K_d K_v 2^{-G_2} s + K_d K_v K_f 2^{-(G_1 + G_{1\_P} + 1)}} \qquad \text{Equation 2}$$

Where:

$$\text{VCO gain: } K_v\left[\frac{rad}{num}\right] = \frac{f_{REFCLK} \cdot 2 \cdot \pi}{2^{32}} \qquad \text{Equation 3}$$

$$\text{PD gain: } K_d\left[\frac{num}{rad}\right] = \frac{2^{15}}{\pi} \qquad \text{Equation 4}$$

$$\text{Integral filter gain: } K_f[s^{-1}] = \sigma \cdot f_{DIN} \qquad \text{Equation 5}$$

Where:

σ is the transition density, a dimensionless number. The value of σ is 1 for maximum transition density (pattern `101010`) and 0 for no transitions. Equation 5 assumes that $\sigma = 0.5$.

The resulting bandwidth, f_{BW}, and damping factor, ξ, are reported in Equation 6 and Equation 7 respectively, with dimensions in square brackets.

Equation 6 is called the natural frequency and is equal to f_{BW} only for $\xi = 0$, i.e., with no integral path. This approximation is very useful because it calculates the bandwidth with only one hardware parameter, G_2. For low values of *x* (typical for DRUs), it is advantageous to not have excessive jitter peaking in the transfer function.

$$f_{BW}[Hz] = \frac{K_D K_V 2^{-G_2}}{2\pi} \qquad \text{Equation 6}$$

$$\zeta[\bullet] = 2^{-(G_2+1)} \sqrt{\frac{K_V \cdot K_D}{2^{-(G_1 + G_{1\_P} + 1)} \cdot K_F}} \qquad \text{Equation 7}$$

Configuring the NI-DRU

This section provides the necessary steps to correctly configure the NI-DRU. These steps include selecting the REFCLK frequency, the CENTER_F input, and the G_1, G_2, and G_{1_P} gains. These values are needed to determine the hardware settings of the NI-DRU:

- Incoming data rate (f_{DIN}), with associated ppm tolerance (Ppm_{DIN})
- Available reference clock frequency (f_{REFCLK}), with associated ppm tolerance (Ppm_{REFCLK})

The NI-DRU does not require that the transceiver oversample the incoming data rate at an integer multiple of the input frequency. In other words, the oversampling ratio can be fractional.

Selecting REFCLK Frequency

The reference clock must be supplied on the CLK pin and serves two purposes:

- Drives the transceiver (the GTP and GTX transceiver user guides contain the limitations on the clock [Ref 2] [Ref 3])
- Clocks the DRU

Both purposes constrain REFCLK to be in a specific range. Because both purposes have to be satisfied, the intersection of the two ranges must be considered. Table 2 lists the allowable ranges of f_{REFCLK} for various speed grades. The lower limit of 60 MHz is determined by the minimum allowed REFCLK frequency of the GTP and GTX transceivers. The upper limit is determined by the FPGA logic performance, and is thus speed-grade dependent.

Table 2: **Maximum NI-DRU Frequency Depending on the Speed Grade**

Speed Grade	Minimum f_{REFCLK} (GTP and GTX Transceivers)	Maximum f_{REFCLK} (GTP and GTX Transceivers)	Units
-1	60[1]	160	MHz
-2		187.5	MHz
-3		187.5	MHz

Notes:

1. For details, refer to the Virtex-5 FPGA data sheet [Ref 7].

For any choice of f_{REFCLK} inside the range defined in Table 2, the resulting oversampling rate (O_R), or average number of samples placed inside each unit interval (UI), is given by Equation 8.

$$O_R = \frac{20 \cdot f_{REFCLK}}{f_{DIN}} \qquad \text{Equation 8}$$

Although O_R needs to be larger than 2 from the theoretical point of view, it is recommended that O_R be kept larger than 3 or 4 to have a reliable link connection. The reason is that any oversampling algorithm reduces the theoretical high-frequency jitter tolerance by a fraction of the UI as given by Equation 9.

$$UI_{LOST} = (O_R)^{-1} \qquad \text{Equation 9}$$

The theoretical high-frequency jitter tolerance is given by 1 UI reduced by the metastability time window of the first sampling flip-flop. Setting O_R close to 2 means reducing the eye aperture by 50%—a very critical condition. This limitation is present in all oversampling algorithms.

The resulting minimum and maximum data rates (f_{DIN}) at which the DRU can work are listed in Table 3. A digital clock manager (DCM) or PLL can be used to increase the flexibility in the choice of f_{REFCLK}. Specifically, the f_{REFCLK} provided to the transceiver can be increased if a DCM or PLL is used to divide the clock provided to the NI-DRU.

