XAPP224 (v1.1) January 10, 2001

Data Recovery in Virtex and Virtex-II Devices

Author: Nick Sawyer

Summary

Data recovery is a mechanism that allows a receiver to extract embedded clock data from an incoming data stream. The receiver usually extracts this information from the data stream concerned, but sometimes the receiver's clock is used for data transmission. The circuit described in this application note provides a partial solution at data rates up to 160 Mb/s in a Virtex™-E, -7 device, and up to 210 Mb/s in a Virtex-II device. The solution is partial in the sense that no clock is actually recovered, but the data arriving is fully extracted. The speed is limited by the maximum frequency that can be accepted by the Data Locked Loop (DLL), in a mode where the DLL is capable of providing both a new clock, and another clock shifted by 90 degrees. A typical application is shown in Figure 1.

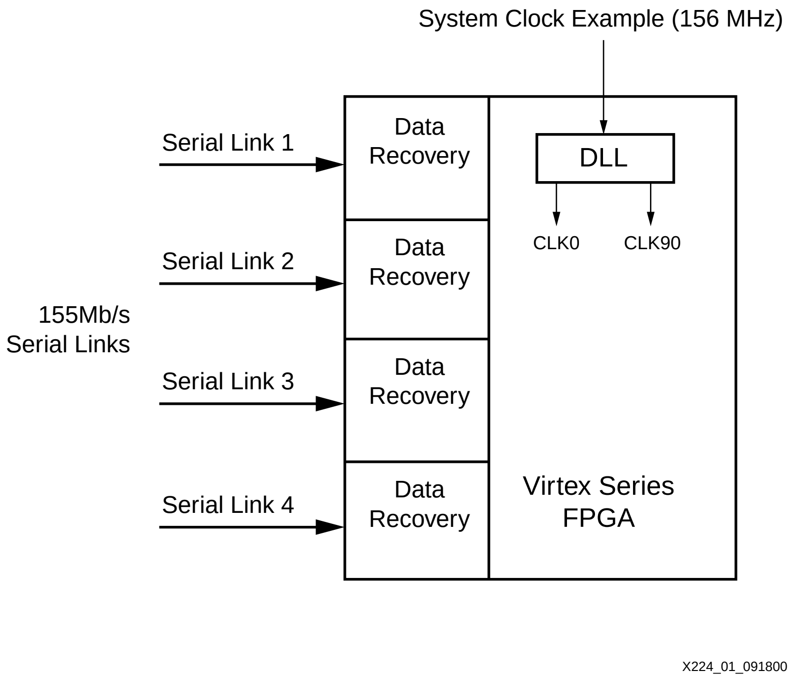

Figure 1: **Typical Data Recovery Application**

Introduction

The circuit described herein uses a clock (local oscillator) that is running slightly faster than the maximum frequency of the data stream being decoded. For example, a typical link may be running at 155 Mb/s (plus or minus a small variation), while the clock on the receiver is required to be running at 156 MHz. The actual performance relative to the clock rate will be discussed after giving a description of how the circuit works.

An assumption made at this point is that the incoming clock/data line is not encoded beyond a "1" being transmitted as line High and a "0" as line Low. Other data encoding possibilities will be discussed in future versions of this application note. Minimum transition requirements are discussed in the following sections.

The incoming system clock is fed to a DLL component, and the DLL CLK0 is used to provide a clock (CLK) for the synchronizer circuit, as well as feedback for the DLL. Another version of the input clock, delayed by 90 degrees (CLK90), and synchronized with the original clock is available. These waveforms are shown in the Figure 2 timing diagram along with the four possible data arrival cases used in the next section.

