XAPP861 (v1.1) July 20, 2007

Efficient 8X Oversampling Asynchronous Serial Data Recovery Using IDELAY

Author: John F. Snow

Summary

Asynchronous serial data interfaces require the receiver to recover the data by examining the bitstream and determining where each bit should be sampled in the absence of a clock sent with the data. There are several ways to implement this type of receiver in Xilinx FPGAs. RocketIO™ transceivers are designed specifically for this task but are not available in all Xilinx FPGAs. Depending on the device family and speed grade, SelectIO™ inputs and FPGA logic resources can implement asynchronous serial receivers at bit rates up to nearly 1 Gb/s.

For data with short run lengths and low jitter, sampling the bitstream about four times per bit period can be sufficient. Such a low oversampling rate data recovery technique is described in XAPP224, *Data Recovery*. However, when the data has long run lengths without bit transitions or high jitter tolerance is required, higher oversampling rates are required, traditionally requiring the use of multiple clock phases that often consumes several digital clock managers (DCMs) and many global clock resources.

Xilinx Virtex™-4 and Virtex-5 devices a have high-precision programmable delay element associated with every input pin. These delay elements, called IDELAY, can be used to implement an oversampler that uses very few FPGA logic resources and, more importantly, just a single DCM and two global clock resources to do 8X oversampling. This solution provides better jitter tolerance than techniques that use multiple DCMs.

When paired with a suitable data recovery scheme, the oversampling technique described here can be used with many different data protocols up to 550 Mb/s in Virtex-5 devices and 500 Mb/s in Virtex-4 devices. As an example of how to use the technique, a reference design implements a SD-SDI (SMPTE 259M) receiver running at 270 Mb/s.

Oversampling Techniques

Asynchronous serial data recovery techniques using oversampling require the receiver to sample the incoming bitstream fast enough to capture multiple samples during each bit period. This sampling provides enough information about the bitstream so that the locations of the transitions can be determined and each bit can be sampled close to the center of the bit and away from the transition regions.

Figure 1 shows a typical oversampling data receiver for asynchronous serial data. There are two main elements to the receiver: the oversampler and the data recovery unit (DRU). The oversampler samples the bitstream multiple times per bit period, capturing enough data to feed into the DRU. The DRU must have sufficient samples per bit period so that it can determine where bit transitions occur and then sample each bit a safe distance from the transitions. Typically, the more samples per bit period that the oversampler can provide to the DRU, the more precisely the DRU can identify where bit transitions are occurring and more accurately recover the data. Having more samples per bit period increases the jitter tolerance of the DRU.

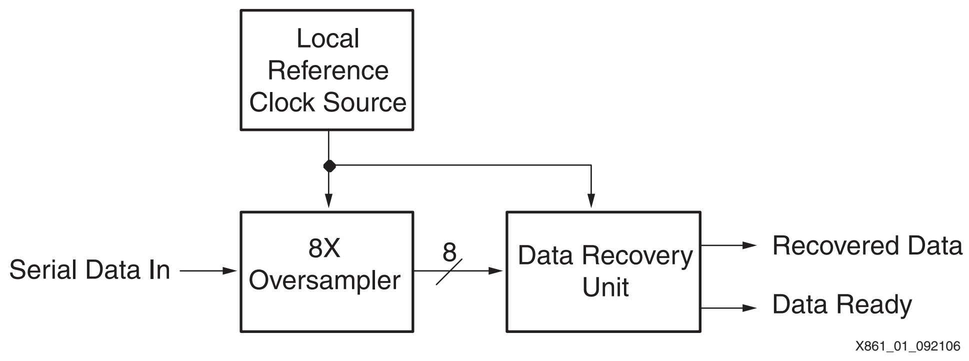

Figure 1: **Basic Diagram of an Oversampling Data Recovery Solution**

There are many ways to implement the oversampler in Xilinx FPGAs. RocketIO transceivers can be used as oversamplers, providing oversampling of bit rates of 1 Gb/s or higher in some cases. In fact, some RocketIO transceivers have a built-in oversampling receiver, called the digital receiver, that supports bit rates below those supported by the normal receiver.

When using SelectIO inputs to receive asynchronous serial bitstreams, the maximum clock rate of the FPGA, not the maximum rate supported by the IOB, typically limits the maximum bit rate that can be supported. Because the oversampler must sample the bitstream multiple times per bit period, the sampling clock must be significantly faster than the bit rate. Using multiple clock phases of the sampling clock can increase the number of samples per bit period without increasing the clock frequency, but this approach requires a method of precisely generating and distributing multiple clock phases, and it uses up multiple global clock routing resources. For example, using four clock phases, each 45 degrees apart, and using both edges of each of the four clock phases allows 8X oversampling with a basic clock frequency that is the same as the bit rate.

DCMs can be used to create multiple clock phases to oversample a serial input (see Figure 2, based on XAPP224). However, depending on the clock frequency and the number of samples per bit period required, multiple DCMs might be required to provide enough clock phases. When multiple DCMs are used, the outputs of these DCMs jitter relative to each other, decreasing the precision of the sampling points and thereby decreasing the jitter tolerance of the receiver. The scheme used in Figure 2 also requires four global clocks for the four clock phases. However, these four global clocks can be used to support multiple oversamplers if the bit rates are all the same.

