Source-Synchronous Clock Designs: Timing Constraints and Analysis

Table of Contents

Introduction

Synchronous logic runs in sync with the clock that exists in the digital system. This means that the clock is used to generate the data or control signals that will be used by logic. Typically, on a later clock edge. The traditional synchronous interfaces in a digital system restrict the overall system performance and limit the printed circuit board (PCB) trace length. This method restricts system designers to achieve the high-speed data signaling that today's market demands. As a solution, the system designers are turning to source-synchronous clock designs that demonstrate the high interconnect speed at distances of few meters.

This application note describes techniques for constraining and analyzing source-synchronous clock interfaces. A basic understanding of Libero® Integrated Design Environment (IDE) design flow is assumed from the user. Refer to the *Libero IDE User's Guide* to understand the design flow.

Single Common Clock Topology

Synchronous designs typically have a topology wherein a clock source (like a crystal clock-oscillator or Phase-Locked-Loop) generates a single clock signal, and this clock-signal is routed across a PCB to various separate integrated circuits. (Sometimes this signal may be duplicated with a special clock-buffer chip, but all clock signals generally receive the clock edges at the same point in absolute time).

The Synchronous system has two logic endpoints: A ‘source’ endpoint and a ‘sink’ endpoint. The ‘source’ endpoint is typically a flip-flop that changes its logic state to a valid logic-level when the main system clock rises, and then propagate through various combinational logic gates, and then must be valid before the next rise-edge on the ‘sink’ endpoint flip-flop which will sample this logic signal. Figure 1 illustrates the common clock topology.

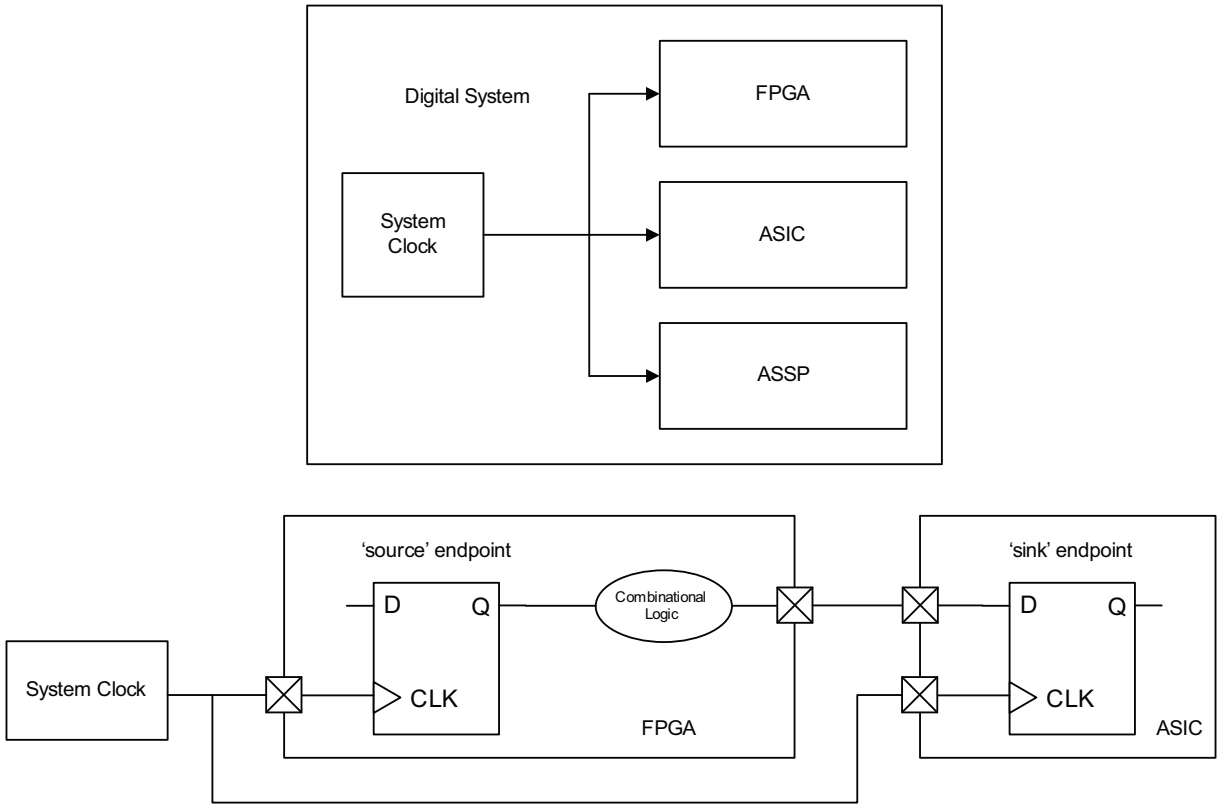

Figure 1 • **Common Clock Topology**

Source-Synchronous Clock Topology

A special variation of a synchronous design is a source-synchronous clock. In source-synchronous clock systems, the source endpoint does not use the same clock as the ‘sink’ endpoint. The ‘source’ endpoint is typically a flip-flop that changes its logic state to a valid logic-level when the main system clock rises, and then may propagate through various combinational logic gates, and then is typically output out of this logic system. In addition, a source-synchronous clock logic system also outputs a clock signal.

