Implementation of the SpaceWire Clock Recovery Logic in Actel RTAX-S Devices

Introduction

This application notes describes the implementation of the SpaceWire clock recovery circuit in an Actel RTAX-S device. SpaceWire uses Data-Strobe (DS) encoding, and is widely used to handle payload data on-board a spacecraft. One of the challenging issues when implementing SpaceWire in an FPGA is to implement the SpaceWire clock recovery circuit that requires minimum timing of propagation delays. This application note will demonstrate how to achieve the minimum delay reliably in the RTAX-S device. The application note will prove, by providing detailed timing analysis, how the minimum timing for SpaceWire clock recovery circuit was achieved in RTAX-S device. This application note will show how the minimum timing for SpaceWire clock recovery circuit was achieved in an RTAX-S device and will provide detailed timing analysis. Additionally, this document will describe Actel's block methodology software flow, explaining how the user can design and import a SpaceWire clock recovery circuit into a main design using identical timing delay.

SpaceWire Overview

SpaceWire is a high-speed data link standard that is intended to meet the needs of future, high capability, remote sensing instruments in space missions. SpaceWire provides a unified, high-speed data-handling infrastructure for connecting sensors, processing elements, mass memory units, downlink telemetry subsystems, and EGSE equipment together. The SpaceWire interface is a point-to-point cable bus used to handle payload data on-board a spacecraft. The SpaceWire specification defines the Physical, Electrical, and Protocol layers of the interface. SpaceWire operates from 2 Mbps to 400 Mbps over a full-duplex, point-to-point serial link, over a distance of 10 meters.

SpaceWire uses a Data Strobe (DS) encoding scheme that encodes the transmission clock with the data into Data and Strobe so that the clock can be recovered by simply XORing the Data and Strobe lines together. This coding scheme is illustrated in Figure 1. The Data signal follows the data bitstream; i.e., high when the data bit is 1 and low when the data bit is 0. The Strobe signal changes state whenever the Data does not change from one bit to the next. The DS encoding and SpaceWire standard is fully described in ECSS-E-50-12A standard from European Cooperation for Space Standardization.

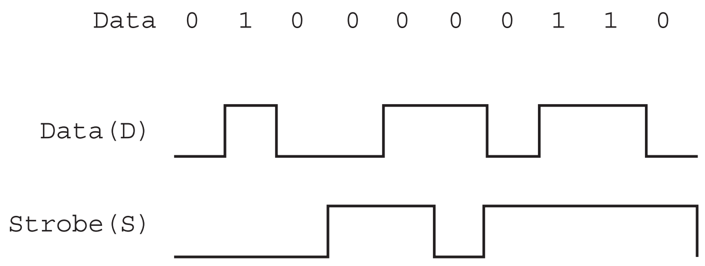

Figure 1 • **Data-Strobe Encoding**

SpaceWire uses low voltage differential signaling (LVDS) for the Data (D) and Strobe(S) signals. LVDS employs balanced signals to provide very high-speed interconnection using a low voltage swing (350 mV typical). The signaling levels used by LVDS are illustrated in Figure 2. Normally, a SpaceWire link comprises two pairs of differential signals, one pair transmitting the D and S signals in one direction and the other pair transmitting D and S in the opposite direction.

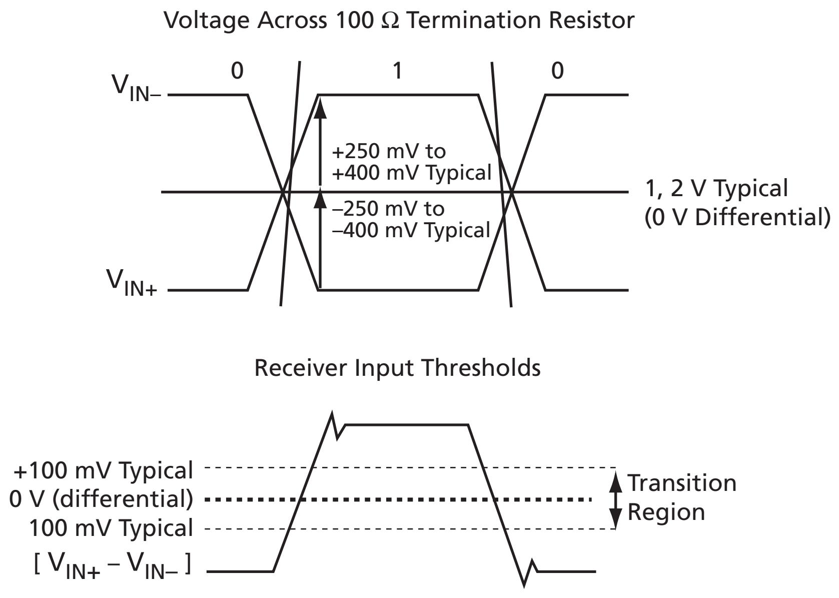
Voltage Across 100 Ω Termination Resistor
V_{IN-}
0
1
0
+250 mV to +400 mV Typical
–250 mV to –400 mV Typical
1, 2 V Typical (0 V Differential)
V_{IN+}
Receiver Input Thresholds
+100 mV Typical
0 V (differential)
100 mV Typical
$[V_{IN+} - V_{IN-}]$
Transition Region

