XILINX®

XAPP881 (v1.1) September 24, 2014

# Virtex-6 FPGA LVDS 4X Asynchronous Oversampling at 1.25 Gb/s

Authors: Catalin Baetoniu and Brandon Day

## Summary

The Virtex®-6 FPGA SelectIO™ technology can perform 4X asynchronous oversampling at 1.25 Gb/s. The oversampling is accomplished using the ISERDESE1 primitive through the mixed-mode clock manager (MMCM) dedicated performance path. The ISERDESE1 is located in the SelectIO logic block and contains four phases of dedicated flip-flops used for sampling. The MMCM is an advanced PLL that has the capability to provide a phase-shifted clock on a low-jitter performance path.

The output of the ISERDESE1 is then sorted using a data recovery unit (DRU). The DRU used in this application note is based on a voter system that compares the outputs and selects the best data sample.

## Asynchronous Oversampling Overview

The most common way of sending data from one device to another is to send synchronized clock and data together. This approach is known as source-synchronous data. When data is sent without a clock, it is known as asynchronous data.

An asynchronous data capture scheme is the focus of this application note. The method used consists of oversampling the data with a clock of similar frequency (±100 ppm). This oversampling technique involves taking multiple samples of the data at different clock phases to get a sample of the data at the most ideal point. Figure 1 is a basic diagram of 4X asynchronous oversampling.

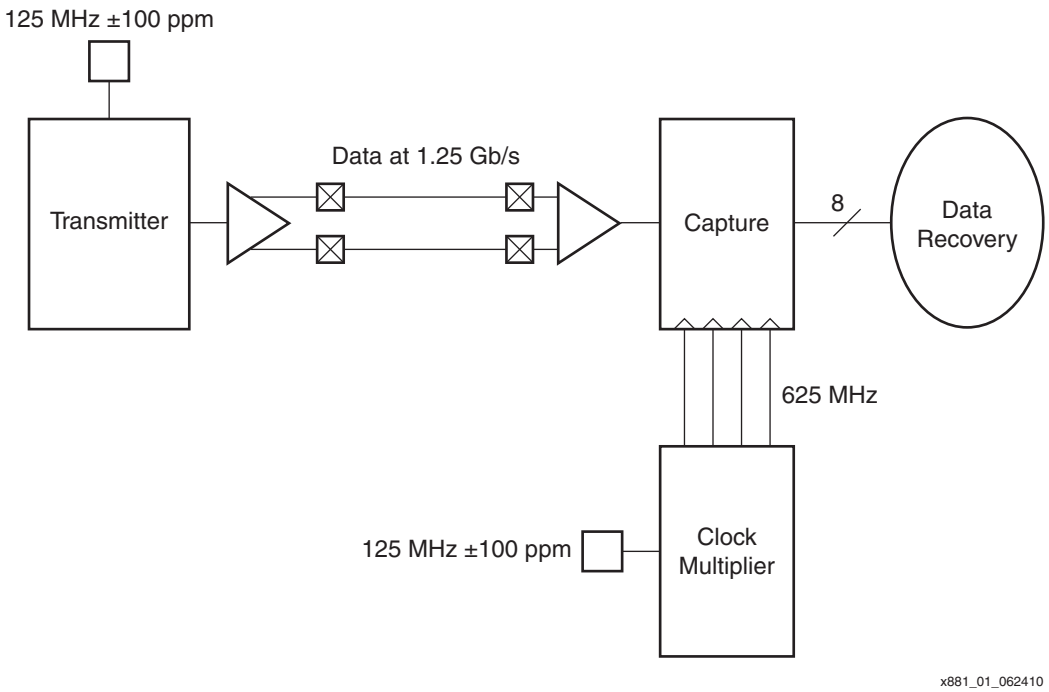


*Figure 1:* **Basic Architecture and Conceptual View of 4X Oversampling**

Figure 1 shows some of the key elements of the 4X oversampling technique. The transmitted data is generated using an oscillator with a particular frequency and drift, and the data is then captured using an independent, asynchronous oscillator. An inherent challenge in such a scenario is the frequency drift differential between the sending and receiving oscillators, which must be taken into account when performing data capture across clock domains. The data sampled is then processed in the data recovery portion of the receiver using a voter system.

Figure 2 shows the sampling clock running at one-half the rate of the data. With the clock phase sample points shown, this produces eight samples of data coming out of the capture block: four for the rising-edge data and four for the falling-edge data.

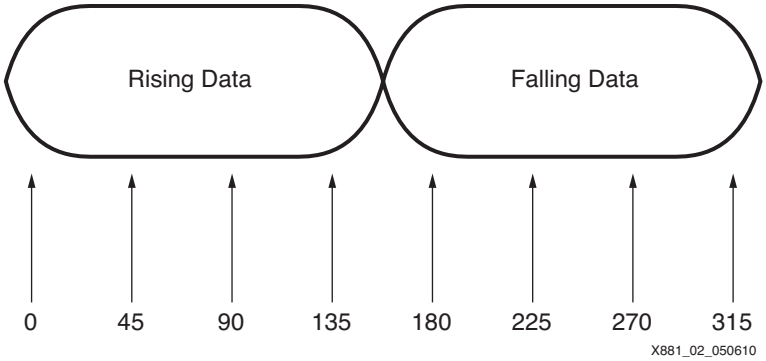


*Figure 2:* **Phase of Sampling Clocks for Rising and Falling Edge Data**

## Virtex-6 FPGA Oversampling Architecture

The Virtex-6 FPGA method of creating four samples per bit has advantages over the methods used previously, where flip-flops in the CLB arrays were used to create multiple samples. The sampling flip-flops are moved from the CLB to inside the ISERDESE1, as shown in Figure 3.