This data signal and new clock signal are output as a pair of signals, to a ‘sink’ endpoint. The ‘sink’ endpoint may be a separate chip or even a separate logic system at the end of a cable. The ‘sink’ endpoint will sample the received data signal, by using the new-clock signal as its clock to sample the data. This is a common topology for high-speed data signaling where the data and clock signals can be sent as output together as a synchronous-pair of signals, and allows the ‘sink’ endpoint to sample the received data at high speeds and high reliability. Figure 2 illustrates the source-synchronous clock topology.

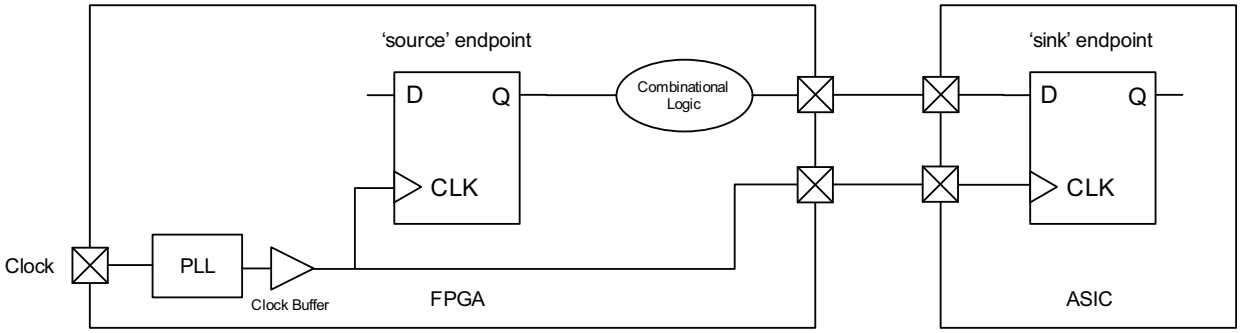

Figure 2 • **Source-Synchronous Clock Topology**

Design Example Overview

The design example is very simple. It consists of a global clock-buffer and a D Flip-Flop. Libero project is provided in the design files attached with this design example (refer to "Appendix A – Design Files" on page 11). Figure 3 illustrates the logic diagram of design example.

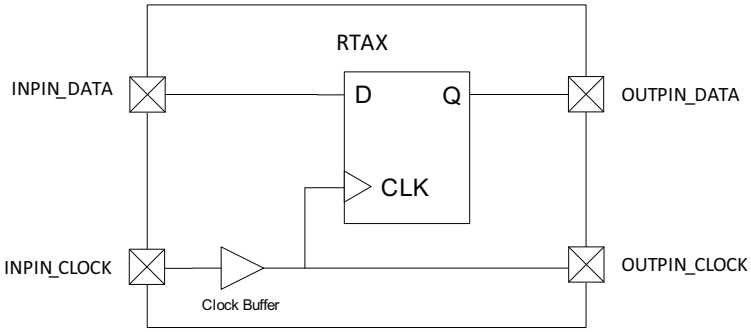

Figure 3 • **Logic Diagram of Design Example**

Note: This application note describes the source-synchronous clock design timing constraints and analysis on RTAXS (Axcelerator product family) FPGAs, although the concept is applicable to any of the Microsemi FPGA product families.

Design Description

This logic-design is intentionally very simple, so that the focus is on the timing rather than the logical circuit.

This design has the following Inputs/Outputs:

Input: INPIN_CLOCK

Input: INPIN_DATA

Output: OUTPIN_CLOCK

Output: OUTPIN_DATA

The logic of this design consists of a global clock buffer (since this is what designers will typically use to drive an internal clock signal), and a D flip-flop. The global clock buffer in this design example drives the FPGA's internal synchronous circuits. It will also drive an output out of the FPGA fabric, and it represents this designs output port named OUTPIN_CLOCK. The flip-flop in this design example represents the final logic element of a design before it is sent as a output out of the FPGA (it's the only element in this example), and this will drive the output out of the FPGA fabric, and represents this designs output port named OUTPUT_DATA.

Figure 4 provides the graphical view of example design in the HDL-Analyst RTL-Viewer tool.

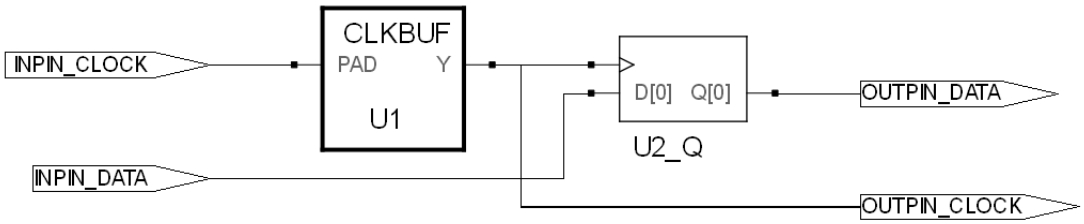

Figure 4 • **Synplify HDL-Analyst RTL-View**

Understanding "OUTPIN_CLOCK" - "OUTPIN_DATA" Path

To understand the path that we are analyzing, we need to be able to view and understand the logic that is implemented in the FPGA device. Inside Designer (Place & Route Software), the 'Netlist-Viewer' tool can be used to display the logic that is actually implemented in the silicon.

In source-synchronous clock designs, the timing relation that we are most concerned with is the one from the "OUTPIN_CLOCK" output pin to the "OUTPIN_DATA" output pin path. These two outputs and the logic connected to them are shown in Figure 5. This relation is highlighted with a pink arrow.