Figure 2 • **LVDS Signaling Levels**

During implementation of SpaceWire inside the FPGA, the designer should make sure that the FPGA supports the LVDS I/O standard. Otherwise, the user has to implement external LVDS transmitter and receiver. Actel RTAX-S supports multiple I/O standards, including LVDS. This feature allows the designer to implement SpaceWire and other blocks with different I/O standard in the same FPGA.

The following sections describe the block diagram of a SpaceWire link interface and explain how to physically extract the data from the DS signals.

SpaceWire Link Interface

SpaceWire provides a means of sending packets of information from a source node to a specified destination node. Figure 3 on page 3 shows an example block diagram of a SpaceWire encoder-decoder. The transmitter is responsible for encoding the N-Chars it receives from the host system data and transmitting it using the DS encoding technique. The transmitter will operate at the desired data rate. The transmit clock is responsible for generating the variable data signaling clock signals used by the transmitter. The receiver is responsible for decoding the DS signals (Data and Strobe) to produce a sequence of N-Chars that are passed on to the host system. The Rx clock is recovered by simply XORing the received Data (Din) and Strobe (Sin) signals together. The RX Clock Recovery blocks generates the receiver clock and provides all the clock signals used by the receiver.

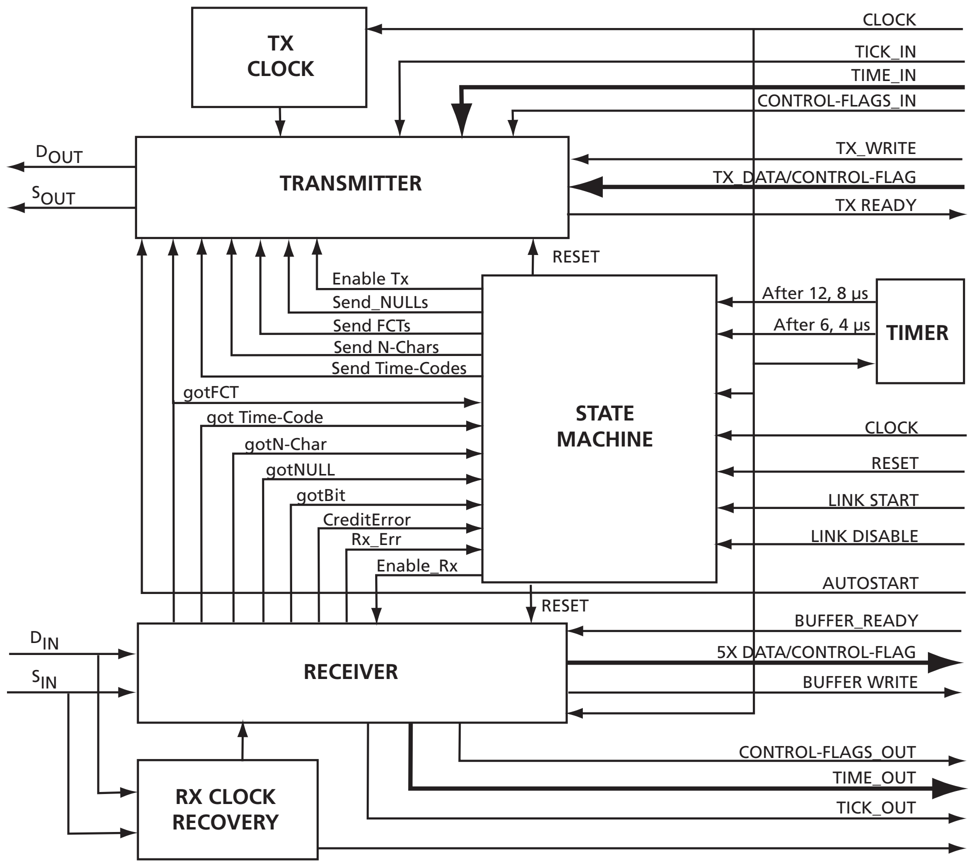

Figure 3 • **SpaceWire Link Interface Block Diagram**

This application note does not cover the full implementation of the SpaceWire encoder-decoder block in Actel FPGAs. Full implementation of the SpaceWire interface (the GRSPW core) is available from Gaisler Research, as part of Actel's CompanionCore program. Actel's CompanionCore program offers a wide selection of synthesizable IP cores that are licensed, supported, and maintained directly by the partners. The GRSPW core implements a SpaceWire codec with RMAP support and an AMBA host interface. See the *GRSPW SpaceWire Codec IP Core* user's manual at http://gaisler.com/doc/grspw.pdf for more details.