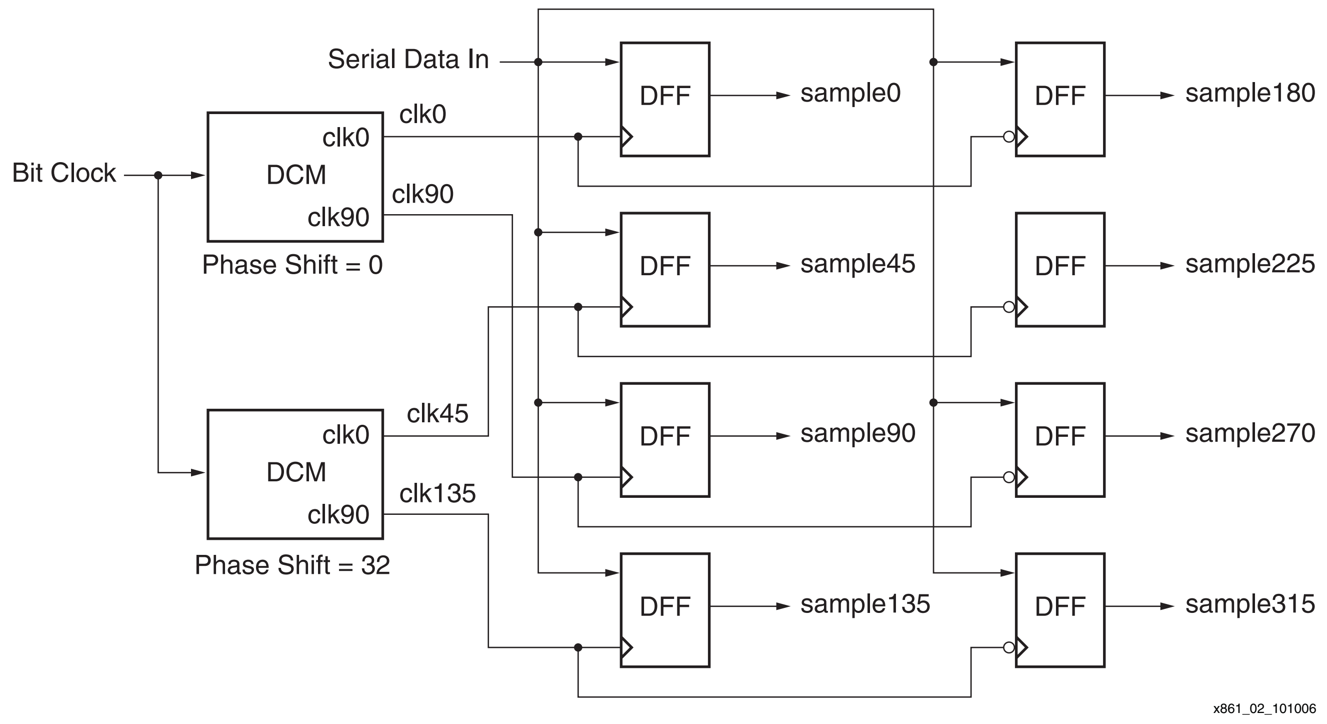

Figure 2: **8X Data Recovery Scheme Using Two DCMs**

Xilinx has experimented with multiple clock phase oversamplers for SD-SDI, a protocol with long run lengths between bit transitions and high jitter tolerance requirements due to the long cable lengths supported. These experiments have shown that any oversampling less than 6X of an SD-SDI bitstream does not provide sufficient jitter tolerance, and 8X oversampling is a more realistic minimum. An 8X oversampling scheme using four clock phases requires two DCMs, as shown in Figure 2. Experiments at Xilinx showed that the relative jitter on the clock phases produced by these two DCMs reduced the jitter tolerance of the DRU to an unacceptable level. If a single DCM can be used, then clock phase jitter problem can be reduced, but four global clocks would still be required. A single Virtex-5 PLL can generate the four clock phases with low jitter, but four global clocks are still required to route these four clock phases to the oversampler. Virtex-4 devices do not have such a PLL.

Using IDELAY for Efficient Oversampling

Figure 3 shows the IDELAY-based oversampler. A single DCM is used to create two clock phases 90 degrees apart. Using both edges of these two clock phases gives four equally spaced sampling points during one bit period. The serial bitstream enters the FPGA through a primitive called IBUFDS_DIFF_OUT. Because this primitive only works with LVDS inputs, the serial data must be LVDS. The normal LVDS input primitives provide a single signal to the FPGA, but the IBUFDS_DIFF_OUT primitive provides both true and inverted copies of the received signal to the FPGA. As shown in Figure 4, IBUFDS_DIFF_OUT uses two differential receivers, one in the P input IOB and the other in the N input IOB of the LVDS pair. Each of these differential receivers provides a separate signal to the FPGA with the signal from the N-side IOB inverted from that provided by the P-side IOB.