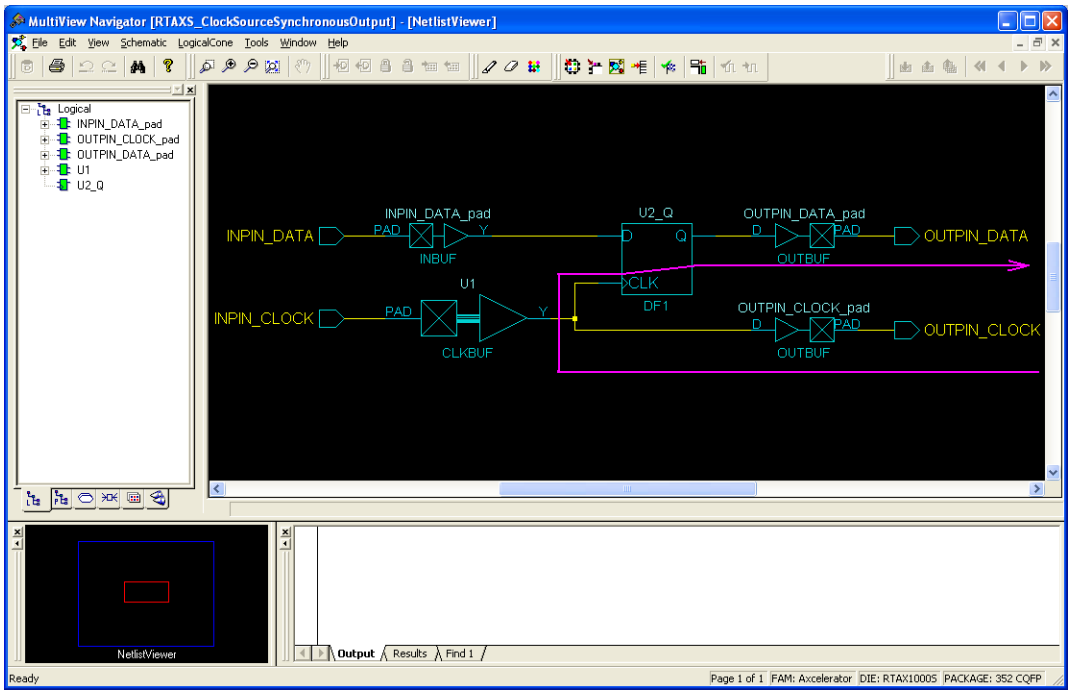

Figure 5 • **Designer Netlist-Viewer: View of Logic in FPGA**

Constraining the Timing

One of the main purposes of this application note is to describe how to ‘constrain’ the timing of the data-output pin labeled “OUTPIN_DATA” with respect to the output-clock pin labeled “OUTPIN_CLOCK”, which are both output out of the FPGA. In order to do this, we first need to define the “OUTPIN_CLOCK” as a clock signal in ‘SmartTime’, and then use this to define the timing of the “OUTPIN_DATA” signal with respect to that generated clock. The following section provides information on defining a Generated Clock.

Defining a Generated Clock

It is important to note that as far as the FPGA is concerned, it has two output signals on two output pins (“OUTPIN_CLOCK” and “OUTPIN_DATA”), and the FPGA fabric thinks these are both regular data outputs, at least until this step.

It is also important to note that the clock source that we want to use for timing analysis is labeled “OUTPIN_CLOCK”, and this is an output out of the FPGA fabric, right at the output pin of the FPGA’s package. The subtle point here is that this output pin “OUTPIN_CLOCK” is not the exact signal that is clocking the flip-flop that is generating the output data that eventually drives the “OUTPIN_DATA” output pin. However, this “OUTPIN_CLOCK” is the node we want to use as the reference point to analyze the timing of “OUTPIN_DATA”.

Since the “OUTPIN_CLOCK” signal is not the actual signal, It is the clock that generates the output data. We needed to define this node as a ‘Generated Clock’, so that we could use this “OUTPIN_CLOCK”, as a clock signal that is used to enter constraints and analyze timing against, later.

Inside the Designer tool, in the SmartTime ‘Constraints Editor’ tool GUI, a new clock is defined, as a generated clock, named “MyDefinedGeneratedClock_OUTPIN_CLOCK”. This generated clock has a starting point “INPIN_CLOCK” and an ending point “OUTPIN_CLOCK”. This is shown in the Figure 6.