SpaceWire Receiver blocks

The SpaceWire clock recovery circuitry is implemented using XOR logic, as mentioned earlier in the "SpaceWire Overview" section on page 1. Figure 4 on page 4 shows the SpaceWire clock recovery blocks. The Data and Strobe travel to the XOR gate through the LVDS input pairs, which generates the clock. Note that the user does not need to use the LVDS I/Os as shown in Figure 4 on page 4. This can also be implemented with external LVDS drivers coming into regular TTL inputs on the RTAX-S device. This is mainly useful for board-to-board connection where people will want to isolate our RTAX-S part and avoid damaging the RTAX-S inputs. The RTAX-S device can still do the decoding on the chip.

The output from the XOR-gate is then connected to a clock network. The specific type of clock network depends on the technology used. In RTAX-S device you can use either HCLK or RCLK network. This clock is used to sample the Data signal and generate the N-Chars. In the full implementation of the SpaceWire encoder-decoder there may be extra additional logic, but in this implementation the XOR-gate is the only logic that belongs to the Rx clock recovery block.

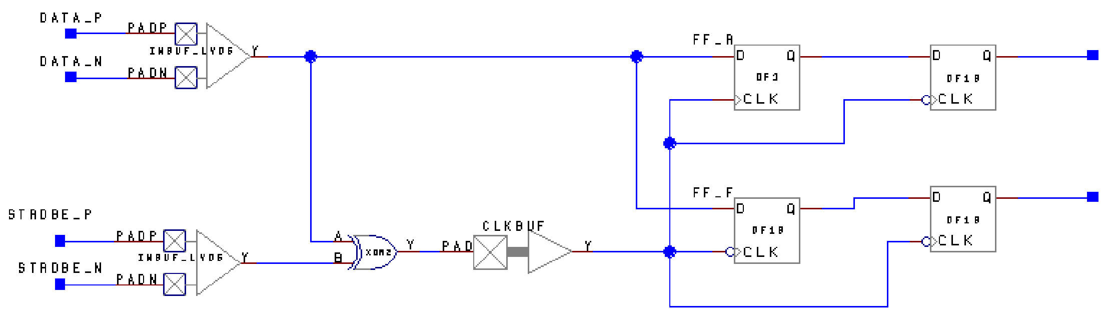

Figure 4 • **SpaceWire Clock Recovery Circuit**

To avoid race conditions between the data path from the DATA_P and DATA_N inputs to the D inputs of the flip-flops should be shorter than the clock path from the DATA_P and DATA_N inputs to the clock inputs of the same flip-flops. This can be verified by running post-layout simulation with the appropriate testbench. However, it may be tough to create a testbench that covers all the scenarios. So, the easiest way to check this is to do static timing analysis on the design.