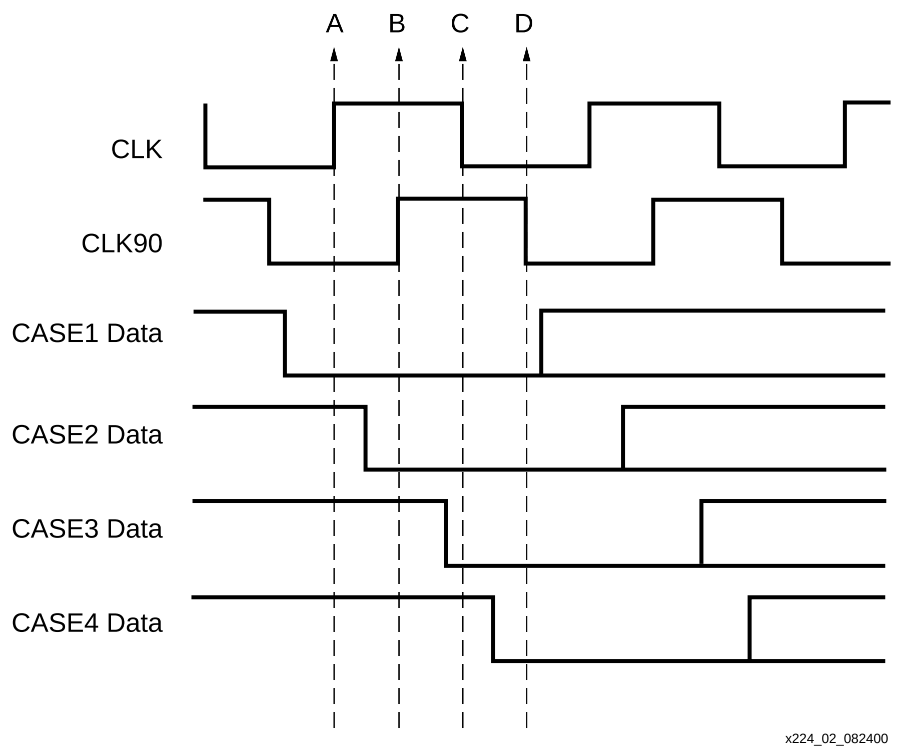

Figure 2: **Timing Diagram**

As shown in Figure 3, the incoming data is applied to four flip flops, two clocked by CLK (one rising edge and one falling edge), and two by CLK90 (rising and falling edges). It is important that the delay from the input pin to these four flip flops be almost equal. This is easily achieved by giving the software a MAXSKEW parameter for this net, of 500 ps, for example. The absolute delay is irrelevant; only the skew is important.