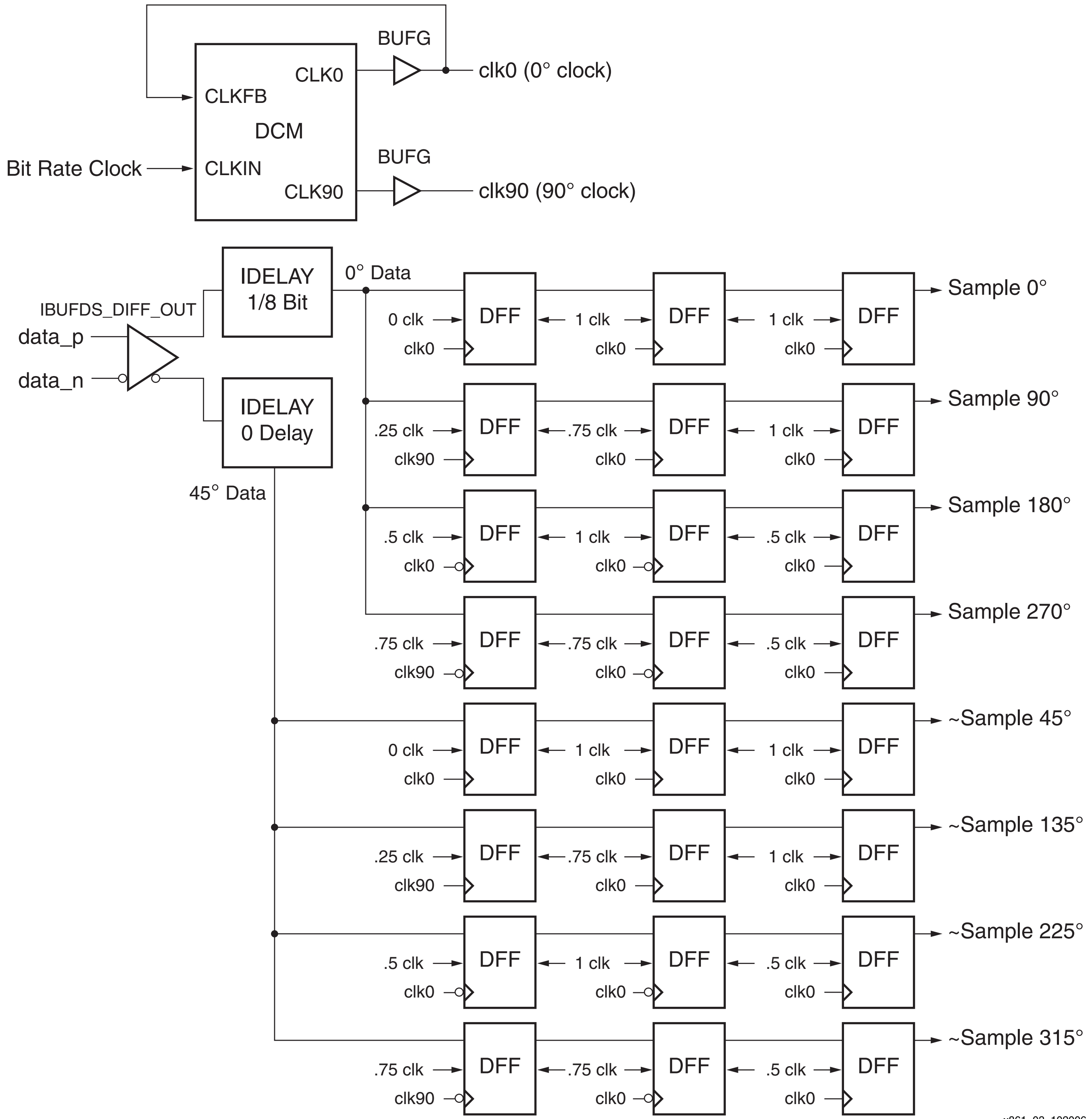

Figure 3: **IDELAY-Based Oversampler**

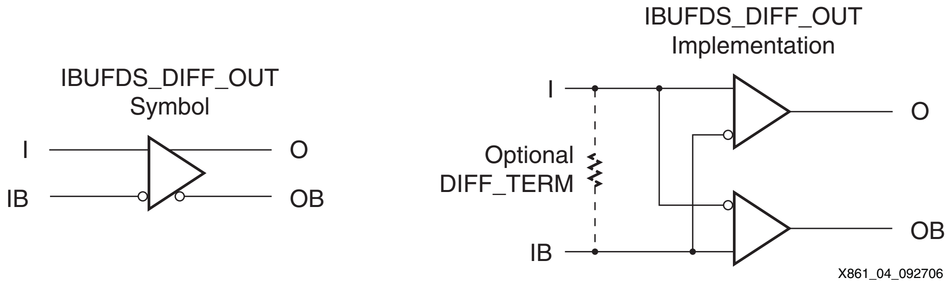

Figure 4: **IBUFDS_DIFF_OUT Primitive**

Each IOB in Virtex-4 and Virtex-5 devices has a single IDELAY primitive associated with it. However, with the IBUFDS_DIFF_OUT buffer providing two copies of the input signal from the two different IOBs of the LVDS pair, these two copies of the bitstream can be sent through separate IDELAY primitives, delaying them by different amounts. The true copy of the bitstream from the P-side of the IBUFDS_DIFF_OUT is delayed by its associated IDELAY by one-eighth of a bit period. The IDELAY for the inverted output of the IBUFDS_DIFF_OUT is set to zero delay. These two IDELAY blocks provide two different "phases" of bitstream. By sampling each of these two bitstream phases with both edges of the two clock phases, the oversampler captures the bitstream eight times per bit period as shown in Figure 5.