Table 3: **Maximum Theoretical and Recommended Data Rates**

Speed Grade	Minimum f_{DIN}	Maximum[1] Theoretical f_{DIN} (2X Oversampling)	Maximum Recommended f_{DIN} (3X Oversampling)	Units
-1	0	1600	1065	Mb/s
-2		1875	1250	Mb/s
-3		1875	1250	Mb/s

Notes:

1. The operating data rate must be strictly less than, and never equal to, the maximum theoretical f_{DIN} in the table.

Setting CENTER_F

CENTER_F is a 37-bit input vector that sets the f_{DIN} at which the DRU operates. This value is derived from Equation 10.

$$\text{CENTER\_F} = f_{DIN} \cdot \frac{2^{32}}{f_{REFCLK}} \qquad \textit{Equation 10}$$

The f_{DIN} in Equation 10 is the incoming data rate at which the DRU is desired to operate. Equation 10 uses this target f_{DIN} to determine a value of CENTER_F, which in turn sets the actual f_{DIN} at which the DRU operates. One of the main advantages of the NI-DRU presented in this application note is that the ratio between f_{DIN} and f_{REFCLK} can be non-integer.

Using Fast Ethernet as an example, the user can select 125 MHz as a reference clock, setting the oversampling ratio to 20. Thus, with f_{DIN} = 125 Mb/s and f_{REFCLK} = 125 MHz (integer operating mode), the CENTER_F value derived from Equation 10 is:

```
b0000100000000000000000000000000000000
```

For OC3 with a reference clock of 125 MHz, f_{DIN} is 155.520 Mb/s and f_{REFCLK} is 125 MHz (non-integer operating mode) and the CENTER_F value is calculated as:

```
b0000100111110100000010100010100001110
```

Selecting G_2

Both the reference clock and the incoming data rate have ppm tolerances. The sum of these two tolerances is the starting point for defining the values for G_1 and G_2. G_2 controls the direct path in the filter loop and is determined by the maximum ppm difference between the incoming data rate and the reference clock.

If the frequency of DIN is $f_{DIN} \pm Ppm_{DIN}$ and the frequency of REFCLK is $f_{REFCLK} \pm Ppm_{REFCLK}$, then the maximum ppm difference is defined by Equation 11.

$$Ppm_{PEAK} = Ppm_{DIN} + Ppm_{REFCLK} \qquad \textit{Equation 11}$$

For example, Fast Ethernet has a $Ppm_{PEAK} = 200$ and OC3/STM1 has a $Ppm_{PEAK} = 40$.

Every bit of the 32-bit encoded CTRL signal steps the VCO through its resolution in frequency (Res), independently of the CENTER_F value, as given by Equation 12.

$$Res = \frac{f_{REFCLK}}{2^{32}} \qquad \textit{Equation 12}$$

Thus, the number of bits (N) required for any Ppm_{PEAK} number is given by Equation 13.

$$N = Roundup\left[\log_2 \frac{2 \cdot Ppm_{PEAK} \cdot f_{DIN} \cdot 10^{-6}}{Res}\right] \qquad \textit{Equation 13}$$

Where:

- Ppm_{PEAK} is the maximum ppm difference between the incoming data rate and reference clock. The factor 2 considers the internal signed notation of the 32-bit encoded CTRL signal.
- f_{DIN} is the incoming data rate in Hz.
- Res is the resolution, defined in Equation 12.

In Equation 11, Ppm_{PEAK} should always have added margin so that a 25 to 50% margin can be safely added when calculating N.

The G_2 coefficient can be calculated from Equation 14.

$$G_2 = 32 - N \qquad \text{Equation 14}$$

Solving Equation 13 for the case of Fast Ethernet, N equals 21. According to Equation 14, G_2 should be set to 11.

According to Equation 6, fixing G_2 means fixing the NI-DRU bandwidth. The criteria described here forces the lock-in bandwidth to equal the worst-case ppm difference between the data rate and the reference clock, meaning that the DRU never undergoes a bit slip during the lock process. Setting G_2 to a value lower than the one resulting from Equation 14 forces the lock-in bandwidth to be higher than the worst-case ppm difference. This has the advantage of increasing the NI-DRU bandwidth but still avoiding the bit-slip phenomenon during the lock-in process.

The bandwidth of the NI-DRU should only be increased up to a point at which the amplification of phase errors does not reduce the high-frequency jitter tolerance. Setting G_2 higher than the worst-case ppm difference forces the NI-DRU to work in the pull-in area and outside the lock-in range, making the lock process highly non-linear.