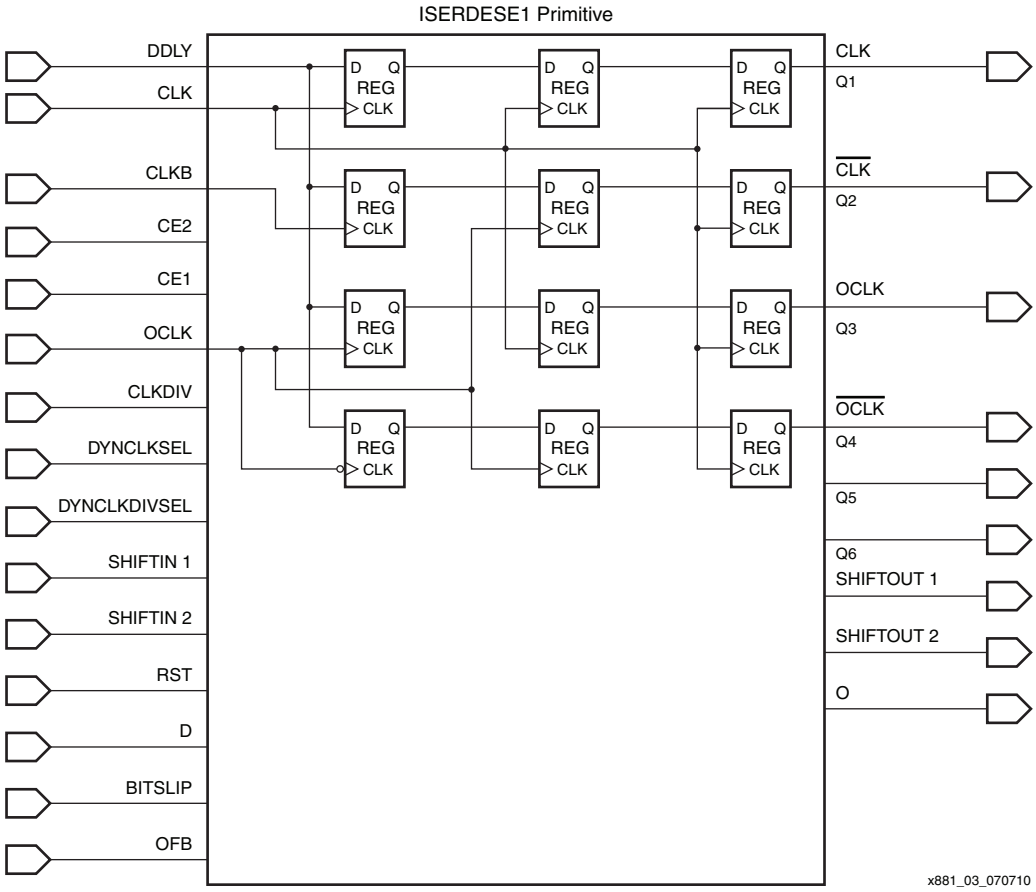


*Figure 3:* **View of ISERDESE1 Primitive in Oversample Mode**

The staged sampling inside the ISERDESE1 is to counter any metastability issues. This also allows all of the data to come out of the ISERDESE1 on one clock domain.

The ISERDESE1 in oversampling mode is essentially a dual set of DDR flip-flops, as shown in Figure 4. The IODELAYE1 is used to create a phase-shifted version of the data on the slave side of the ISERDESE1 that can be sampled along with the unshifted original version.