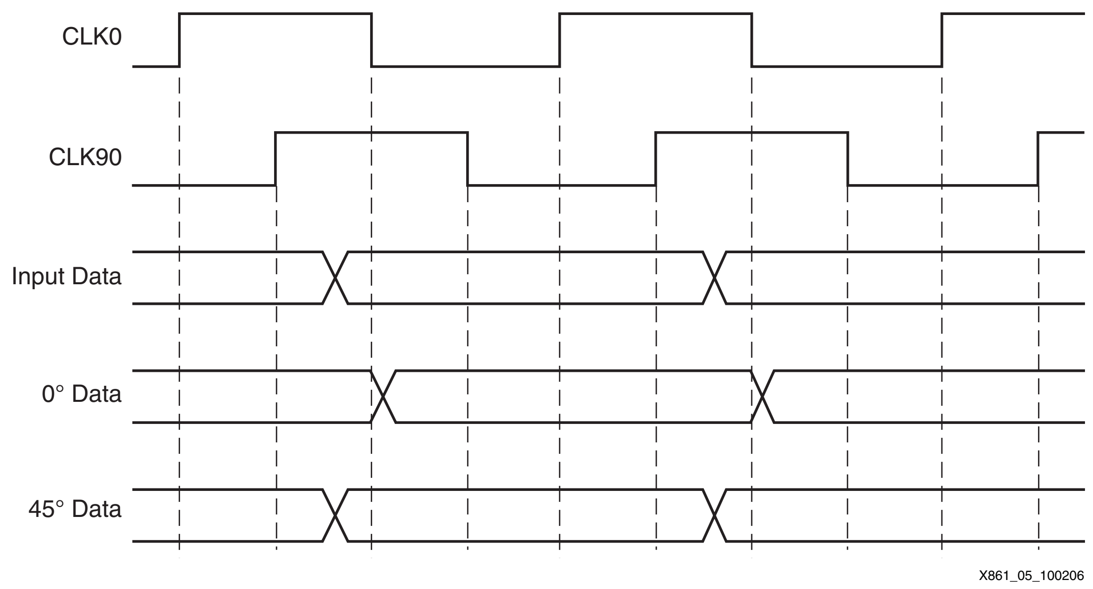

Figure 5: **Timing Diagram**

At first glance, it seems odd that the data labeled as having zero phase shift delay is actually coming from the IDELAY block with one-eighth of the bit period of delay, while the data labeled as having 45 degrees of phase shift comes from the IDELAY block with zero delay. However, this is correct. CLK0 in Figure 5 illustrates this example. Both phases of the bitstream are sampled on the rising clock edge of CLK0. The sample captured from the IDELAY with the one-eighth bit period delay represents the input data one-eighth of a bit time *earlier* than the sample captured from the IDELAY with zero delay.

It might also seem odd to use two IDELAYs when one of them is set to zero delay. However, this ensures that both copies of the signal use equivalent routing resources, minimizing the skew between the two phases of the data signal.

As shown in Figure 3, page 4, each sample is moved into the main clock domain (rising edge of the zero degree clock) through a pipeline of three flip-flops. The first flip-flop is the sampling flip-flop, which occasionally enters the metastable condition because the bitstream is being sampled asynchronously. To get around problems caused by metastability, the design allows at least three-quarters of a clock cycle between the first flip-flop and the second flip-flop in the pipeline to give the first flip-flop time to settle before the sample is clocked into the second flip-flop. By the time the sample is clocked into the third flip-flop of the pipeline, the sample has been moved into the main clock domain and metastability has been resolved.

The total delay through the pipeline for any sample is exactly two clock cycles, if the amount of delay on the first flip-flop in the pipeline is included. For example, with the 315 degree sample, the initial flip-flop is clocked three-quarters of the way through the clock cycle because the flip-flop uses the falling edge of the 90 degree clock. The delay from the first flip-flop to the second flip-flop is three-quarters of a clock cycle, and the delay from the second flip-flop to the third flip-flop is half a clock cycle. The total adds up to exactly two clock cycles.

This IDELAY-based oversampling technique has advantages both in performance and in resource usage over the dual-DCM technique shown in Figure 2, page 3. Obviously, this technique uses one less DCM and two less global clocks. Because only a single DCM is used, the clock phases are more stable relative to each other. Thus, this oversampling technique provides more accurate repetitive sampling of the bitstream, increasing the jitter tolerance of the receiver.

The IDELAY primitives provide delays of about 78 ps per tap. A delay value of 6 provides a delay of 468 ps, very close to one-eighth of a 270 Mb/s bit period (463 ps). 270 Mb/s is a common bit rate in video applications and is used as the example bit rate in the reference design.