If reduced bandwidth is needed, it is possible to dynamically control G_2 to achieve high lock-in bandwidth at start-up and low bandwidth after the lock process is over. Refer to technical publications for definitions of lock-in and pull-in bandwidths [Ref 4] [Ref 5].

Selecting G_1 and G_{1_P}

G_1 and G_{1_P} control the gain of the integral path in the loop filter. The presence of integral loop gain has both advantages and disadvantages. The advantage is, in the steady state, the NI-DRU is locked with zero phase error. In other words, a phase error is not required at the input of the phase detector to pull the VCO from its center frequency.

The disadvantages are:

- The overall structure with two complex poles can be oscillating, reducing the high-frequency jitter tolerance.
- The transfer function always exhibits jitter peaking, which can be reduced but never set to zero. From an engineering perspective, jitter peaking can be constrained by properly sizing G_1 and G_{1_P}, but never reduced to 0 dB.

G_1 can be calculated from Equation 15.

$$G_1 = 31 - N \qquad \text{Equation 15}$$

Selecting a much different value for G_1 can force the NI-DRU to work with unexpected jitter peaking (in the case of a higher G_1) or not cover the full, required ppm range (in the case of lower G_1).

Many criteria can be defined to set G_{1_P}. One choice is to set G_{1_P} such that the transfer function exhibits maximum flatness in the transient response. In this case, Equation 16 applies.

Note: The criterion described here is one of several possible choices. An alternative, not described in this application note, is to set G_{1_P} to constrain the jitter peaking.

$$\xi = \frac{\sqrt{2}}{2} \qquad \text{Equation 16}$$

After ξ has been defined by Equation 16, G_{1_P} can be calculated from Equation 17.

$$G_{1\_P} = round\left\langle \log_2 \frac{K_F \xi^2}{K_D K_V 2^{(G_1 - 2G_2 - 2)}} \right\rangle \qquad \text{Equation 17}$$

Simulating the DRU

The purpose of the TB_SIM_DRU_JITTER testbench is to simulate the ability of the NI-DRU to operate at many different data rates (shown in Table 4) with synchronous and plesiochronous inputs. This testbench works with a single reference clock frequency of 155.52 MHz. The user can also apply any ppm difference to evaluate the capability of the NI-DRU.

Table 4: **Supported Data Rates in the Simulation Testbench**

Application Name	Data Rate(1) (Mb/s)	Oversampling Ratio (O_R)
Fast Ethernet	125	24.88X
SONET OC1	51.84	60X
SONET OC3 / SDH STM1	155.520	20X
PDH E4 (ITUT G703)	139.264	22.33X
Proprietary example	510	6.1X
Proprietary example	1000	3.11X
Proprietary example	1250	2.49X

Notes:

1. Other rates can be tested by reconfiguring the inputs G_1, G_2, G_{1_P}, and CENTER_F of the NI-DRU.

The architecture implemented in the TB_SIM_DRU_JITTER testbench is shown in Figure 2.

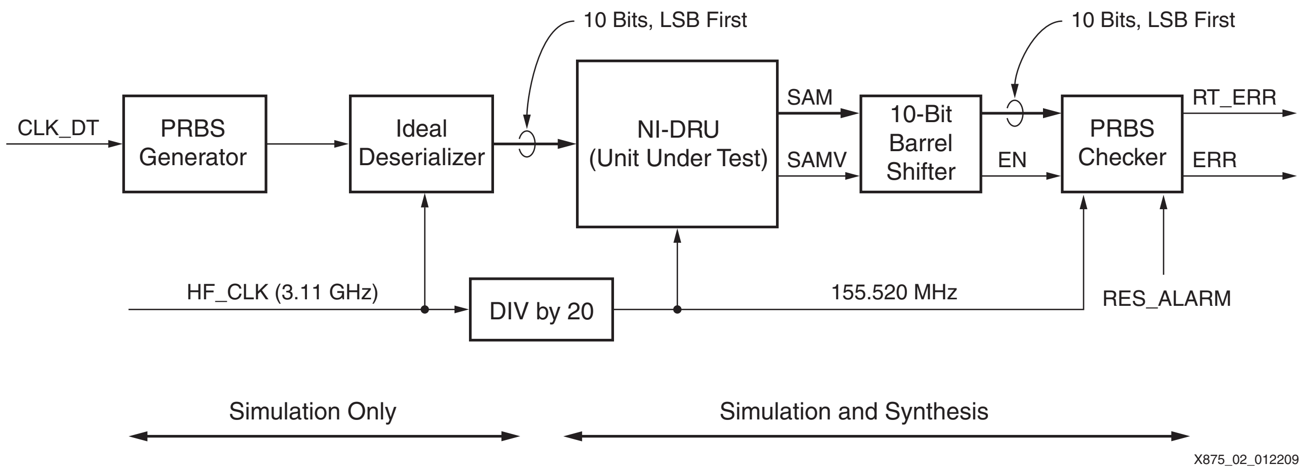


Figure 2: **Block Diagram of TB_SIM_DRU Testbench**

The testbench contains two clock domains:

- The clock domain of the line, synchronized to HF_CLK
- The clock domain of the DRU, synchronized to REFCLK

The DRU (configured for the Fast Ethernet case) works with a maximum 250 ppm difference in frequency between the two clock domains, exceeding the 200 ppm requirement from the Fast Ethernet physical specification [Ref 8].

The pseudorandom binary sequence (PRBS) generator works at full speed on CLK_DT and can generate a PRBS 15 pattern. The ideal deserializer, the DRU, and the PRBS checker all work on the local REFCLK domain. The DRU and the PRBS checker are synthesizable.

In the testbench, an instance of the GTX transceiver is replaced with an ideal deserializer working at 3.11 GHz. This choice allows for faster simulation times and preserves the validity of the results. The user can also set up a full simulation where the GTP or GTX transceiver SmartModel is used in place of the ideal deserializer.

The PRBS checker is synthesizable and works in parallel, while the PRBS generator is serial. The 10-bit output of the DRU is processed only when EN = 1.

The RT_ERR (real-time error) output is useful during simulation debug. The signal is set High when an error is detected and Low when the 10-bit vector is correct. The ERR output goes High when an error is detected and is reset by ERR_RST. Thus, RT_ERR is useful in simulation, while ERR is commonly used during hardware tests.

An example output is shown in Figure 3. A 125 Mb/s ±25 ppm input is recovered using a 155.52 MHz (24.88X) reference clock (the control signal is shown going Low to adjust the VCO frequency to the incoming data rate). The `/scripts/wave.do` file can be used to plot the waves and scale the CTRL signal to view the linear response of the DRU. Although the plot in Figure 3 looks like a first-order response, it is actually a second-order response with very low overshoot.

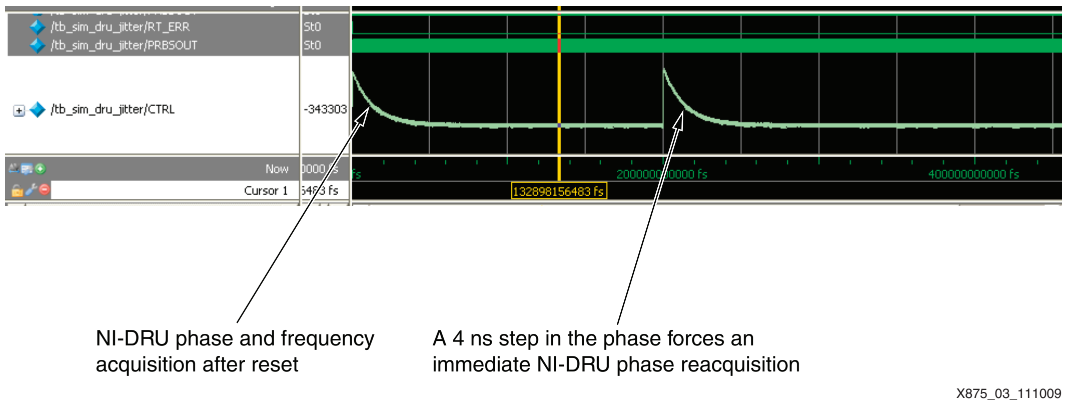

/tb_sim_dru_jitter/RT_ERR
/tb_sim_dru_jitter/PRBSOUT
/tb_sim_dru_jitter/CTRL
St0
St0
-343303
Now
Cursor 1
200000000000 fs
400000000000 fs
132898156483 fs
NI-DRU phase and frequency acquisition after reset
A 4 ns step in the phase forces an immediate NI-DRU phase reacquisition
X875_03_111009

Figure 3: **Example Simulation Output**

Testing the DRU on the ML523 Characterization Platform

An additional testbench, TB_HW_DRU, is available in the reference design (Figure 4, page 10) to show the NI-DRU capability of data recovery with both synchronous and asynchronous input data streams. This testbench includes four bidirectional STM1/OC3 channels (two GTX_DUAL tiles are used). Channel 0 and Channel 1 transmit data synchronized with REFCLK_0, while Channel 2 and Channel 3 transmit data synchronized with REFCLK_1. The GTX_DUAL tiles are cross connected via SMA cables so that each receiver receives data at a frequency not synchronized to its own reference clock.