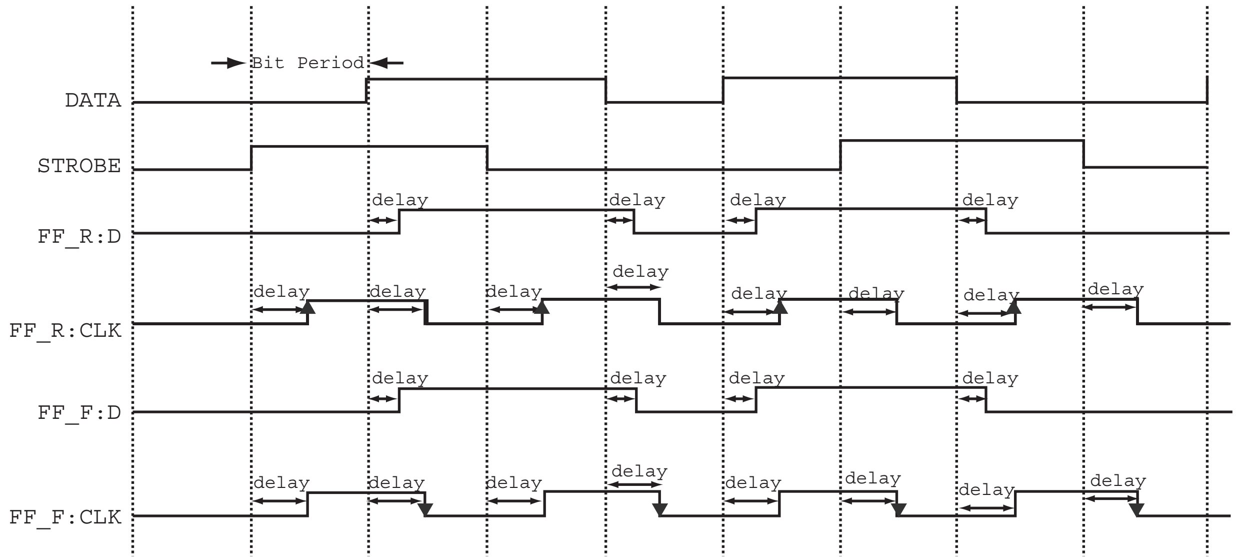

Figure 5 • **Timing Waveform for the SpaceWire Clock Recovery Circuit**

To ensure proper operation of the SpaceWire clock recovery circuit, there are three timing verifications that must be met:

1. A setup check to ensure that a data event arrives before the clock edge generated with this same data event (EQ 1):

$$\text{Longest (DATA ---> FF:D)} < \text{Shortest(DATA ---> FF:CK)} - \text{Setup (FF)}$$

EQ 1

2. A hold check to ensure that a Strobe event is not generating a clock edge that captures the wrong data:

If the external DATA is skewed (D_Skew), EQ 2:

$$\text{Bit\_Period} + \text{D\_Skew} + \text{Shortest (DATA ---> FF:D)} > \text{Longest (STROBE ---> FF:CK)} + \text{Hold (FF)}$$

EQ 2

If the external Strobe is skewed (S_Skew), EQ 3:

$$\text{Bit\_Period} - \text{S\_Skew} + \text{Shortest (DATA ---> FF:D)} > \text{Longest (STROBE ---> FF:CK)} + \text{Hold (FF)}$$

EQ 3

3. A minimum pulse width on the clock DS_CLK:

If the external Data is skewed (D_Skew), EQ 4:

$$\text{Bit\_Period} - \text{D\_Skew} > \text{Longest (DATA ---> FF:CK)} - \text{Shortest (STROBE ---> FF:CK)} + \text{Setup (FF)} + \text{Hold (FF)}$$

EQ 4

If the external Strobe is skewed(S_Skew), EQ 5:

$$\text{Bit\_Period} - \text{S\_Skew} > \text{Longest (STROBE ---> FF:CK)} - \text{Shortest(DATA ---> FF:CK)} + \text{Setup (FF)} + \text{Hold (FF)}$$

EQ 5

These inequalities must be adapted for the different edge combinations. Note that FF_R is a rising edge flip-flop and FF_R is falling edge flip-flop. As a result, the delay in the data path to FF_R when the clock is falling is not needed to verify the timing calculation. The opposite scenario holds for the FF_F. For example, inequality (1) must use the following combinations of edges for FF_R flip-flop:

DATA(Rise) ---> FF:D(Rise) with DATA(Rise) ---> FF:CK(Rise)

DATA(Fall) ---> FF:D(Fall) with DATA(Fall) ---> FF:CK(Rise)

This can be understood clearly by examining Figure 5 on page 4.