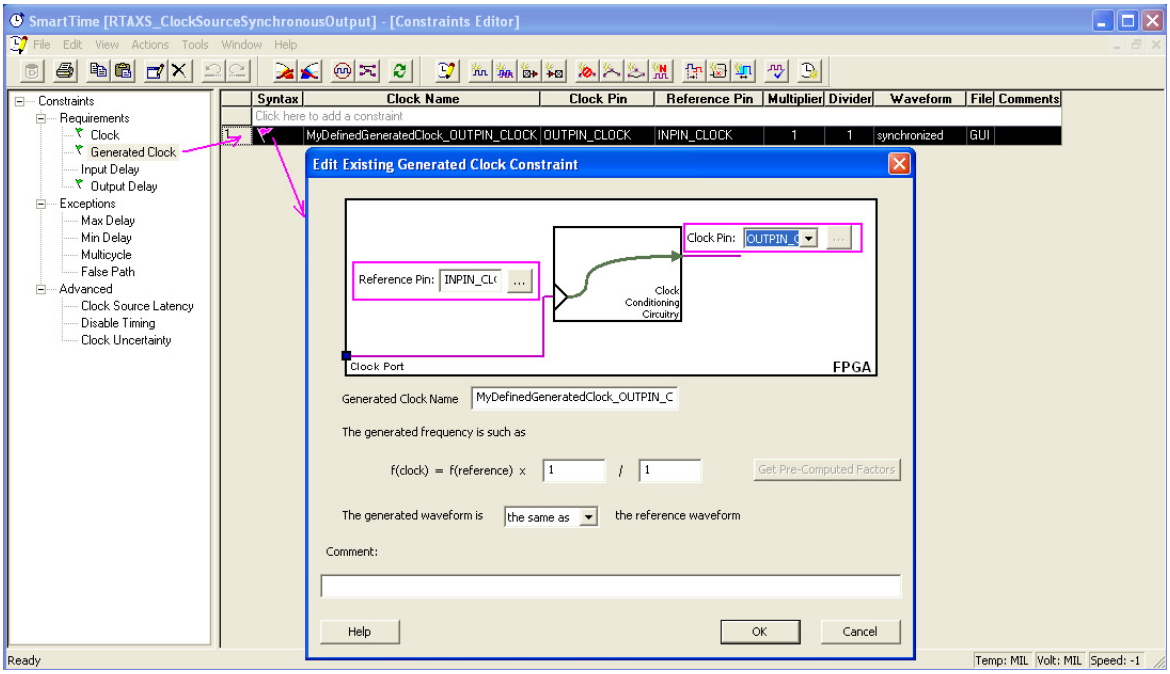

Figure 6 • **Designer SmartTime Tool: Defining a Generated Clock**

The newly defined clock now shows up under the “Generated Clock” section of the “Constraints” section.

Defining the “OUTPIN_CLOCK” to “OUTPIN_DATA” Constraint

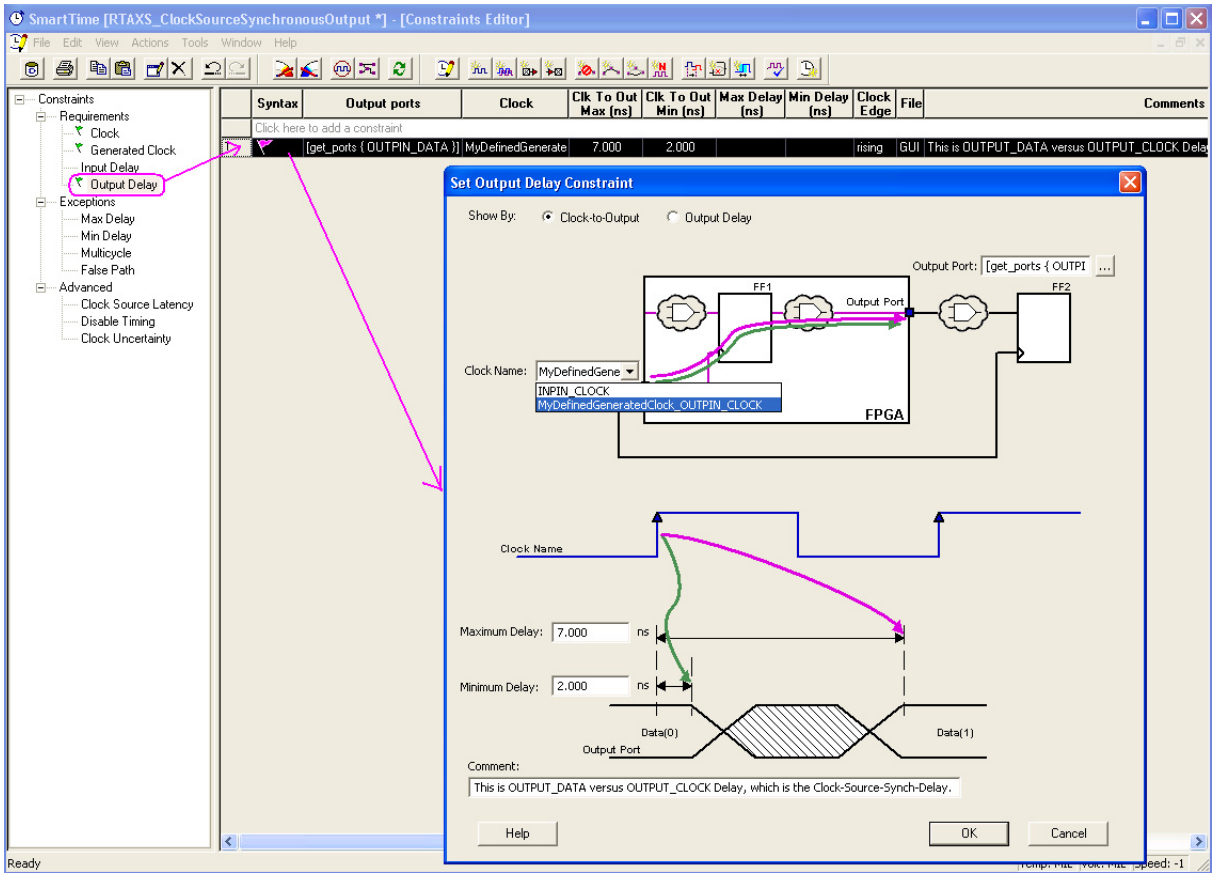

Figure 7 • **Designer SmartTime Tool: Defining a Clock-to-Output Delay**

We need to define the output pin labeled “OUTPIN_DATA”, to have minimum and maximum delay relative to the “OUTPIN_CLOCK” pin.