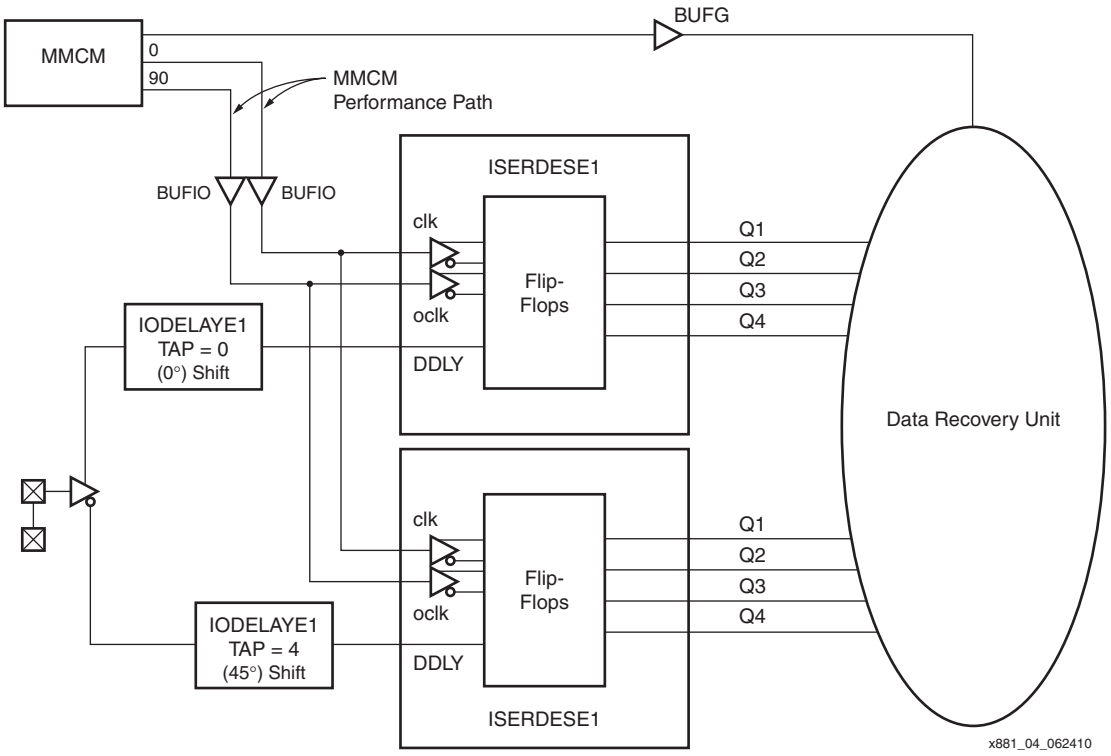


*Figure 4:* **Method Used in Virtex-6 FPGA to Create Four Samples**

Figure 4 shows in detail how the MMCM, IODELAYE1, and ISERDESE1 can be combined to create 4x oversampling. Two clock phases (CLK0 and CLK90) are sent to the ISERDESE1; both the positive- and negative-going edges of the two clocks are used, effectively creating four clock phases. In addition, IBUFDS_DIFFOUT is used to create two copies of the data, and IODELAYE1 is used to shift one copy by 200 ps (that is, 45° of phase shift). By sending a phase-shifted version of the data to the slave ISERDESE1, the number of phases of the samples is effectively doubled. In all, therefore, eight phases are created by using four clock phases and two data sample phases in combination. These eight phases are reflected in Figure 8, page 7.

# Data Recovery Unit (DRU)

## State Machine Edge Detection

Figure 5 shows the locations of the sampling and comparison points relative to the data stream coming into the FPGA. There are two streams of data, one of which is offset from the other by 200 ps. The bottom data stream is the complement of the top, accomplished through use of the IBUFDS_DIFFOUT primitive. The data is sampled through four clock phases 400 ps (or 90°) apart, named CLK0, CLK90, CLK180, and CLK270.

Sampling points occur where the clocks intersect the data streams. These points are named according to the format:

$$Qx\,[S\ or\ M]x$$

where

Q*x* = the ISERDESE1 outputs Q1, Q2, Q3, or Q4
S*x* or M*x* = the source ISERDESE1 (master or slave) of the data outputs (Q*x*).

For example, sample point Q1M1 shows where CLK0 samples the data and creates an output at port Q1 of the master ISERDESE1.

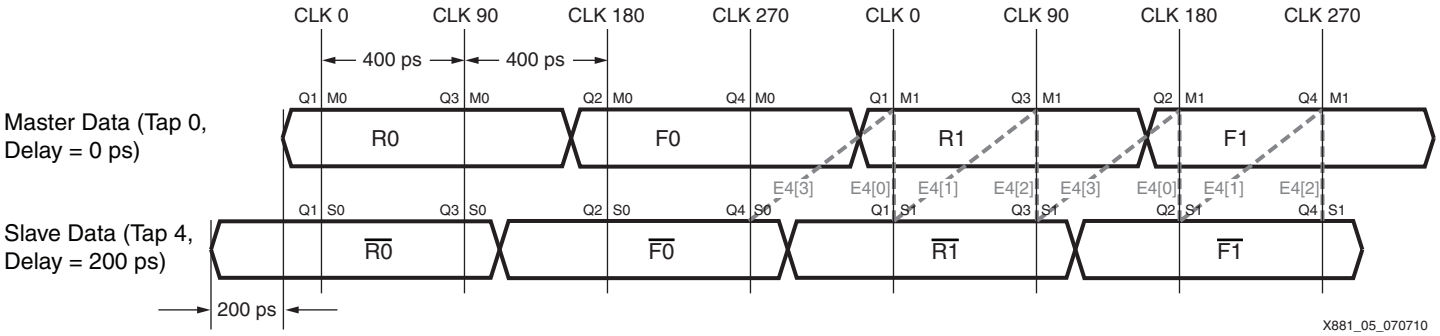


*Figure 5:* **Data Stream Sample and Comparison Points**

The lines labeled E4[0] through E4[3] that connect the sample points show where the DRU is comparing data and looking for a data edge. The formulas for the four comparisons are shown in Equation 1 through Equation 4:

$$E4[0] = [Q1M1 \; xor \; \overline{Q1S1}] \quad or \quad [Q2M1 \; xor \; \overline{Q2S1}] \qquad Equation\ 1$$

$$E4[1] = [Q3M1 \; xor \; \overline{Q1S1}] \quad or \quad [Q4M1 \; xor \; \overline{Q2S1}] \qquad Equation\ 2$$

$$E4[2] = [Q3M1 \; xor \; \overline{Q3S1}] \quad or \quad [Q4M1 \; xor \; \overline{Q4S1}] \qquad Equation\ 3$$

$$E4[3] = [Q1M1 \; xor \; \overline{Q4S0}] \quad or \quad [Q2M1 \; xor \; \overline{Q3S1}] \qquad Equation\ 4$$