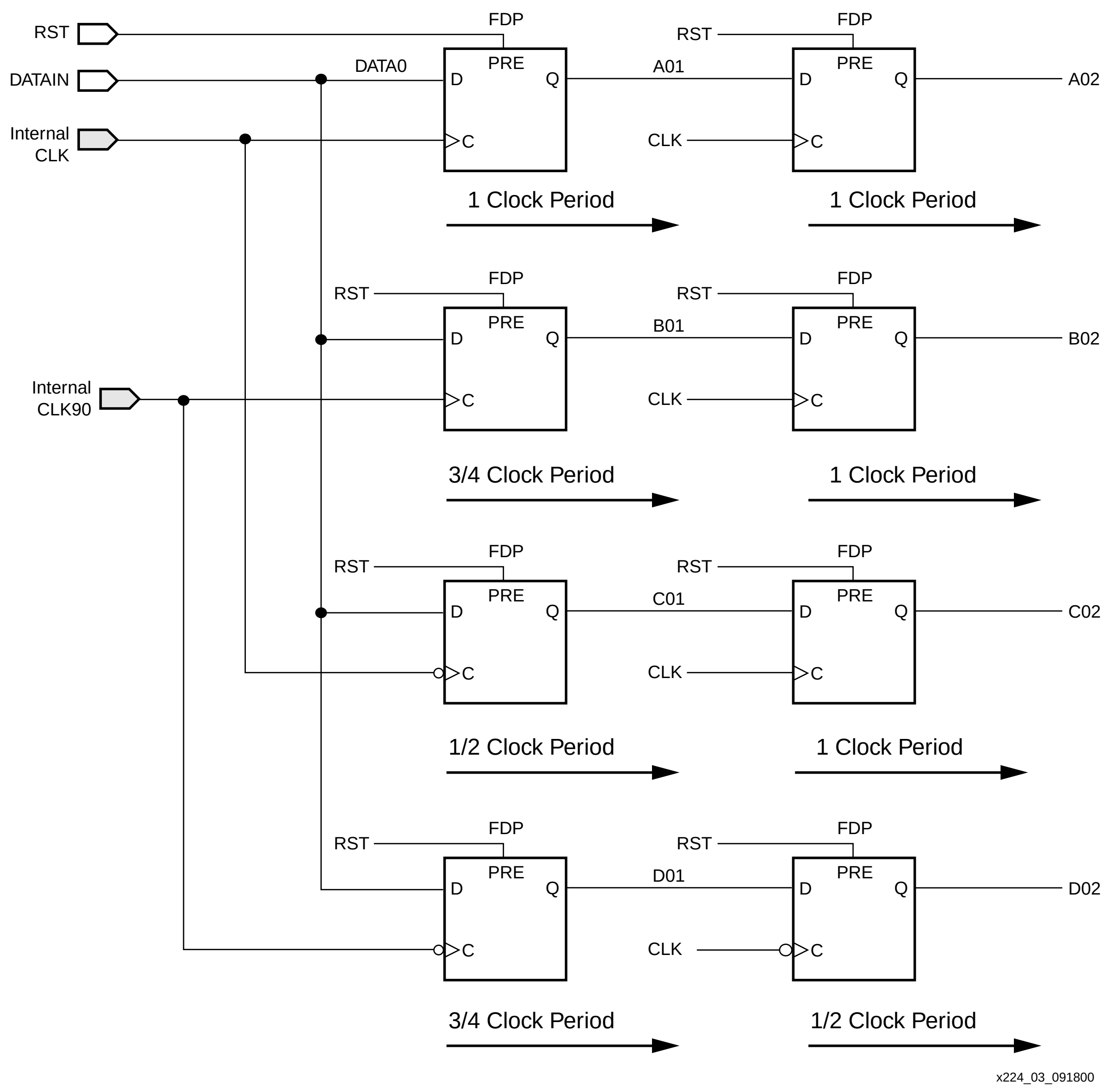

Figure 3: **Input Stage**

The first flip flop is clocked by the rising edge of the clock described as time domain A. The second flip flop is clocked by the rising edge CLK90 (time domain B); the third flip flop is clocked on the falling edge of CLK, (time domain C); and the fourth is clocked on the falling edge CLK90, (time domain D). As shown in the timing diagram (Figure 2), this gives four data sample points, each separated by 90 degrees of the original clock frequency. In the case of a 160 MHz system clock, this logic is effectively running at 640 MHz.

These four sample points are then clocked once more, to remove any metastability issues and to move them into the same time domain. This actually takes place in two stages (again to avoid any four times clock frequency logic paths).

In the first decision stage, shown in Figure 4, the circuit detects only falling edges on the data lines. This is easily changed to rising edges by reversing the connections to the AND gate. Seven signals are now available for the decision process. These are labeled A to G. In Figure 5, four mutually exclusive signals can be decoded, where only one transitions High whenever there is a negative data transition. These four conditions are as follows:

1. B = 1 and C = D = E = 0. Time domain C recognized the transition first. Use the data clocked in during time domain A for forwarding into the system. This is the data that has been sampled midway through its period, i.e., the best noise margin.
2. C = 1 and D = E = F = 0. Time domain D was the first to see the data. Therefore, the data from B is used for forwarding.
3. D = 1 and E = F = G = 0. Time domain A was the first to see the data. Therefore, the data from C is used for forwarding.
4. A = 1 and B = C = D = 0. Time domain B was the first to see the data. Therefore, the data from D is used for forwarding.