SpaceWire Receiver and Reliable Timing Analysis

Reliable minimum timing for propagation delays in the FPGA is needed to support the timing demands in SpaceWire core implementation. Without this reliable timing it is not possible to ensure that the delay of one path is shorter than the other. Since the delay through the DATA paths need to be shorter, the user needs to be assured that the minimum delay information available in the RTAX-S is reliable. In addition, it is required to prove that the timing available in the static timing analysis tool is correct over the full temperature and voltage range, the full life span of the device, after experiencing radiation total dose effects, process variations, etc.

SmartTime is a gate-level static timing analysis tool for the RTAX-S family. With SmartTime, users can perform complete timing analysis of their design to ensure that all timing constraints are met and that the design operates at the desired speed with the right amount of margin across all operating conditions. It is safe to say that the minimum timing and maximum timing delay information supported for the RTAX-S in SmartTime is very reliable. With SmartTime the user can analyze the required delay for the SpaceWire clock recovery circuit and validate the timing needs for the SpaceWire implementation.

The next sections introduce a design example for implementing the clock recovery circuit, place-and-route tips for implementing the design, and static timing analysis.

Design Example

This section provides a design example to extract data from the DS signal using a clock recovery circuit. The design example is created using Actel Libero® Integrated Design Environment (IDE). Libero IDE is Actel's comprehensive FPGA design and development software and combines the latest design creation and Designer physical implementation tools from Actel. It also includes the best-in-class synthesis and verification tools from leading EDA vendors. Using the default options and settings, the user should be able to follow the design example and meet all the timing requirements as required by the SpaceWire implementation. Additionally, the user can achieve better timing by using timing-driven layout with the proper constraint; or, if required, manually place the macro.

The design example shown in this application note uses a data extraction scheme with two selectable interface ports and a test interface. This was used in a NASA design as a frontend with the SpaceWire IP from NASA. Figure 6 shows the simplified block diagram of this design example. Based on the operating mode, only one of the XOR logic gates will be functional. For example, when A3R and A3T are disabled (the output of the A3R and A3T AND2 gates are '0'), the clock is generated by mode P (PDATARX and PSTROBERX inputs). The first sets of flip-flops to capture the Data as quickly as possible. The second sets of flip-flops to synchronize the data. The extracted clock is connected to a routed clock network. Alternatively, a hardwired clock network can be connected if the macros are positioned on the top side of the die. In addition, Test mode (T) is not critical for this implementation, so no timing analysis will be shown for this mode. The design example is available for download from the Actel website at http://www.actel.com/documents/SpaceWire_DF.zip.