These comparison points, *relative to the original data stream,* are actually 200 ps apart. For example, Equation 1 (E4[0]) compares Q1M1 against Q1S1 and Q2M1 against Q2S1. These comparisons are shown by two gray dashed lines each labeled E4[0]. Looking first at the Q1M1 *xor* Q1S1 comparison, it can be seen that both points are sampled by CLK0; however, the Q1S1 sample is delayed 200 ps (by the action of IODELAYE1) relative to Q1M1, thus allowing comparison of two samples 200 ps apart. Similarly, Q2M1 and Q2S1 are both sampled by CLK180, but again the sample points are separated by 200 ps due to the action of IODELAYE1 on the slave data stream. If either the CLK0 or CLK180 sample points produces an *xor* result of 1—that is, the levels of the sampled data do not match—it can then be concluded that there is an edge (a level transition) between those two sample points. In the example shown in Figure 5, the first E4[0] sample point comparison occurs in rising-edge zones R1 and $\overline{R1}$, while the second E4[0] sample comparison point occurs in falling-edge zones F1 and $\overline{F1}$. Thus, both comparisons would match, and the *xor* outputs for these tests would both be 0. The DRU state machine would know that there are no data transition edges there.

A contrasting example is shown by Equation 4, which compares Q1M1 against Q4S0, as well as Q2M1 against Q3S1. Q1M1 is sampled by CLK0 from the master data stream, while Q4S0 is sampled by CLK270 from the slave (phase-delayed) data stream, and is then stored for an extra cycle in the DRU. CLK0 and CLK270 are 400 ps (90°) apart, but since the slave data is delayed by 200 ps, the Q1M1 and Q4S0 sample points are really only 200 ps apart, again *relative to the original data stream*. Similarly, Q2M1 is sampled by CLK180 and Q3S1 is sampled by CLK90, and again, the sample points are 200 ps apart relative to the original data stream. For each comparison, one sample point falls in a rising-edge zone and the other falls in a falling-edge zone. These two comparisons would produce an *xor* result of 1, indicating that an edge (level transition) exists somewhere between the two sample points in each comparison.

Continuing where Figure 5 left off, Figure 6 shows what Equation 1 through Equation 4 look like in logic and how the data flows out of the ISERDESE1 and into that logic. There is a stage of registers between the ISERDESE1 and the logic to facilitate the timing. This also shows how the Q4 output of the slave ISERDES is stored from the previous sample set to be compared with the new sample set.

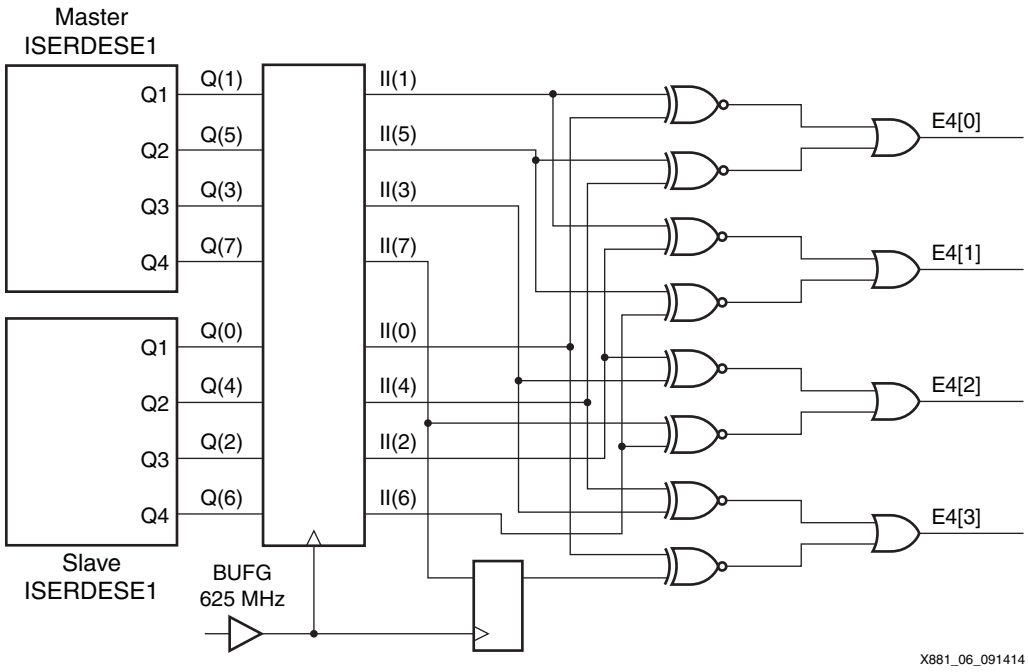


*Figure 6:* **Logical Representation of Edge Detection Circuit**

At this point, it should be clear how the data comes into the FPGA and is then fed into the DRU for edge detection. The next step in the DRU is to process the comparison data. This simple state machine, based upon where the data edge was and where it moves to, then chooses a sample point away from the data edge.