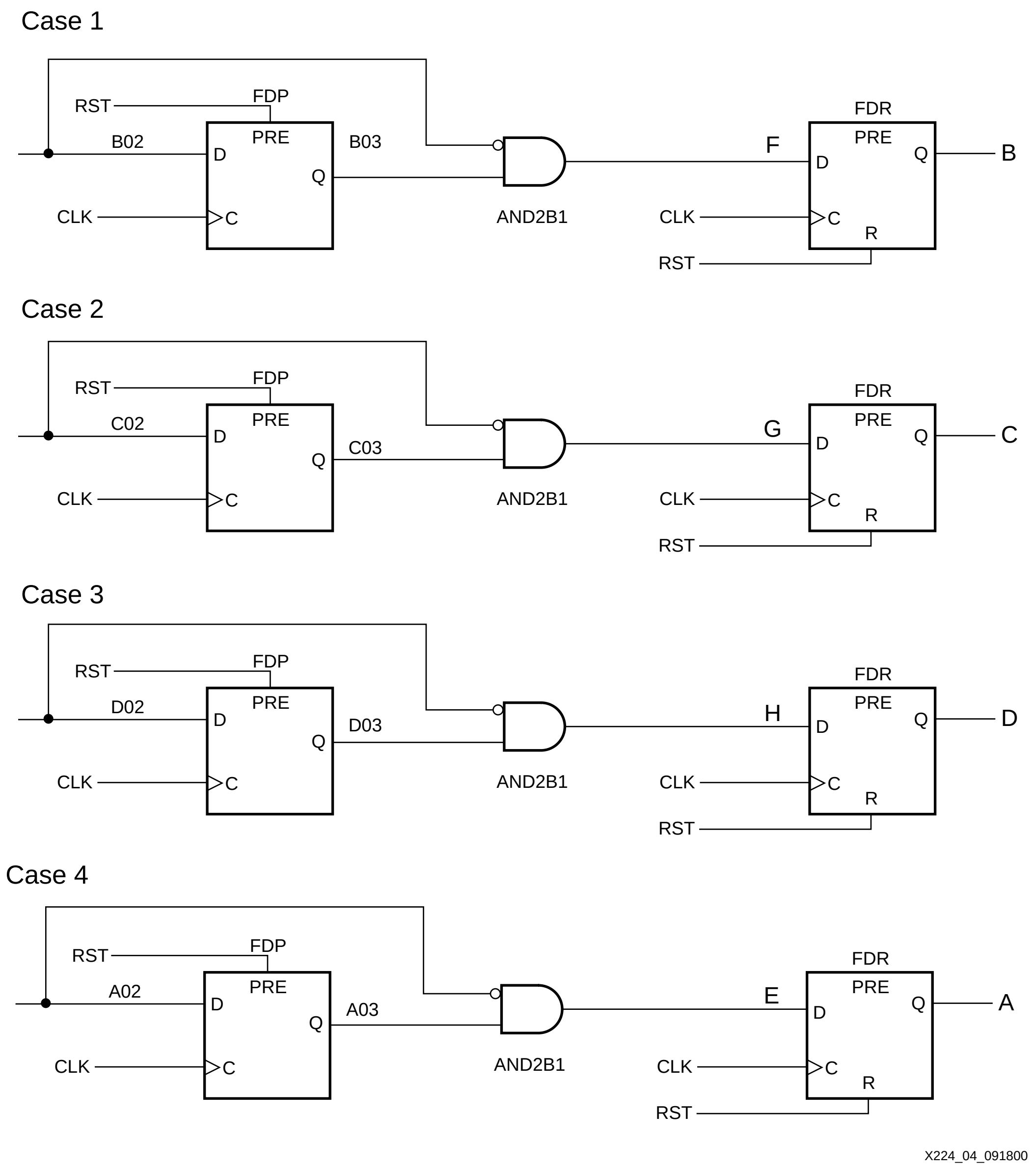

Figure 4: **Decision Stage 1**

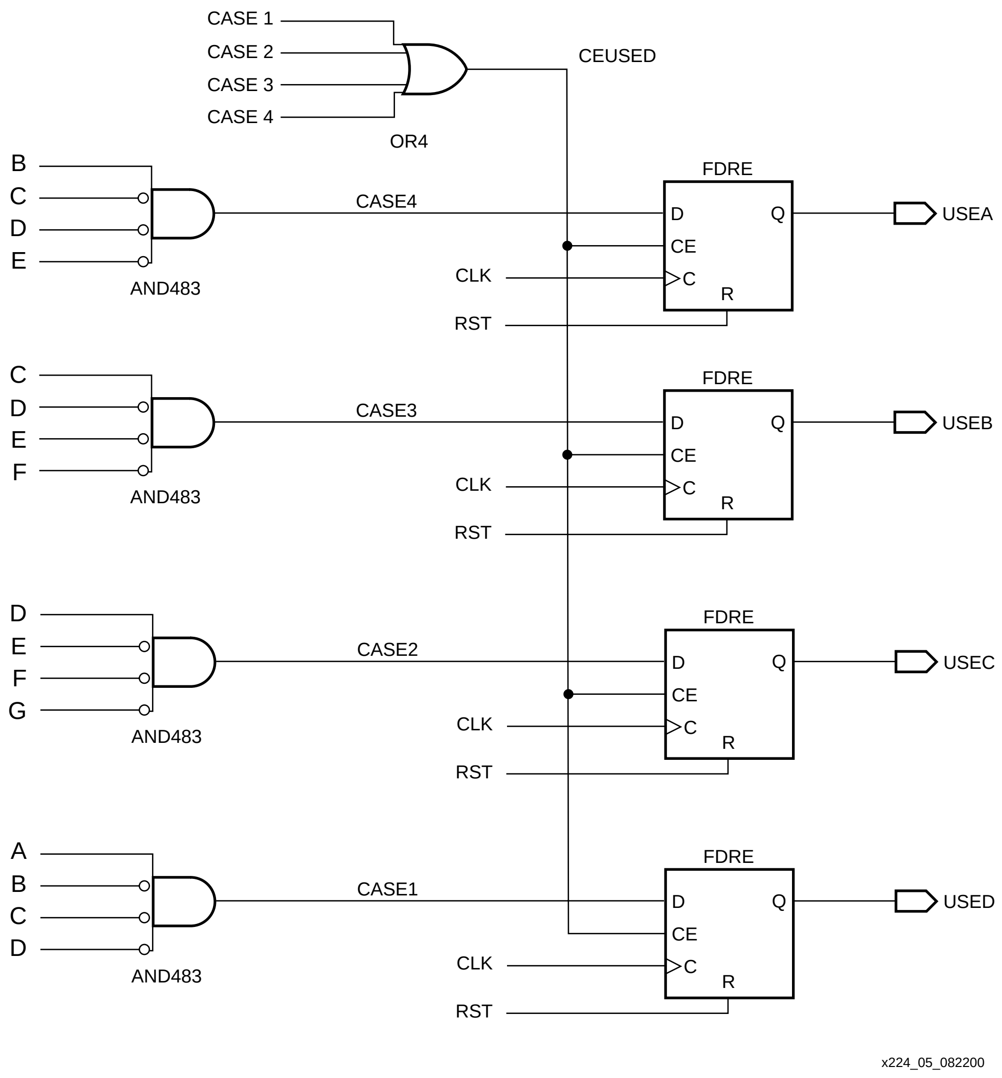

Figure 5: **Decision Stage 2**

The selection of data is done with a very simple multiplexer used to select data from the appropriate time domain (Figure 6).

The local clock is a bit faster than the incoming clock/data stream, and there will therefore be clock cycle where the received data is invalid. As the clock/data 'moves' in time we can use the multiplexer to select data from one of the four available time domains, but with one exception. If the data moves from domain A to domain D, then the data being output is actually invalid. This is indicated to the user by lowering the data valid (DV) signal. This is shown in Figure 7.