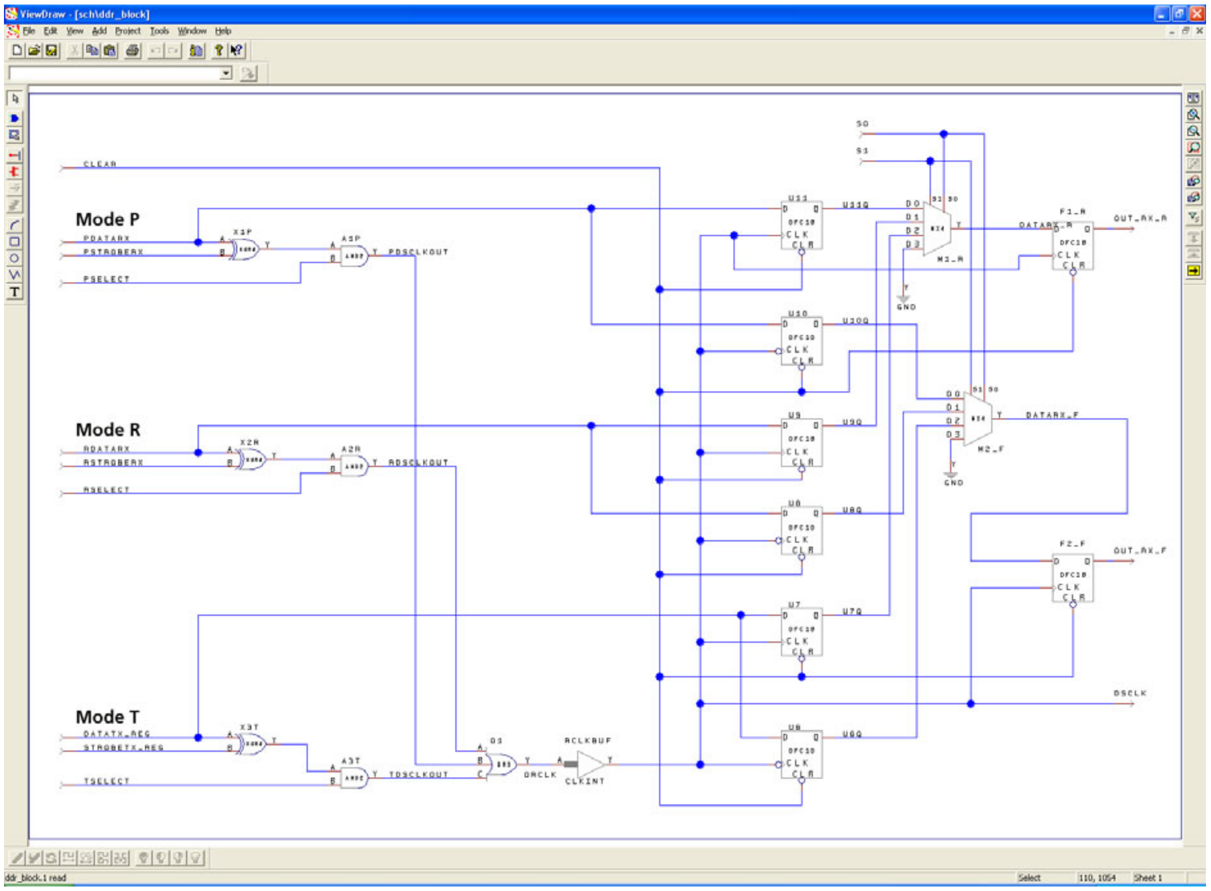

Figure 6 • **SpaceWire Receiver Design**

Design Implementation and Timing Analysis

The Designer place-and-route tool is part of Actel Libero IDE. Designer supports timing-driven layout, which is closely integrated with the SmartTime tool. Timing-driven layout works well with a synchronous design. The use of a RCLK or HCLK tree and its inherently large minimum propagation delay can help eliminate any race condition. The design example for the SpaceWire is asynchronous in nature, the user may need to manually place some of the macros to get better performance. The key to achieving this performance is to place the Data, Strobe I/Os, the XOR-gate, and the first set to flip-flops (U11, U10, etc.) close to each other. The design example uses a manual placement that is defined in a PDC (placement constraint) file, which is part of the download along with the example design file.

We already mentioned that SpaceWire operates from 2 Mbps to 400 Mbps. For this design example, we will use a clock period on 10 ns. Figure 7 shows a timing waveform for mode P. For simplicity, we are only showing the P signal for the LVDS pair (P and N) inputs. The timing delay number can be found from the SmartTime tool. The detailed timing analysis is located in an Excel file on the Actel website at http://www.actel.com/documents/SpaceWire_analysis.xls. The tabs Setup1 and Setup2 show setup check; the Hold tab shows the hold check; and the Pulse tab shows minimum pulse width check on the clock. For simplicity, we have used a skew of 2.5 ns on both Data and Strobe during calculation. The analysis shows that this SpaceWire recovery circuit meets all the required timing with a bit period of 10 ns.