The ideal sample point can be expected to move around because of voltage and temperature variations, jitter, and offset between the source and receiver clocks. This means that the comparison point equations are always changing value, and the state machine is always updating based on these changing results. Figure 7 shows the flow of the state machine from one set of data to the next.

| EQ | DO |
|---|---|
| 00 | Use Samples Q(0) & Q(4) |
| 01 | Use Samples Q(1) & Q(5) |
| 11 | Use Samples Q(2) & Q(6) |
| 10 | Use Samples Q(3) & Q(7) |

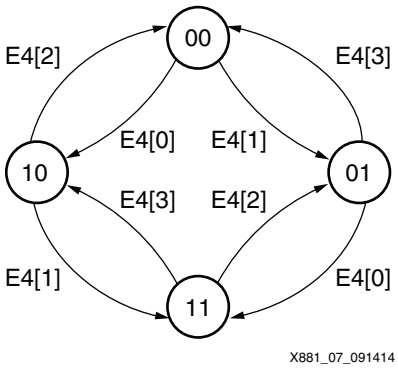


*Figure 7:* **Logic for Choosing Data Selected with Edge Information**

In the table included in Figure 7, the EQ column shows the current position of the state machine with input from Equation 1 through Equation 4. The DO column shows what sample set is used in the interconnect logic. Remembering that each ISERDESE1 in oversampling mode is like two sets of IDDR flip-flops, DO is indicating which IDDR flip-flops should be used as the ideal sample points.

The state machine diagram to the right of the table shows, for each given state or sample set, where the state machine would go next. For example, assume the state machine starts in

state 01, which uses the Q(1) and Q(5) signals. This maps to Q1 and Q2 out of the ISERDESE1 master, which would be CLK0 and CLK180 respectively.

Then, if the data edge were to move to the left, the center point would be shifted from CLK0/CLK180 to CLK90/CLK270. This would be seen by E4[3] changing its value from 0 to 1. When this happens, the state machine moves from state 00 to state 01.

## Bit Skip

Bit skip occurs when an edge moves to the left of the first sample of a data bit or to the right of the last sample of the data bit.

When an edge is detected to the left of the last sample, the new current sample is moved from the last sample to the right which corresponds to the first sample of the next data. In Figure 8, when the state machine is in state 10, it samples Q(3) and Q(7). The state machine then changes to state 00, sampling Q(0) and Q(4). However, a sample of data was already taken in the previous state when the state machine was 10; as a result, during the first 00 state of the state machine, one bit of the sampled bits is dropped. This is called a negative bit skip. Negative bit skip outputs five bits per clock.

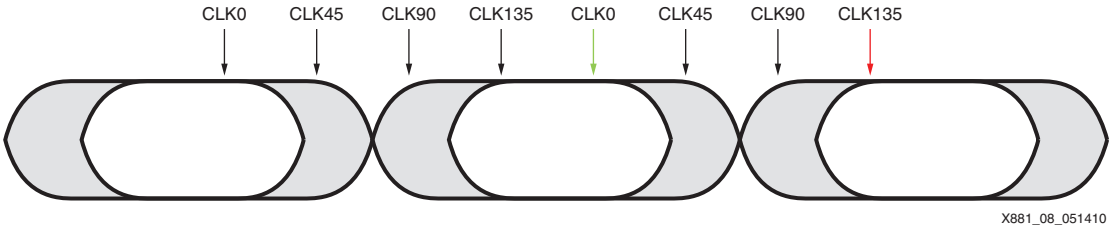


*Figure 8:* **Negative Bit Skip**

When the edge is detected to the right of the first sample, the new current sample is moved to the left, which corresponds to the last sample of the next data. In Figure 9, when the state machine is at 00, it samples Q(0) and Q(4). The state machine then changes to state 10. In this state, it samples Q(3) and Q(7). However, no sample of data was taken during states 00 and 10; as a result, during state 10 of the state machine, the last sample is taken along with current samples, causing seven bits to be output. This is called a positive bit skip. Positive bit skip outputs seven bits per clock.

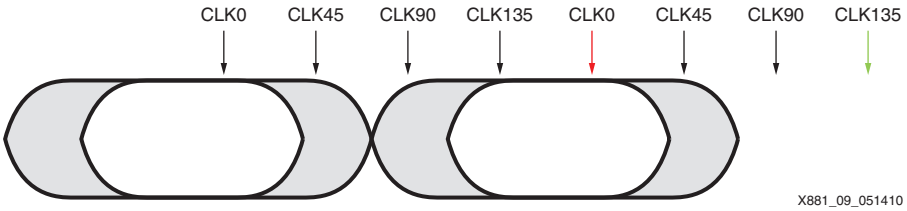


*Figure 9:* **Positive Bit Skip**

From Figure 8, and Figure 9, bit skip occurs when there is a transition between states 00 and 10.

When there are no bit-skip conditions, the sampled data output has one bit per clock in the SDR mode and two bits per clock in the DDR mode.

Therefore, for six bits of parallel data:

- The number of bits for a negative bit skip condition is five
- The number of bits for a positive bit skip condition is seven
- The number of bits for no bit skip condition is six

## Clocking and Data Flow

There are several clocks and phases of clocks, each performing a needed function, as shown in Table 1.