If multiple asynchronous bitstreams of the same bit rate are being received by the FPGA, the two global clock phases provided by the DCM can be used to clock the oversamplers for all the bitstreams. Thus, this technique is very efficient for receiving multiple bitstreams of the same bit rate.

The DCM in Figure 3, page 4 cannot simultaneously do clock multiplication and provide the two clock phases, 90 degrees apart. Thus, a bit-rate clock is required on the input of the DCM. However, the PLL present in the clock management tiles of Virtex-5 devices can simultaneously multiply the incoming clock and provide the two different clock phases, allowing a lower frequency, input reference clock to be used.

Figure 6 shows the IDELAY-based oversampler design connected to a DRU. This DRU supports 8X oversampling and was initially developed for use with a RocketIO transceiver as the oversampler. Because of this, it expects a 20-bit input vector, so a small amount of gasket logic is required to interface the 8-bit output of the oversampler to the 20-bit input of the DRU. The DRU produces 10-bit recovered data words on its output port along with a signal that is asserted whenever the data word is valid. The DRU can also produce 20-bit output words.

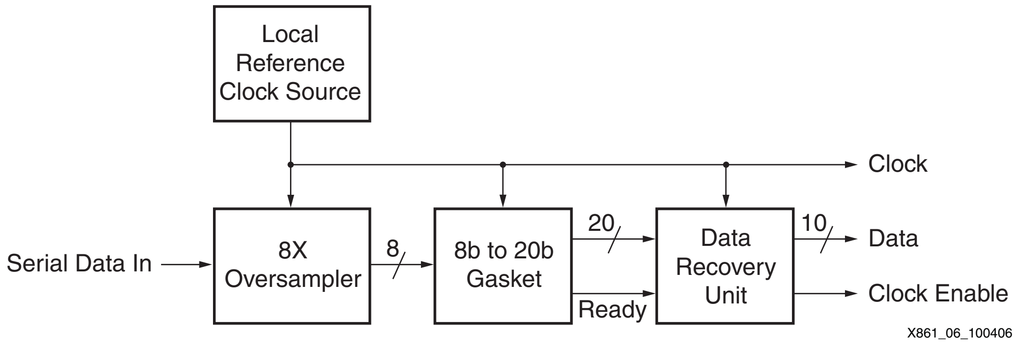

Figure 6: **IDELAY-Based Oversampler with DRU**

Reference Design

The reference design files can be downloaded from http://www.xilinx.com/bvdocs/appnotes/xapp861.zip.

The oversampler module is called oversample_8x. Its ports are shown in Table 1.

Table 1: **oversample_8x Module Ports**

Port Name	I/O	Width	Description
clk0	In	1	Bit rate clock
clk90	In	1	Bit rate clock delayed 90 degrees from clk0
PAD_din_p	In	1	P input of serial data LVDS pair
PAD_din_n	In	1	N input of serial data LVDS pair

Table 1: **oversample_8x Module Ports *(Continued)***

Port Name	I/O	Width	Description
dout_rdy	Out	1	Asserted High for one clk0 cycle when data is ready on dout
dout	Out	20	Oversampled data output

The oversample_8x module also accepts one Verilog parameter or VHDL generic called IDELAY_45_DEGREES, which defaults to 6. This value sets the IDELAY amount to obtain a 45 degree shift on the data (one-eighth of a bit period). The default value of 6 is the correct IDELAY value for 270 Mb/s (78 ps * 6 ≈ (1 / 270 MHz) / 8). If the bit rate is something other than 270 Mb/s, the correct value for IDELAY_45_DEGREES must be passed to the module.

The IBUFDS_DIFF_OUT input buffer is instantiated inside the oversample_8x module with an IOSTANDARD of “LVDS_25” with the internal differential termination (DIFF_TERM) turned on. If the application requires a different IOSTANDARD or requires the differential termination to be turned off, then the constraints placed on the IBUFDS_DIFF_OUT buffer in the oversample_8x module must be edited. IBUFDS_DIFF_OUT can only be used with LVDS I/O standards.

The DRU module is called os48_1011x20bTo10b_top2. Its ports are shown in Table 2. All port names beginning with A/B are actually two ports, one for the A channel and one for the B channel.

Table 2: **os48_1011x20bTo10b_top2 Module Ports**

Port Name	I/O	Width	Description
rst	In	1	Asynchronous reset
bitOrderDinLSB	In	1	Specifies the bit order of the data into the DRU (1 = LSB first)
bitOrderDoutLSB	In	1	Specifies the bit order of the data output from the DRU (1 = LSB first)
A/B_recclk	In	1	Bit rate clock
A/B_en	In	1	Clock enable. Must be asserted High for one cycle of A/B_recclk when data is available on the A/B_din20b port
A/B_mode	In	2	Specifies the oversampling factor (“01” specifies 8X oversampling)
A/B_din20b	In	20	Oversampled input data
A/B_dout10bEn	Out	1	Asserted High for one cycle of A/B_recclk when data is ready on A/B_dout10b
A/B_dout10b	Out	10	10-bit recovered data output port
A/B_dout20bEn	Out	1	Asserted High for one cycle of A/B_recclk when data is ready on A/B_dout20b
A/B_dout20b	Out	20	20-bit recovered data output port

The DRU’s A_recclk and B_recclk ports must be driven by the same bit rate clocks connected to the clk0 input port of the respective oversample_8x modules. The oversampler’s dout and dout_rdy ports drive the A/B_din20b and A/B_en ports of the DRU, respectively.