Inside the Designer tool, in the SmartTime ‘Constraints Editor’ tool GUI, the category “Output Delay”, and the clock constraint from the “OUTPIN_CLOCK” to “OUTPIN_DATA” outputs is defined.

When you expand this constraint, you will see that the starting point for the clock name is “MyDefinedGeneratedClock_OUTPIN_CLOCK” (as you may recall, this is the clock definition we defined in the previous step). And the output pin is named “OUTPIN_CLOCK”.

In this example, the constraint defines that the “OUTPIN_DATA” to occur a minimum of 2.0 ns after the rise edge of the “OUTPIN_CLOCK” output pin, and a maximum delay of 7.0 ns after the rise-edge of the “OUTPIN_CLOCK” output pin. This is shown in the Figure 7.

Note: The value of 2.0 ns and 7.0 ns has no specific significance. These values were chosen to illustrate how to constrain and analyze this source-synchronous clock design. Each design will have its own timing requirements.

Thus, we have now successfully constrained the timing relationship from the “OUTPIN_CLOCK” pin of the FPGA to the “OUTPIN_DATA” pin of the FPGA fabric.

Analyzing the Timing

One of the main purposes of this application note is to describe how to ‘analyze’ the timing of the data-output pin labeled “OUTPIN_DATA” with respect to the output-clock pin labeled “OUTPIN_CLOCK”, which are both output out of the FPGA fabric.

Setting Up SmartTime to Analyze Inter-Clock Domain Paths

In order to show the “OUTPIN_CLOCK” to “OUTPIN_DATA” constraint analysis, we first need to turn-on a timing-analysis switch setting in the Designer’s SmartTime tool. To do this, in Designer, select the SmartTime “Timing Analyzer” tool, and then in the toolbar, select ‘Tools’, then in the pull-down, select ‘Options…”. In the ‘SmartTime Options’ screen, select to check the radio-button (by default it not selected) for the setting ‘Include inter-clock domains in calculations for timing-analysis’. Then click ‘OK’ to accept. This is shown in the Figure 8.

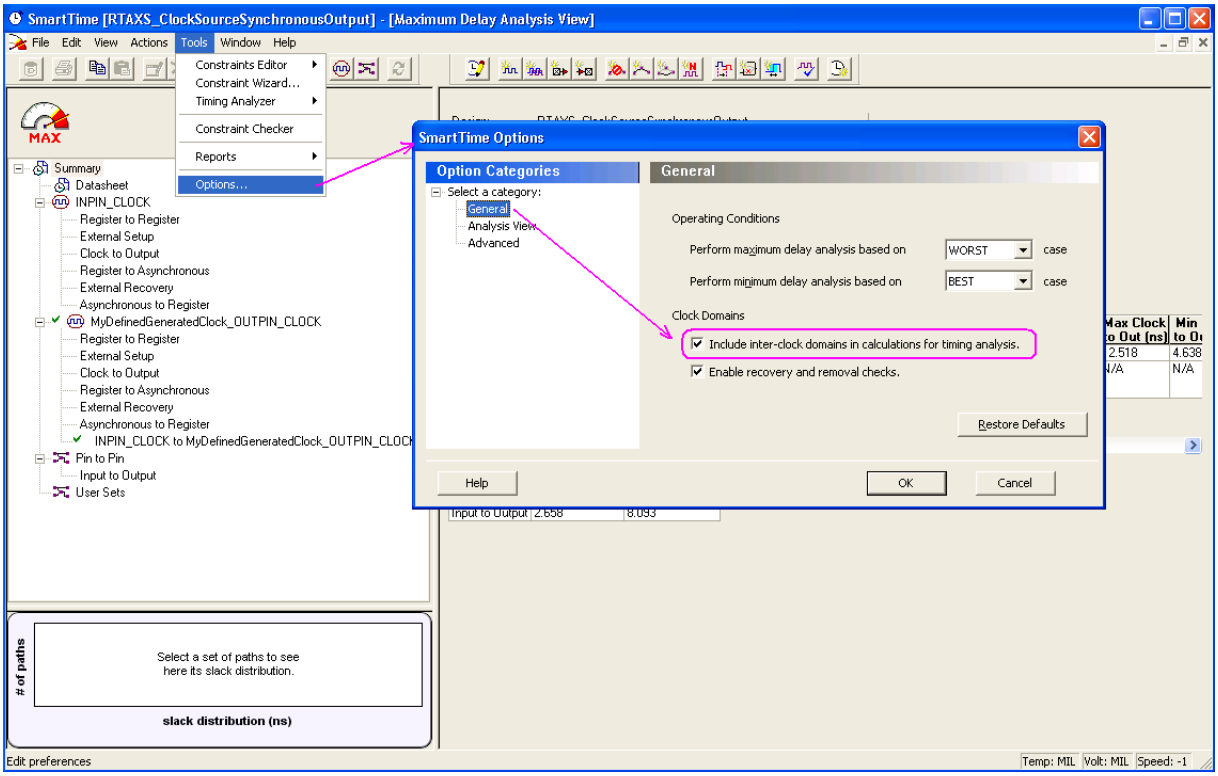

Figure 8 • **Designer SmartTime Tool: Enabling Inter-Clock Domain Analysis**

After this switch is set, you should be able to see the analysis between the two clock constraints that were defined in the previous step. In the **Maximum delay Analysis view**, both these constraints are shown in the Figure 9.