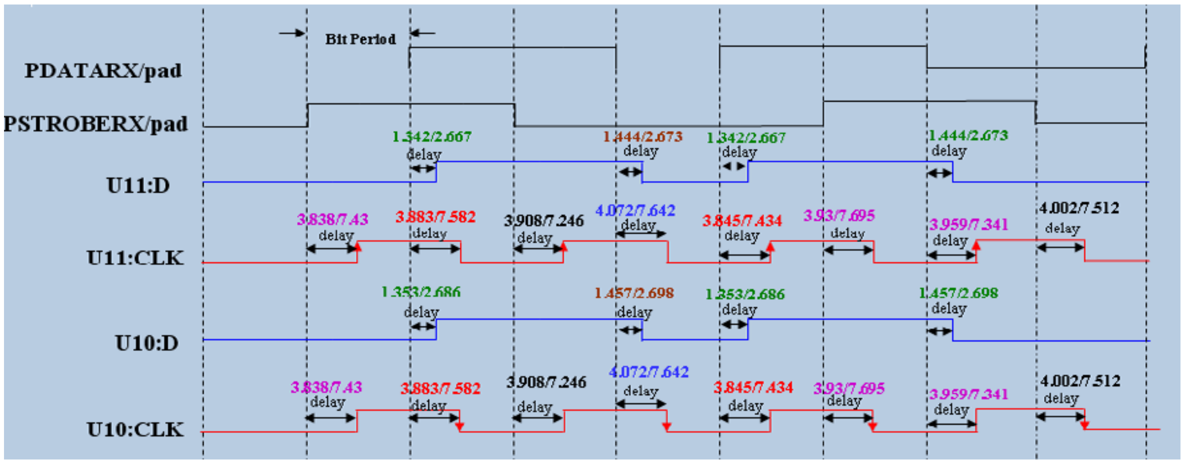

Figure 7 • **Timing Analysis for Mode P**

Block Flow

Actel Block Flow is a new feature within Libero IDE that allows the user to reuse a block of design in a new application and ensures consistent performance. The user can lock the place-and-route of the original block, which can be integrated as a Designer Block at the top-level project. The example discussed in this application note uses a single-port SpaceWire interface. If the user creates the SpaceWire clock recovery circuit with a decoder block as a standalone, it can be defined as a block that can be added to other designs using Actel block flow. This section briefly describes how to use the block for this scenario. Refer to the *Actel Libero IDE User's Guide* for detailed information on using Actel Block flow.

The user must create two Libero IDE projects in order to instantiate the Designer Block in Libero IDE: one to create and publish the Designer Block, and another in which to instantiate the Designer Block. First the user needs to create a new Designer Block in Libero IDE with a 1, 2, 4, or 8-port SpaceWire interface, as shown in the design example:

1. Start a new project for RTAX-S.
2. After your project opens, click the **Enable Designer Block** creation check box in the main toolbar.
3. Create a SpaceWire interface in Libero IDE.
4. Go through the standard design flow: create schematic/RTL, synthesize, run place-and-route. Check the timing of the SpaceWire interface.
5. Publish the Designer Block. To publish your Designer Block design in Libero IDE, Choose **Publish Designer Block** from the **File** menu.
6. Save your design to return to edit your Designer Block later. When you publish your Designer Block, Libero IDE creates a netlist file (*.v; *.vhd), info file for Libero IDE (*.cxf), and Designer Block file (*.cdb).

The next step is to instantiate this Designer Block into the top level, which includes other SpaceWire blocks. The user needs to import the Designer Block in Libero IDE by importing the design netlist and CXF file. The CXF file imports all the files that are needed for the Designer Block. After the files have been imported, the user will need to follow the normal design flow for regular Libero IDE designs.

Conclusion

The SpaceWire clock recovery circuitry is implemented using XOR logic, and it is required that the delay through data path is shorter than clock path to avoid any race condition. The use of the clock tree and its inherently large minimum propagation delay will help eliminate any race condition that may arise. Using the SmartTime timing analysis tool gives reliable timing analysis, quickly and easily. Actel Block Flow allows the reuse of any design block that has already met required timing. The SpaceWire receiver clock recovery circuitry can be reliably implemented in the RTAX-S devices with the aid of various tools in Actel Libero IDE.

Related Files

Timing Analysis

http://www.actel.com/documents/SpaceWire_analysis.xls

Design Example

http://www.actel.com/documents/SpaceWire_DF.zip www.actel.com

Actel Corporation
2061 Stierlin Court
Mountain View, CA
94043-4655 USA
Phone 650.318.4200
Fax 650.318.4600

Actel Europe Ltd.
River Court, Meadows Business Park
Station Approach, Blackwater
Camberley, Surrey GU17 9AB
United Kingdom
Phone +44 (0) 1276 609 300
Fax +44 (0) 1276 607 540

Actel Japan
EXOS Ebisu Bldg. 4F
1-24-14 Ebisu Shibuya-ku
Tokyo 150 Japan
Phone +81.03.3445.7671
Fax +81.03.3445.7668
www.jp.actel.com

Actel Hong Kong
Suite 2114, Two Pacific Place
88 Queensway, Admiralty
Hong Kong
Phone +852 2185 6460
Fax +852 2185 6488
www.actel.com.cn

51900167-0/8.07