It is essential that the bit order of the signals be specified correctly to the DRU. The DRU has two bit order input ports, one specifies the bit order of the input data to the DRU and the other specifies the bit order of the data output from the DRU. The bit order of the two channels of the DRU cannot be specified independently. The oversample_8x module always captures data LSB first, so the bitOrderDinLSB of the DRU should always be set to 1. The bitOrderDoutLSB must be set as required by the application. If the application uses a serial protocol where the LSB is sent first, bitOrderDoutLSB should be set to 1 so that the first recovered data bit is output on the LSB of the output data ports. If the serial protocol is MSB first, bitOrderDoutLSB should be set to 0 and the first recovered data bit is output on the MSB of the output data ports.

Both 10-bit and 20-bit output ports are provided by the DRU. Both ports are always active and the application can choose to use either one. The 10-bit and 20-bit ports of each channel have separate output enables typically used as clock enables to downstream logic. In typical applications, the bit rate reference clock that drives the clk0 port of the oversample_8x and the A/B_recclk port of the DRU is used as the clock for downstream logic in the receiver, such as decoders, framers, error checkers, etc. The clock enable outputs of the DRU are used to enable the downstream logic only on those clock cycles when the DRU has a new word of recovered data available on its output ports.

Module FPGA Resource Usage

Table 3 shows the FPGA resources required to implement the complete 8X oversampling unit plus data recovery unit. The DRU is block RAM based. Because the block RAMs in Xilinx FPGAs are dual-ported, the DRU design can handle two asynchronous data streams. Thus, the same number of block RAMs can handle either one or two asynchronous channels. In Virtex-4 devices, two RAMB16 block RAMs are required for the DRU, regardless of whether one or two channels are processed by the DRU. In Virtex-5 devices, the DRU requires one and a half RAMB36 block RAMs (one RAMB18 and one RAMB36).

The results in Table 3 were obtained with ISE8.2 with XST set to optimize for area and with a serial bit rate of 270 Mb/s. In a slowest speed grade of both Virtex-4 and Virtex-5 FPGAs, it was possible to meet timing for a 270 Mb/s bit rate with XST set to optimize for area.

Table 3: **FPGA Resource Usage for 8X Oversampling DRU**

Family	Modules	FFs	LUTs	Block RAMs	DCM
Virtex-4 FPGAs	One oversampler + DRU	224	149	2 RAMB16	1
Virtex-4 FPGAs	Two oversamplers + DRU	448	297	2 RAMB16	1
Virtex-5 FPGAs	One oversampler + DRU	224	101	1.5 RAMB36	1
Virtex-5 FPGAs	Two oversamplers + DRU	448	201	1.5 RAMB36	1

Timing

The granularity of the IDELAY primitive is 78 ps. Thus, some bit rates are better suited to this technique than others because the 78 ps granularity matches well with the one-eighth bit period delay required by this technique. For example, at 270 Mb/s, one-eighth of a bit period is about 463 ps and 6 * 78 ps is 468 ps. Thus, using a delay of 6 causes the IDELAY to delay the signal by very nearly one-eighth of a bit period.

The CLKFX outputs of the DCM must *not* be used with this technique. It is important that the clock phases have the lowest jitter possible and the DCM's CLKFX outputs have much higher jitter than the CLK0 and CLK90 outputs. In Virtex-5 devices, the PLL in the clock management tile can also be used to provide the two clock phases.

It is important to properly constrain the timing of the DRU and downstream logic. This typically requires multi-cycle timing constraints because the clock used is a bit rate clock, but DRU and downstream logic typically run at the word rate by using clock enables. Two different clock enables are used and must be constrained properly.

The oversampler produces a clock enable (dout_rdy) to the A_en or B_en input of the DRU so that the DRU is only clocked when 20 bits of oversampled data are ready to be input to the DRU. If the instance name of the oversampler is OVRSAMPLE, then the following constraints are used to indicate that the DRU input is only clocked once every 20 cycles of the clock called *sclk*:

```
NET “sclk” TNM_NET = BIT_CLK;
```

```
TIMESPEC TS_BIT_CLK = PERIOD BIT_CLK 270 MHz HIGH 50 %;
NET "OVRSAMPLE/period<0>" TNM = DRU_IN_CE;
TIMESPEC TS_DRU_IN_CE = FROM DRU_IN_CE TO DRU_IN_CE BIT_CLK / 20;
```

The PERIOD constraint for the serial clock is given in frequency, thus the TIMESPEC for the clock enable is equal to the TIMESPEC for the serial clock divided by 20. If the PERIOD constraint for the serial clock is given as a period instead of frequency, the TIMESPEC for the clock enable needs to be equal to the TIMESPEC for the serial clock multiplied by 20.