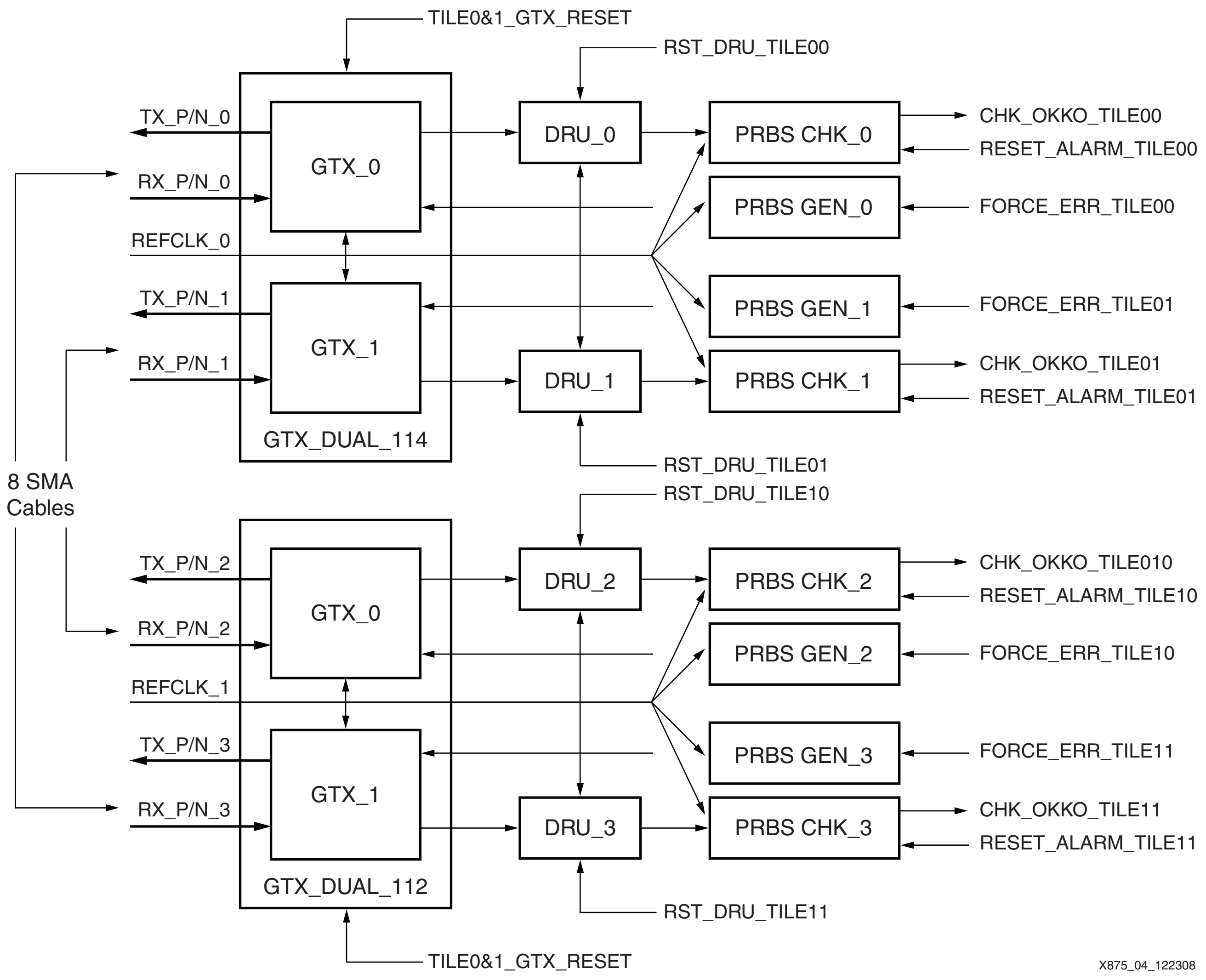


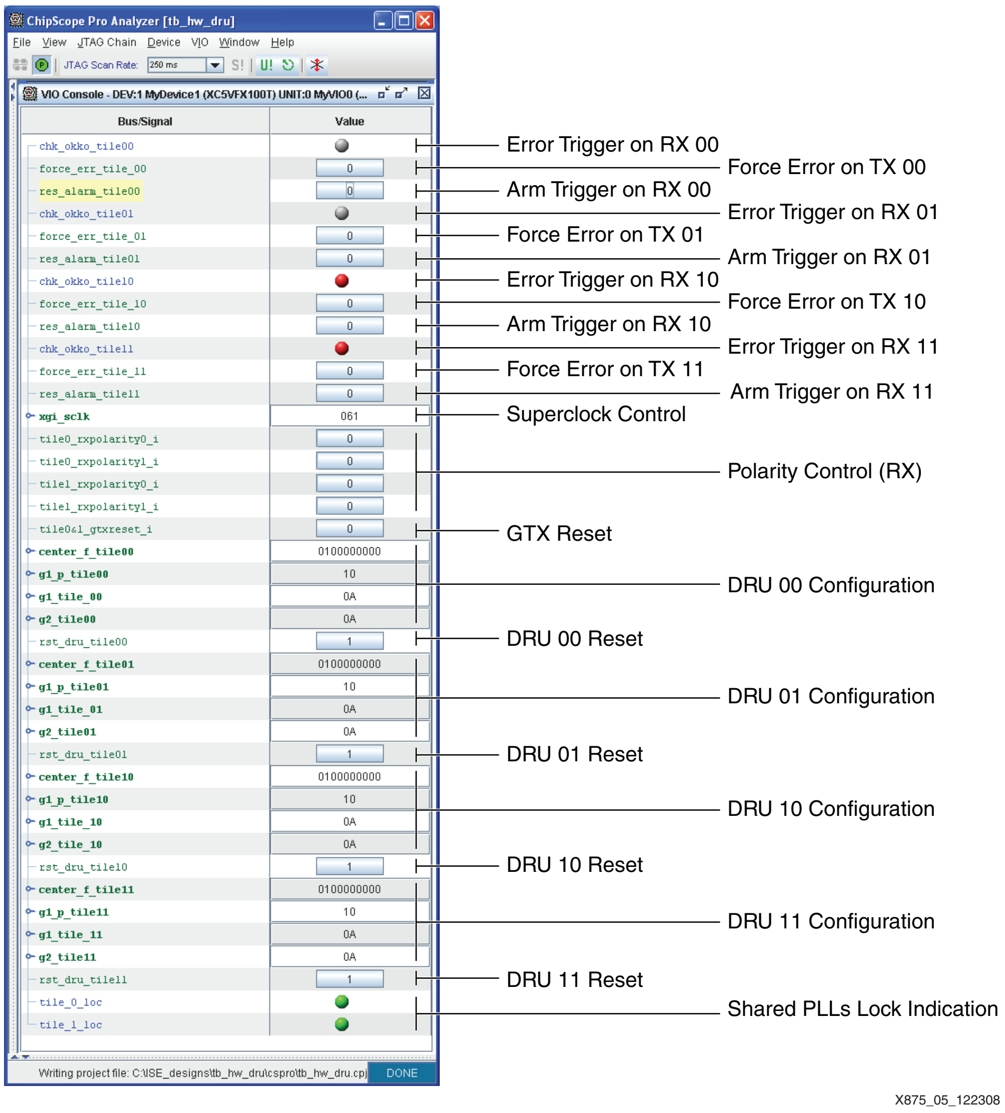
Figure 4: **Architecture of the TB_HW_DRU Testbench, Fully Controllable via ChipScope Pro Analyzer**

Each of the four channels is equipped with:

- A PRBS generator continuously sending a PRBS 15 pattern. The user can force each of the four PRBS generators to generate an error via the ChipScope™ Pro analyzer to show error detection on the corresponding PRBS checker.
- A PRBS checker continuously checking the incoming PRBS 15 pattern. The ERR output indicates detection of at least one error from the last ERR_RST. ERR can be armed with the ChipScope Pro analyzer via ERR_RST independently for each channel (to wait for the detection of the next bit error).

Note: The specific PRBS pattern used in this application note for both the generator and the checker is based on the polynomial $x^{15} + 1$. The VHDL and Verilog code for both the generator and the checker can be found in the reference design.

Each PRBS checker works on the data delivered by the barrel shifter, instantiated right after the DRU block. Figure 5, page 11 reports the detailed description of all pins of the testbench controlled by the ChipScope Pro analyzer. The pin names are consistent across the VHDL code, the Verilog code, the `.cpj` project, and this application note.