*Table 1:* **Clocks from MMCM RX**

| Source | Frequency/ Phase Shift | Buffer | Destination |
|---|---|---|---|
| Off-chip oscillator | 125 MHz | Input buffer | MMCM |
| MMCM RX | 625 MHz, 0° phase shift | Single-region BUFIO | ISERDESE1 |
| | 625 MHz, 90° phase shift | Single-region BUFIO | ISERDESE1 |
| | 625 MHz, dynamic phase shift | BUFG | CLB(DRU) |
| | 312.5 MHz, dynamic phase shift | BUFG | CLB(DRU) |
| MMCM IDELAYCTRL | 310 MHz | BUFG | IDELAYCTRL |

With the design running at full rate to capture data and run the DRU, timing is critical and requires both timing and placement constraints. Timing constraints and special requirements are summarized in Table 2.

*Table 2:* **Timing Constraints and Special Efforts Required**

| Data Flow(1) | Frequency/ Phase Shift | Capture Elements | Timing Constraints | Special Efforts That Could Be Required to Close Timing |
|---|---|---|---|---|
| | 125 MHz | RX source clock feeding MMCM | TIMESPEC TS_RxC_P = PERIOD "RxC_P" 8 ns HIGH 50%; | None |
| 1, 4 | 625 MHz, 0° phase shift | ISERDESE1 | TS_CLKOUT0 = PERIOD TIMEGRP "CLKOUT0" TS_RxC_P / 5 HIGH 50%; | None |
| 1 | 625 MHz, 90° phase shift | ISERDESE1 | TS_CLKOUT1 = PERIOD TIMEGRP "CLKOUT1" TS_RxC_P / 5 HIGH 50%; | None |
| 2A, 4 | 625 MHz, dynamic phase shift | ISERDESE1 to CLB DRU | TS_CLKOUT2 = PERIOD TIMEGRP "CLKOUT2" TS_RxC_P / 5 HIGH 50%; | • Dynamic phase alignment to match BUFIO<br>• Maximum delay constraint<br>• RLOC for CLB array |
| 2B, 3, 4 | 312.5 MHz, dynamic phase shift | CLB DRU | TS_CLKOUT3 = PERIOD TIMEGRP "CLKOUT3" TS_RxC_P / 2.5 HIGH 50%; | • Dynamic phase alignment to match BUFIO<br>• RLOC for CLB array |
| 1, 2A | | ISERDESE1 to CLB DRU | NET "*/Q<*>" MAXDELAY = 0.6 ns; | • RLOC for CLB array |

**Notes:**

1. Numbers in this column refer to the data flow stages and the numbered clocking transfer points shown in Figure 10.

The flow of data from the pins of the FPGA to the 10-bit wide interface presented to the FPGA interconnect logic takes several steps. Figure 10 shows a diagram with the clocking transfer points and registers used. The numbers above each section represent a clocking transfer point.