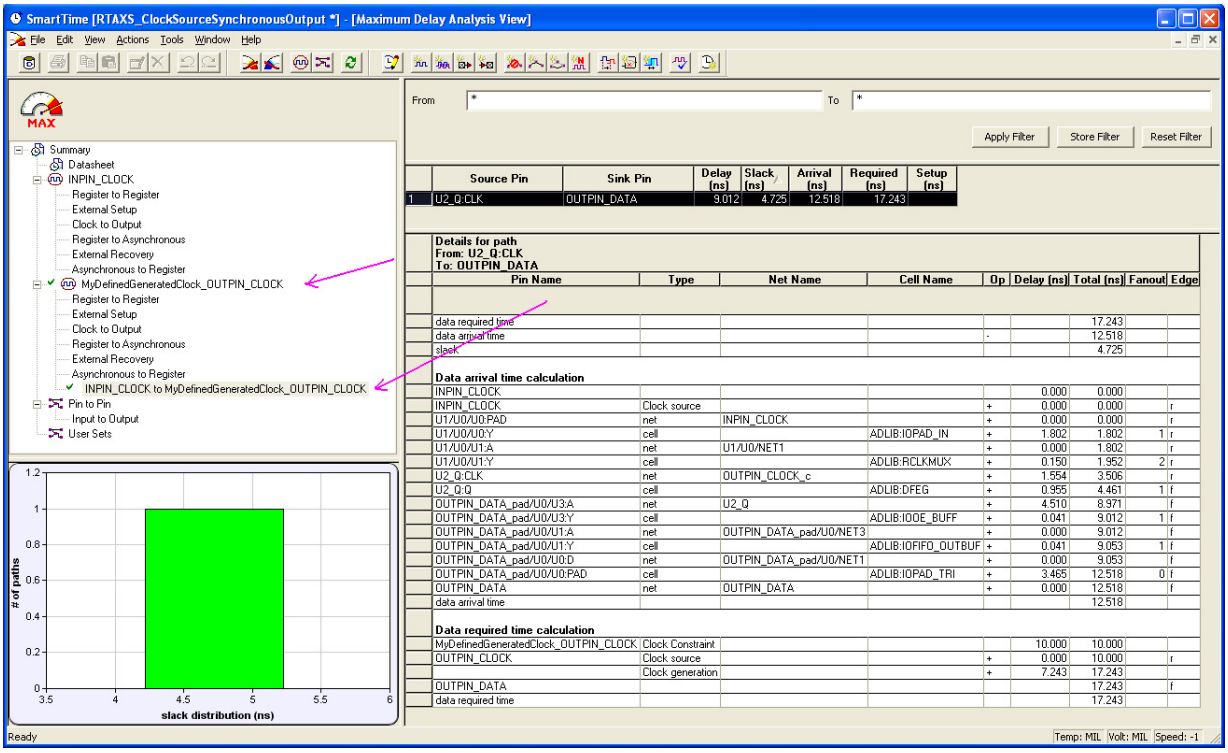

Figure 9 • **Designer SmartTime Tool: Locating the 2 Timing Constraints**

Finding the “OUTPIN_CLOCK” to “OUTPIN_DATA” Path

Here, we are interested in analyzing the constraint from the data-output pin labeled “OUTPIN_DATA” with respect to the output-clock pin labeled “OUTPIN_CLOCK”, which are both output out of the FPGA fabric.

In the SmartTime Analysis window, shown in the Figure 9, the inter-clock domain set named “INPIN_CLOCK to My_Generated_OUTPIN_CLOCK” is one that contains an analysis of this constraint. This path containing this analysis is shown with a background color of black, and has a rank of 1.

Understanding the “OUTPIN_CLOCK” to “OUTPIN_DATA” Path

To understand the path that we are analyzing, we can expand this path to see the graphical schematic equivalent of this path. To do this, double-click on the path that is shown with rank 1 (highlighted in black in Figure 9), this path will be expanded, so that all the logic elements and their respective connections are shown.

In Figure 10 on page 9, the path is shown at the bottom of the screen, and the path that we are analyzing is highlighted with a pink arrow.

Note: The path that we are analyzing in this source-synchronous clock design, is from the output pin named “OUTPIN_DATA” with respect to the output pin named “OUTPIN_CLOCK”. However, since the clock port “OUTPIN_CLOCK” does not actually drive the clock port of the flip-flop that generates the “OUTPIN_DATA” output pin signal, there is no actual logic that starts at the pin “OUTPIN_CLOCK” that drives the output “OUTPIN_DATA”, and that is why there is no logic shown that starts at the port “OUTPIN_CLOCK” in Figure 10. However, we are successfully analyzing timing on “OUTPIN_DATA” versus “OUTPIN_CLOCK”.