The other clock enable is produced by the DRU on its enable outputs. The DRU has four output enables, two per channel. Of the two output enables per channel, one corresponds to the 10-bit output and the other corresponds to the 20-bit output. The 10-bit output port enable (A_dout10bEn or B_dout10bEn) is asserted, on average, once every 10 cycles of the serial clock. However, because the actual data bit rate is asynchronous to the serial clock, occasionally there are 9 or 11 clock cycles between assertions of the 10-bit output enable. In fact, the DRU can also produce data words with as few as 5 clock cycles between assertions of the 10-bit output enable. Thus, the multi-cycle timing constraint for the logic driven by the DRU's clock enable output should be written with the minimum timing in mind. If the DRU's 10-bit clock enable output is used and the signal name is *rx_ce*, the following timing constraints properly constrain any logic downstream from the DRU enabled by the rx_ce signal. It is necessary to apply a KEEP constraint to the clock enable signal from the DRU in the Verilog or VHDL code in order that the clock enable retain the name it is given in the source code so that the same name can be used in the user constraints file (UCF).

```
NET "rx_ce" TNM = RXCE;
TIMESPEC TS_RXCE = FROM RXCE TO RXCE BIT_CLK / 5;
```

Placement Constraints

To get good performance from the oversampler, it is important that the initial set of eight flip-flops (those flip-flops that capture the data directly from the outputs of the IDELAY primitives) be placed as near as possible to the IDELAY primitives and that routing from the two IDELAY primitives to the eight flip-flops be as identical as possible for all eight flip-flops. The oversampler module is designed so that an RLOC origin can be placed in the constraints file to constrain this initial set of eight flip-flops. The RLOC origin should be set so that the flip-flops are located in the CLB immediately to the left of the IDELAY primitives, if possible, because this placement provides the most direct routing of the signals. If it is not possible to locate the flip-flops to the left of the IDELAY primitives, they can be placed immediately to the right with some small degradation in performance.

Figure 7 shows a Virtex-4 device with the flip-flops located immediately to the left of the IDELAY primitives. In this figure, the routing of the data signals from the two IDELAY primitives to the flip-flops is shown in red.

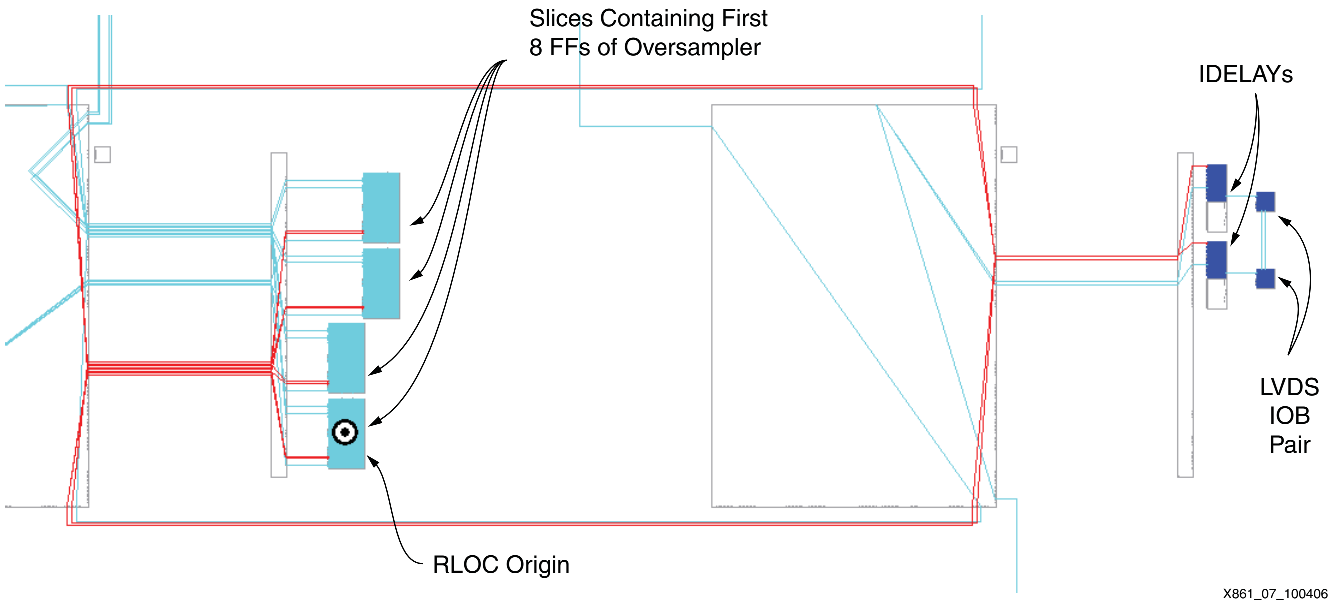

Figure 7: **Flip-Flop Location and Signal Routing for Virtex-4 FPGAs**

In Virtex-5 devices, the relative sizes of the CLB versus a pair of IOBs is different because a CLB contains more logic in Virtex-5 devices. So the placement and routing are somewhat different, as shown in Figure 8.