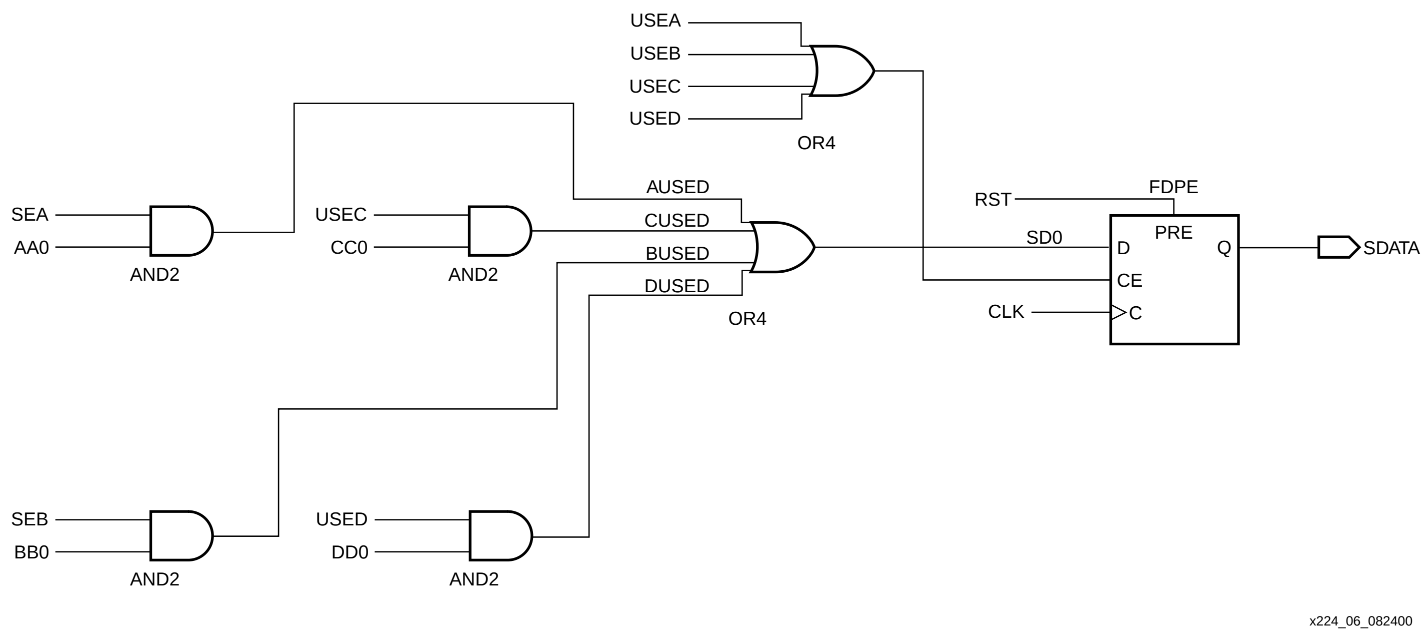

Figure 6: **Data Multiplexer**

.

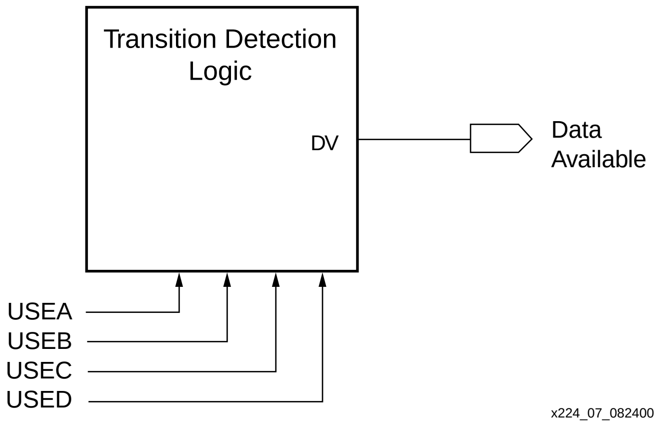

Figure 7: **Data Available Logic**

Metastability

Since it is possible that the data will not change state properly between clock sample points, there is the possibility of metastability. As the data transition approaches a sample point, (in this example "A"), it will eventually end up inside the setup time to the clock CLK, and one of several outcomes could occur.

If the flip flop is fast enough to still see the zero, then A = 1 and B = C = D = 0, (i.e., case 1) and all will work as previously described. If the flip flop does not see the zero, and remains High, then B = 1 and C = D = E = 0, (i.e., case 2,) and again all will work properly.

Finally, the flip flop could briefly enter a metastable state. If this occurs, then the second synchronizing flip flop will still "see" a one or a zero and will register that state, leading to the same arguments as above. There is no problem as long as the load on the potentially metastable flip flop is one, except when the metastable period is EXACTLY equal to the input clock period. However, this event is extremely unlikely, and even then, there is one further register in the data path. Therefore, the chance of a metastable event upsetting the operation of the circuit is small.

Lock Requirements

The circuit has a lock requirement for a negative transition to occur often enough to maintain data integrity. At least one negative transition is required in the time that the circuit takes to drift one quarter clock period. In the example in Figure 1 the data is arriving at 155 Mb/s, making one data period is approximately 6.45 ns. One quarter period is then 1.61 ns. The local oscillator frequency is 156 MHz, this equates to a period of 6.41 ns making it (6.45 – 6.41) = 0.04 ns faster than the incoming data. The quarter period (1.61 ns) divided by 0.04 ns is approximately 40. Therefore, the circuit requires at least one negative transition every 40 clock cycles to function correctly. By reducing the local oscillator frequency to 155.5 MHz, the clock cycle requirement number is increased to 80. If the received data is coded in some method, such as 8b/10b, this will not be a problem as an adequate number of transitions will exist. Care should be taken if the received data is a raw bitstream as an adequate number of transitions may not exist.

Conclusion

Virtex devices can be used to extract data from a serial link at speeds up to 160 Mb/s in a Virtex-E device, and up to 210 Mb/s in a Virtex-II device.

Reference Design

The reference design circuit is implemented in HDL. It is fully synthesizable. The reference design files (**xapp224.zip**) include a top.ucf file, containing all the timing constraint information. It is important to use this file, because some paths are very fast.

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
09/18/00	1.0	Initial Xilinx release.
01/10/01	1.1	Updated for Virtex-II series of FPGAs.