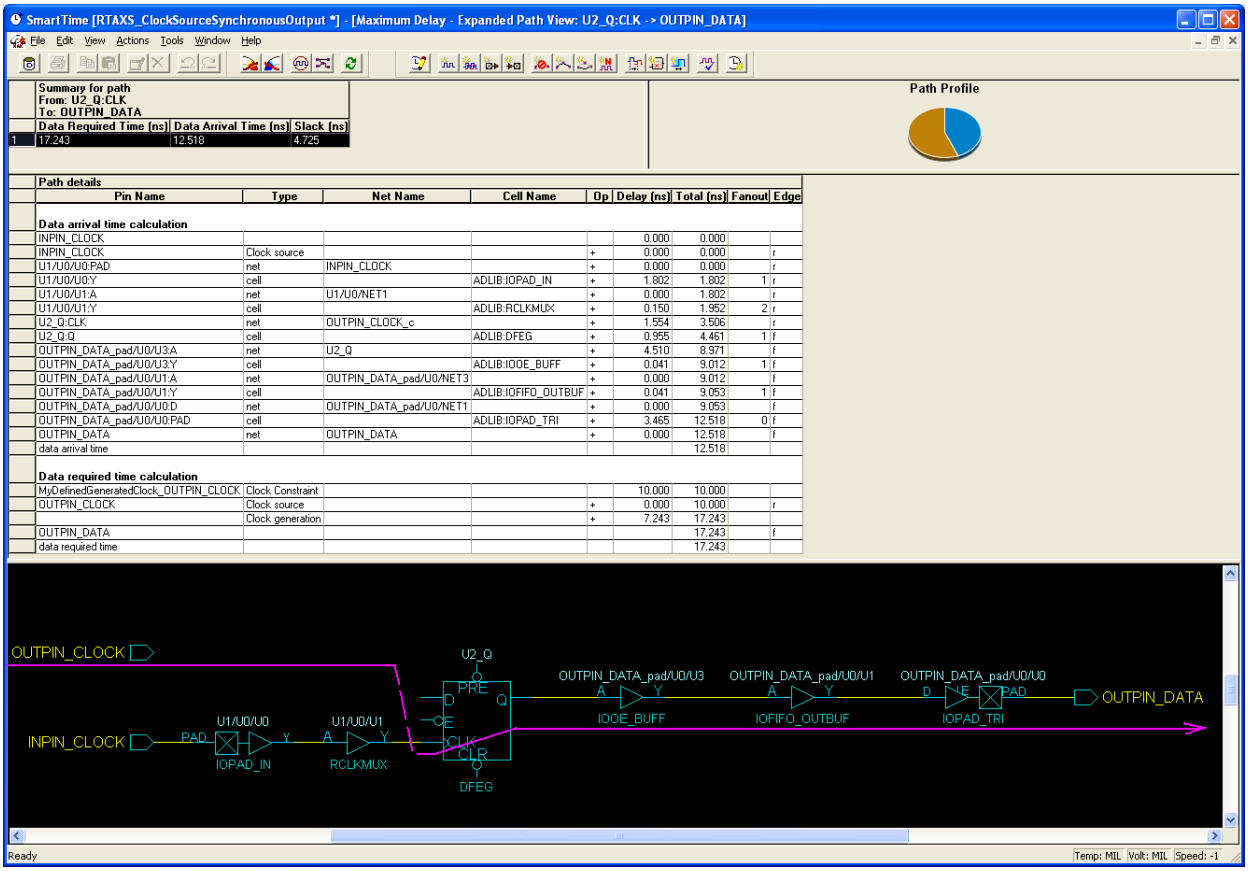

Figure 10 • **Designer SmartTime Tool: Expanding the Path**

Analyzing the Maximum Delay

In the **Maximum Delay Analysis View**, this design has met the desired constraint and has a positive slack of 4.725 ns. This slack is highlighted and shown in the Figure 11.