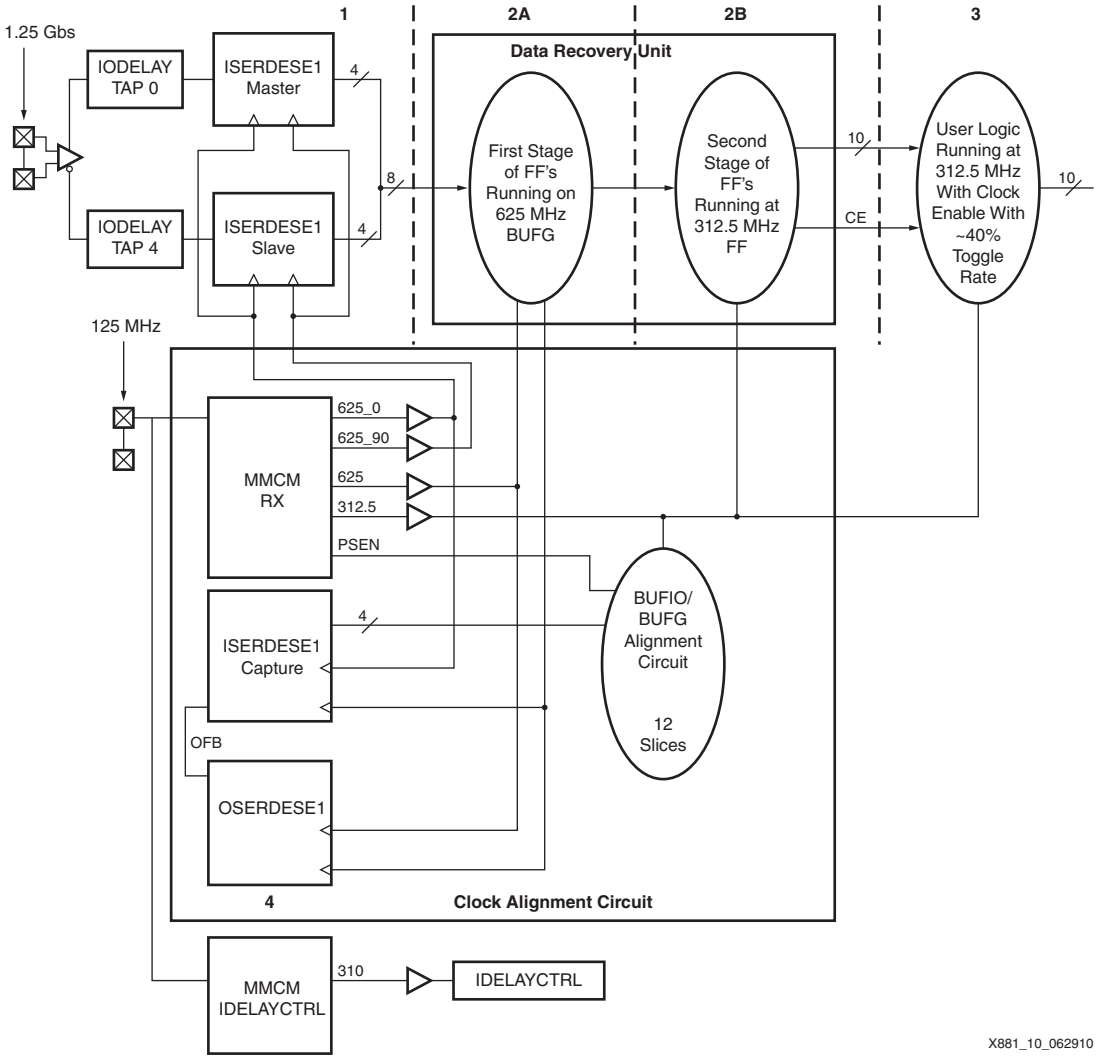


*Figure 10:* **Flow of Data through the Clocking Transfer Points**

1. Data enters the FPGA at 1.25 Gb/s. The ISERDESE1/IODELAY capture block shows where the data is being captured during the four clock phases. Figure 10 shows these capturing clocks labeled as 625_0 and 625_90. These are single-region BUFIO clocks, a requirement for this design.
2. DRU
   a. The next stage of data capture is to transfer the data from ISERDESE1 to the CLB flip-flops. *It is important that the delay from ISERDESE1 to all the registers being used does not exceed 600 ps.*

      The important part of this transfer is that it moves from the BUFIO clock network to the BUFG clock network; the BUFIO clock network only spans to the ISERDESE1 but does not span to the CLBs.

   b. In this stage, data is being handed from the 625 MHz BUFG clock to the 312.5 MHz BUFG clock. These clocks are in phase with each other.

3. When the 10 bits of data are selected by the DRU, they are presented to the user interconnect logic with a clock enable. The clock enable is present and running at this speed to account for when the data is running faster or slowing the capture clock.

### Clock Alignment Circuit

BUFIO and BUFG have an undefined phase relationship with each other. To define the phase relationship, a calibration scheme is performed. This clock alignment circuit uses the output feedback path from the OSERDESE1 to the ISERDESE1. By forwarding the 625 MHz BUFG clock to the OSERDESE1 and using the 625 MHz BUFIO clock to capture the clock with the ISERDESE1, the phase relationship between the two clocks can be measured. Then, by using the independent phase-shift capability of the MMCM, the BUFG clocks are phase-shifted to match the BUFIO clocks.

The phase calibration process is illustrated in Figure 11.

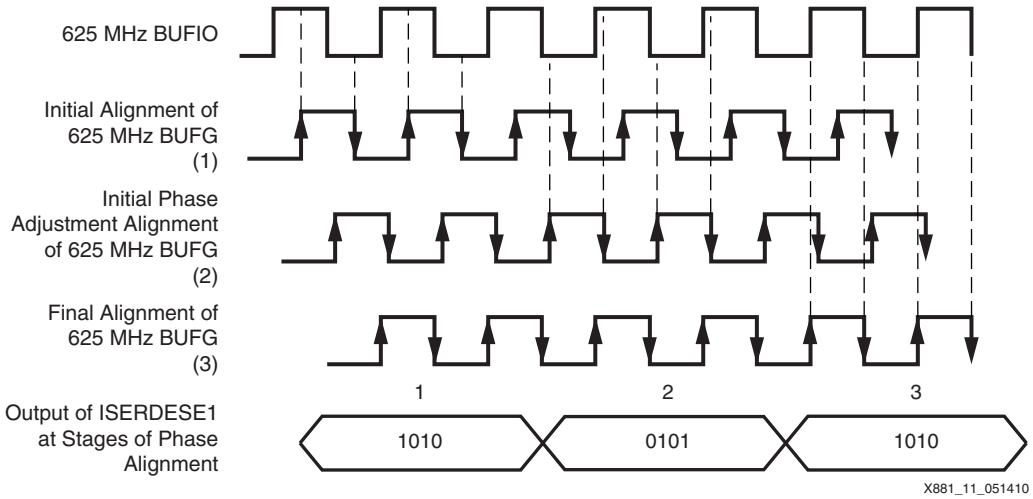


*Figure 11:* **BUFIO to BUFG Clock Phase Calibration Process**

## Reference Design

The reference design files are available for download at:

https://secure.xilinx.com/webreg/clickthrough.do?cid=148941

The reference design software tree structure is shown in Figure 12.