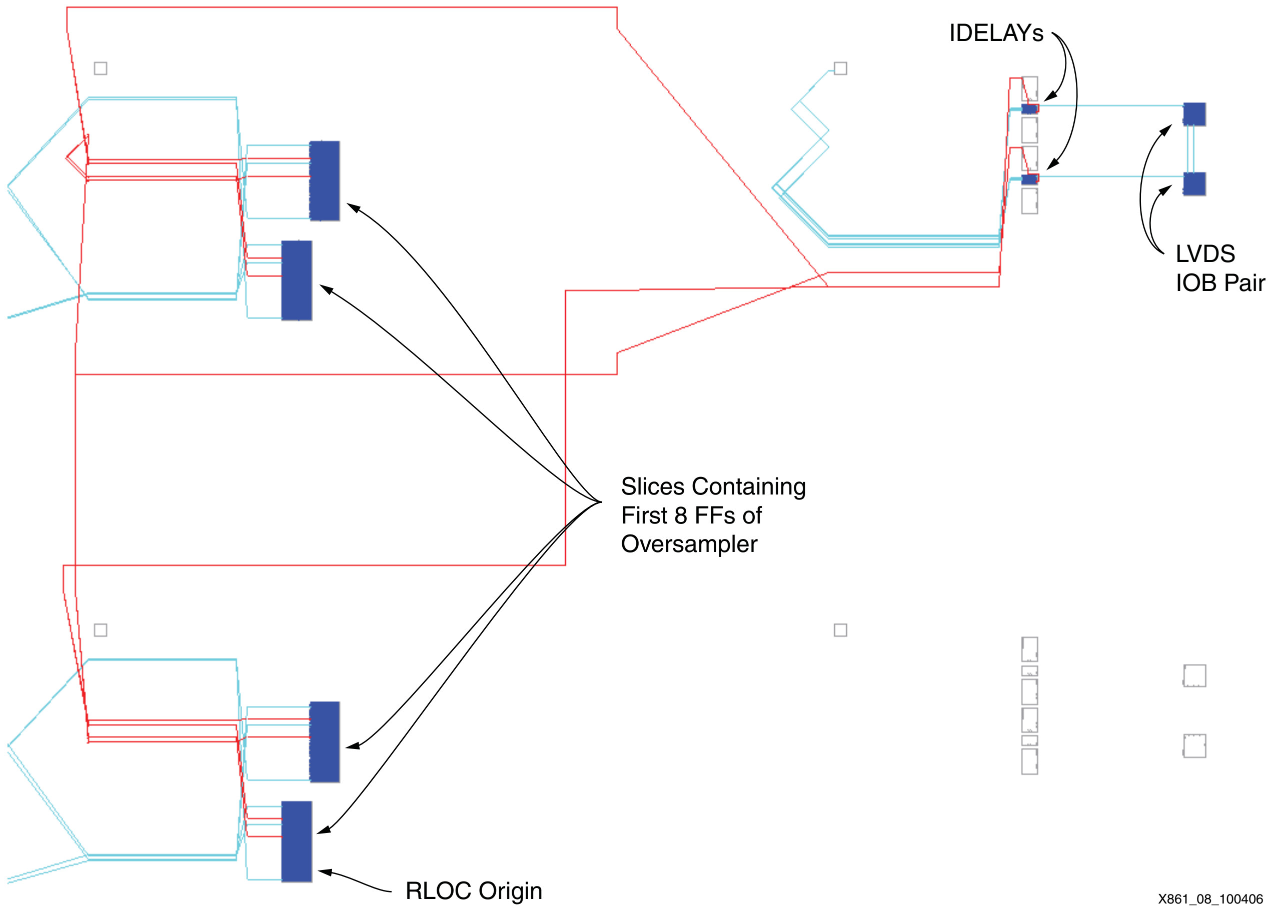

Figure 8: **Flip-Flop Location and Signal Routing for Virtex-5 FPGAs**

The following constraints show how to assign each of two oversample_8x modules in a design to different U_SETs and assign RLOC_ORIGIN constraints to them to place them next to their respective IOBs. In this example, the instance names of the two oversample_8x modules are OVRSAMPLE1 and OVRSAMPLE2.

```
INST "OVRSAMPLE1" U_SET=USET_OVR1;
INST "OVRSAMPLE1" RLOC_ORIGIN = X0Y24;
INST "OVRSAMPLE2" U_SET=USET_OVR2;
INST "OVRSAMPLE2" RLOC_ORIGIN = X50Y24;
```

Conclusion

The technique described here uses the IDELAY resources built into every IOB of Virtex-4 and Virtex-5 devices to implement an efficient and high performance 8X oversampler for asynchronous serial bitstreams. It can provide better sampling accuracy than a multiple clock phase approach when those clock phases are generated by multiple DCMs. One DCM and two global clocks provide all the clocks necessary to do 8X oversampling of any number of asynchronous bitstreams running at the same bit rate because the IDELAY resources can provide the additional sample points for the individual bitstreams.

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
10/13/06	1.0	Initial Xilinx release.
07/20/07	1.1	Minor updates to IDELAY-based oversampling technique.