Bus/Signal	Value
chk_okko_tile00	●
force_err_tile_00	0
res_alarm_tile00	0
chk_okko_tile01	●
force_err_tile_01	0
res_alarm_tile01	0
chk_okko_tile10	🔴
force_err_tile_10	0
res_alarm_tile10	0
chk_okko_tile11	🔴
force_err_tile_11	0
res_alarm_tile11	0
xgi_sclk	061
tile0_rxpolarity0_i	0
tile0_rxpolarity1_i	0
tile1_rxpolarity0_i	0
tile1_rxpolarity1_i	0
tile0&1_gtxreset_i	0
center_f_tile00	0100000000
g1_p_tile00	10
g1_tile_00	0A
g2_tile00	0A
rst_dru_tile00	1
center_f_tile01	0100000000
g1_p_tile01	10
g1_tile_01	0A
g2_tile01	0A
rst_dru_tile01	1
center_f_tile10	0100000000
g1_p_tile10	10
g1_tile_10	0A
g2_tile_10	0A
rst_dru_tile10	1
center_f_tile11	0100000000
g1_p_tile11	10
g1_tile_11	0A
g2_tile11	0A
rst_dru_tile11	1
tile_0_loc	🟢
tile_1_loc	🟢

Writing project file: C:\ISE_designs\tb_hw_dru\cspro\tb_hw_dru.cpj DONE

Figure 5: **The ChipScope Pro Analyzer Controlling the Demonstration**

Resources

The NI-DRU is designed with efficient structures only (adders, multipliers, accumulators, and shifters). The NI-DRU block can be synthesized inferring one DSP, depending on the specific application. The resource requirements for Virtex-5 FPGAs are reported in Table 5.

Table 5: **Hardware Resources Required for the NI-DRU in Virtex-5 FPGAs**

Synthesis Type	Flip-Flops	LUTs	DSP48E Slices	BUFGs[(1)]	Virtex-5 FPGA Slices
With DSP	1048	1267	1	1	386
Without DSP	999	1488	0	1	442

Notes:

1. Only one BUFG is required even if many channels are being set up, and even if all are working at different data rates.
2. The resource requirements shown here do not include barrel shifters.
3. These results were obtained using ISE® software, version 10.1, SP3. The goal was timing performance, MAP was used with maximum compression enabled, and the CLK period was 6.4 ns.

Hardware Measurements

The NI-DRU has been tested with an Agilent J1408A OmniBER 725 communications performance analyzer to prove jitter tolerance compliance to the OC3/STM1 telecommunications optical standards using a FiberOn FTM-3128C-SL2G optical module and a Xilinx® SFP paddle card.

Figure 6 and Figure 7 show the bandwidth being doubled by changing the values of G_1 and G_2. G_{1_P} is kept constant at 16.