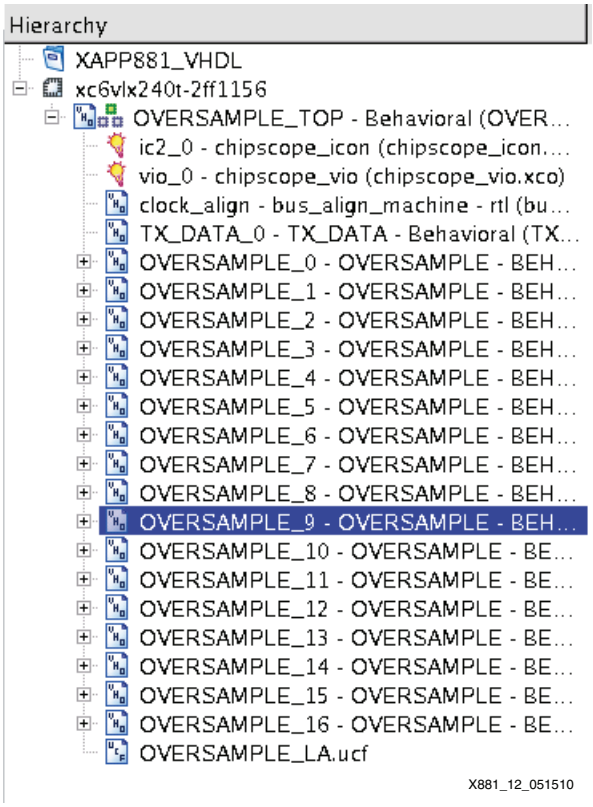


*Figure 12:* **Reference Design Structure**

## Logical Resources Used

Resources used by this reference design is summarized in Table 3 through Table 5.

*Table 3:* **Per Device**

| | |
|---|---|
| MMCM | 1 |
| BUFG | 1 |

*Table 4:* **Per Bank**

| | |
|---|---|
| MMCM | 1 |
| BUFG | 2 |
| Single Region BUFIOs | 2 |
| Clock Alignment Circuit | 45 LUTs |
| IDELAYCTRL | 1 |
| ISERDESE1 for Clock Alignment Circuit | 1 |
| OSERDESE1for Clock Alignment Circuit | 1 |

*Table 5:* **Per Channel**

| | |
|---|---|
| Data Recovery Unit | 87 LUTs |
| ISERDESE1 | 2 |
| IODELAYE1 | 2 |

## Receiver UI Requirement and Jitter Tolerance

To understand the receiver jitter tolerance, establish a starting point. Given the DRU method used, two valid sampling points are required at all times. This means the starting point is 0.500 UI. The oversampling is based on evenly-spaced sampling points; any error in that spacing causes the receiver jitter eye requirement to increase.

Receiver Jitter Eye Requirement =
Eye Requirement of DRU + Sampling Phase Error *Equation 5*

0.625 UI = (0.500 UI) + (0.125 UI) *Equation 6*

This next section describes what is included in the sampling phase error and what is *not* included. Sampling error comprises all of the effects of taking a 125 MHz clock and multiplying it up to 625 MHz, then feeding it to the two BUFIOs with a phase shift and the IODELAYE1's 200 ps phase shift.

**Sampling phase error includes:**

- MMCM jitter for the exact setting in the reference design
- MMCM phase error between CLK0 and CLK90
- MMCM DCD
- IODELAYE1 delay accuracy (ability to create a 200 ps phase shift)
- IODELAYE1 pattern-dependent jitter
- Any offset between the two paths of the master and slave ISERDESE1

**Sampling phase error does *not* include:**

- Any other frequency of clock or setting of the MMCM
- Signal integrity losses (ISI, board jitter…)
- Internal device jitter
- Anything not expressly stated as included

Extensive characterization was performed over process, voltage, and temperature on multiple pins to qualify this interface. Table 6 summarizes the amount of jitter that can be tolerated.

*Table 6:* **Jitter Tolerance Test Results at 1.25 Gb/s**

| Device | $V_{CCINT}$ | $V_{CCAUX}$ | $V_{CCO}$ | TEMP | TOTAL JITTER (UI) |
|---|---|---|---|---|---|
| -2 speed grade | 0.95 | 2.325 | 2.325 | 100°C | 0.375 |
| | | | | –40°C | 0.375 |
| | 1.05 | 2.625 | 2.625 | 100°C | 0.375 |
| | | | | –40°C | 0.375 |
| -3 speed grade | 0.95 | 2.325 | 2.325 | 100°C | 0.375 |
| | | | | –40°C | 0.375 |
| | 1.05 | 2.625 | 2.625 | 100°C | 0.375 |
| | | | | –40°C | 0.375 |

## Conclusion

The Virtex-6 FPGA is designed to implement FPGA-to-FPGA asynchronous interfaces that helps to reduce cost and conserve transceivers.

## Revision History

The following table shows the revision history for this document.

| Date | Version | Description of Revisions |
|---|---|---|
| 07/23/2010 | 1.0 | Initial Xilinx release. |
| 07/25/2010 | 1.0.1 | Title update. |
| 09/24/2014 | 1.1 | Updated E4[2] in Equation 3. Updated the XNORs in Figure 6. Revised the direction of arrows in Figure 7. Revised link in Reference Design section, the ZIP file is also updated with this release. Updated Notice of Disclaimer. |

## Notice of Disclaimer