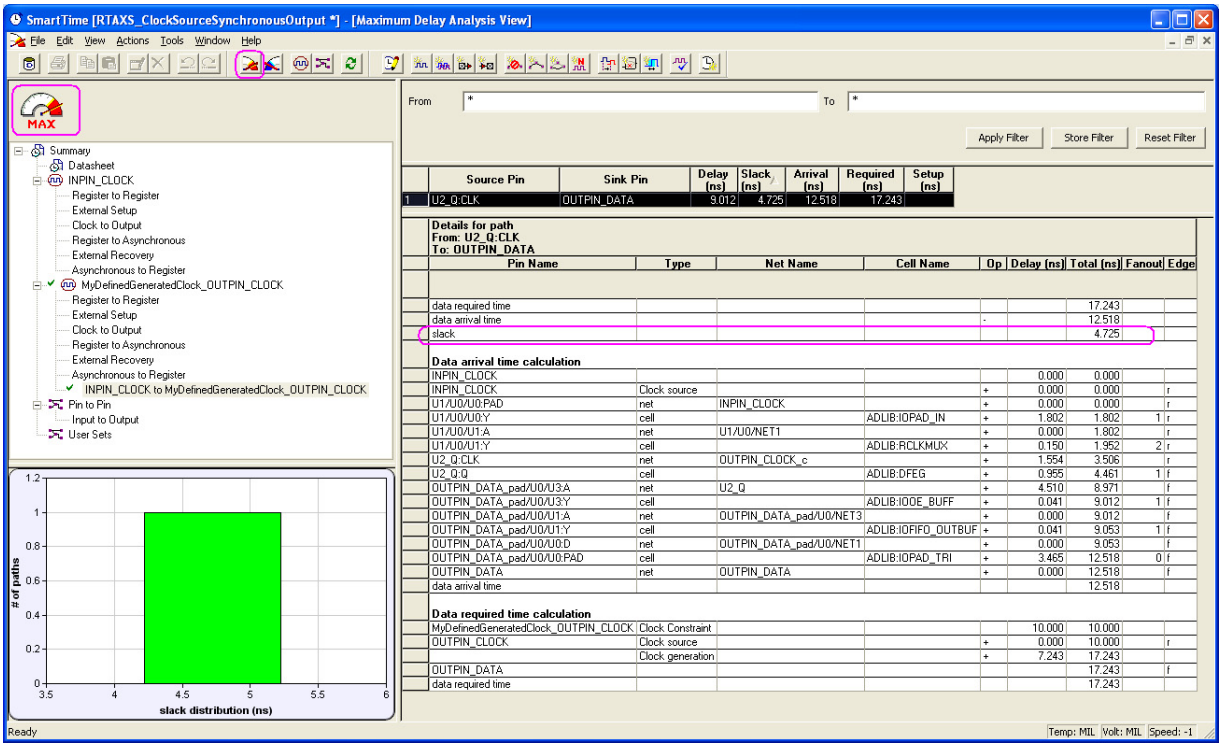

Figure 11 • **Designer SmartTime Tool: Maximum Delay Analysis View**

Analyzing the Minimum Delay

In the **Minimum Delay Analysis View**, this design has met the desired constraint, and has a positive slack of 1.980 ns. This slack is highlighted and shown in the Figure 12.

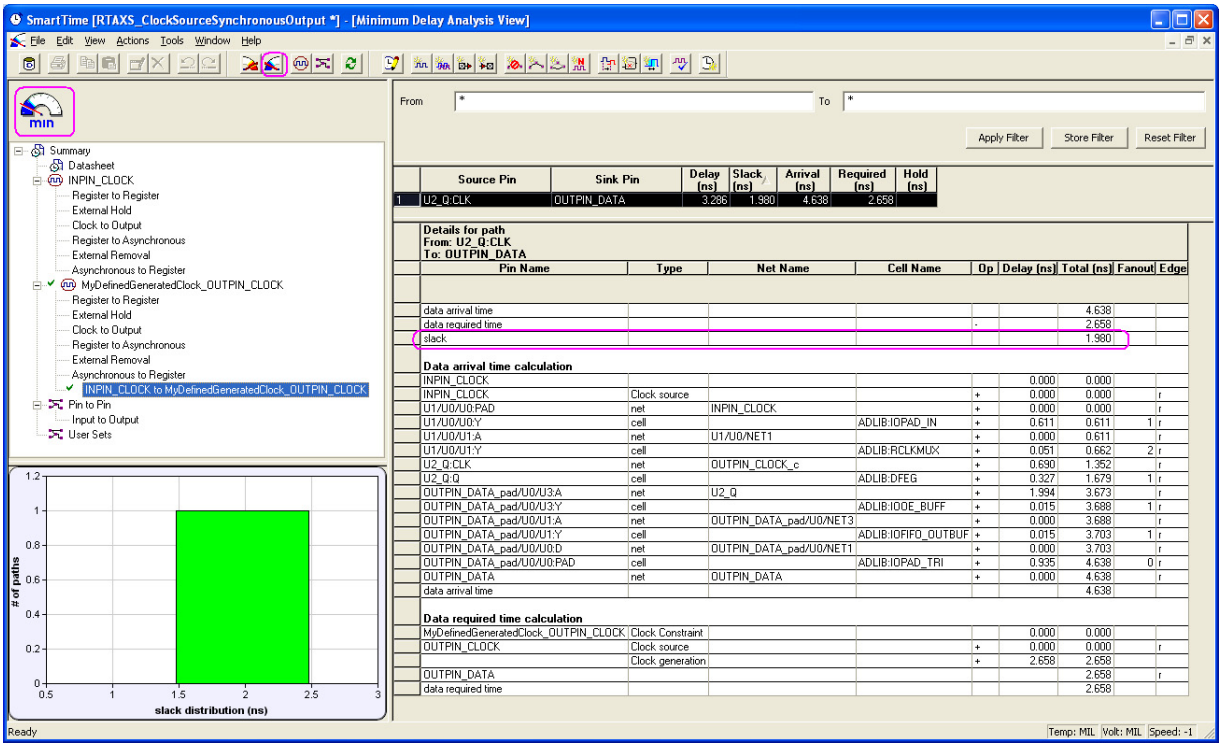

Figure 12 • **Designer SmartTime Tool: Minimum Delay Analysis View**

Conclusion

This application note shows that the digital logic designer can implement, constrain, and analyze the source-synchronous clock digital logic designs using the Microsemi Libero IDE tool. This includes designing the logic, directly constraining the source-synchronous clock path, and analyzing the source-synchronous clock path, in the Designer SmartTime tools.

Appendix A – Design Files

Design files are available on Microsemi SoC Product Groups website for download:

www.microsemi.com/soc/download/rsc/?f=ALL_AC373_DF

The design file consists of Libero IDE projects. Refer to the Readme.txt file included in the design file for directory structure and description.

Microsemi Corporate Headquarters
One Enterprise Drive, Aliso Viejo CA 92656
Within the USA: (800) 713-4113
Outside the USA: (949) 221-7100
Fax: (949) 756-0308 · www.microsemi.com

Microsemi Corporation (NASDAQ: MSCC) offers a comprehensive portfolio of semiconductor solutions for: aerospace, defense and security; enterprise and communications; and industrial and alternative energy markets. Products include high-performance, high-reliability analog and RF devices, mixed signal and RF integrated circuits, customizable SoCs, FPGAs, and complete subsystems. Microsemi is headquartered in Aliso Viejo, Calif. Learn more at **www.microsemi.com**.

51900241-0/08.11