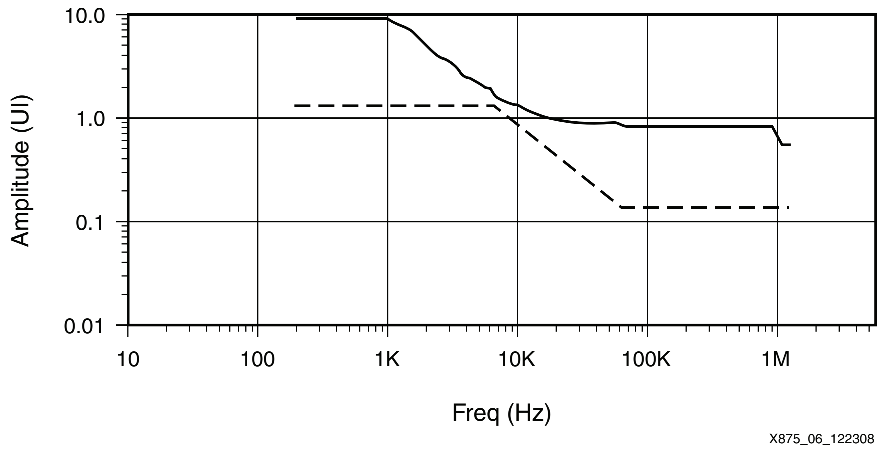


Figure 6: **Jitter Tolerance for OC3/STM1 with G_1 = 11 and G_2 = 11**

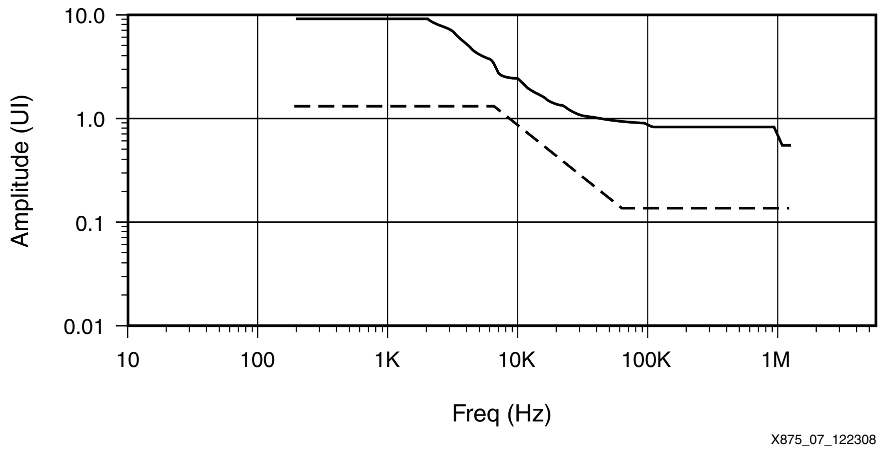


Figure 7: **Jitter Tolerance for OC3/STM1 with G_1 = 10 and G_2 = 10**

Hardware Requirements

The NI-DRU described in this application note can be implemented in the Virtex-5 LXT or FXT platforms. The example design in Testing the DRU on the ML523 Characterization Platform, page 9 targets the Virtex-5 device installed on the ML523 characterization platform.

The following hardware is required for the design:

- The ML523 characterization platform, revision C or higher
- A programmable USB cable, model DLC9 or newer
- A real-time scope to probe data and extracted clocks from the characterization platform

Software Requirements

The following software is required for the design:

- ChipScope Pro analyzer, version 9.1i or higher
- ISE software, version 9.1i or higher
- Mentor Graphics ModelSim, version 6.1a or higher (for simulation)

Reference Design

The reference design files (VHDL and Verilog) along with the testbenches can be downloaded at: https://secure.xilinx.com/webreg/clickthrough.do?cid=114961.

The reference design checklist is shown in Table 6.

Table 6: **Reference Design Checklist**

Parameter	Description
General	
Developer Name	Paolo Novellini and Giovanni Guasti
Target Devices (Stepping Level, ES, Production, Speed Grades)	Virtex-5 LXT/SXT/TXT/FXT platforms, production, all speed grades
Source Code Provided	Yes
Source Code Format	VHDL, Verilog
Design Uses Code/IP from Existing Application Note, Reference Designs, Third Party, or CORE Generator™ Software	Yes, CORE Generator software with: • GTX Transceiver Wizard, version 1.5 • ChipScope VIO core, version 1.02a • ChipScope ICON core, version 1.03a
Simulation	
Functional Simulation Performed	Yes
Timing Simulation Performed	No
Testbench Used for Functional and Timing Simulations	Yes
Testbench Format	VHDL and Verilog
Simulator Software/Version Used	ModelSim, version 6.2g
SPICE/IBIS Simulations	No
Implementation	
Synthesis Software Tools/Version Used	XST, version 10.1 SP3
Implementation Software Tools/Versions Used	• ISE software, version 10.1 SP3 • IPUPDATE, version 10.1.03
Static Timing Analysis Performed	Yes
Hardware Verification	
Hardware Verified	Yes
Hardware Platform Used for Verification	ML523 characterization platform, revision D

References

This application note uses the following references:

1. UG225, *ML52x User Guide.*
2. UG196, *Virtex-5 FPGA RocketIO GTP Transceiver User Guide.*
3. UG198, *Virtex-5 FPGA RocketIO GTX Transceiver User Guide.*
4. Best, Roland E. 1999. 4th ed. *Phase-Locked Loops: Design, Simulation, and Applications.* McGraw-Hill Professional Publishing.
5. Gardner, Floyd M. 2005. 3rd ed. *Phaselock Techniques.* Wiley-Interscience.
6. XAPP868, *Clock Data Recovery Design Techniques for E1/T1 Based on Direct Digital Synthesis.*
7. DS202, *Virtex-5 FPGA Data Sheet: DC and Switching Characteristics.*
8. IEEE 802.3 Ethernet standard: http://www.ieee802.org/3/.

Acknowledgement

Xilinx wishes to thank Silvio Cucchi for his key contributions to this reference design and document.

Revision History

The following table shows the revision history for this document.

Date	Version	Description of Revisions
03/09/09	1.0	Initial Xilinx release.
01/13/10	1.1	Revised the theoretical high-frequency jitter tolerance discussion for Equation 9. Revised Selecting G1 and G1_P, page 7 and Figure 3, page 9.

Notice of Disclaimer