AC444
Application Note
Implementing SpaceWire Clock and Data Recovery in RTG4 FPGAs

Microsemi
a Microchip company

Microsemi Headquarters
One Enterprise, Aliso Viejo,
CA 92656 USA
Within the USA: +1 (800) 713-4113
Outside the USA: +1 (949) 380-6100
Sales: +1 (949) 380-6136
Fax: +1 (949) 215-4996
Email: sales.support@microsemi.com
www.microsemi.com

About Microsemi

Microsemi, a wholly owned subsidiary of Microchip Technology Inc. (Nasdaq: MCHP), offers a comprehensive portfolio of semiconductor and system solutions for aerospace & defense, communications, data center and industrial markets. Products include high-performance and radiation-hardened analog mixed-signal integrated circuits, FPGAs, SoCs and ASICs; power management products; timing and synchronization devices and precise time solutions, setting the world's standard for time; voice processing devices; RF solutions; discrete components; enterprise storage and communication solutions, security technologies and scalable anti-tamper products; Ethernet solutions; Power-over-Ethernet ICs and midspans; as well as custom design capabilities and services. Learn more at www.microsemi.com.

51900444. 4.0 1/21

Contents

Figures

Tables

1 Revision History

The revision history describes the changes that were implemented in the document. The changes are listed by revision, starting with the current publication.

1.1 Revision 4.0

The following is a summary of changes made in revision 4.0 of the document:

- Updated Table 3, page 12 and Table 4, page 17 to correct the I/O bank types and update example design top-level pin names.
- Updated I/O Delay Adjustment, page 11 to explain the Libero warning seen during the design flow.
- Updated Timing Analysis of RTG4 Data Recovery Block, page 13 to use the “-all_registers” keyword.
- Updated Figure 18 on page 16.
- Updated Running the Design, page 18.
- Added Appendix 1: Programming the Device Using FlashPro Express, page 22.
- Added Appendix 2: Running the TCL Script, page 25.
- Updated Appendix 3: Design and Programming Files, page 26.
- Removed the references to Libero version numbers.

1.2 Revision 3.0

Revision 3.0 of the document was published in March 2018. In this revision, the document was updated for Libero SoC v11.8 SP2 software release. The following is a summary of changes made in revision 3.0 of the document:

- Demo design is migrated to enhanced constraint flow (ECF) in Libero SoC v11.8 SP2 software.
- Updated the document to incorporate RTG4-DEV-KIT boards which use PROTO devices in place of the discontinued Engineering Silicon (ES) units.
- Added Table 3, page 12, “Recommended SpaceWire I/O Delay Settings,” to align with SpaceWire performance limits as per *DS0131: RTG4 FPGA Datasheet.*

1.3 Revision 2.0

Updated the document for Libero SoC v11.7 software release (SAR 78009).

1.4 Revision 1.0

Revision 1.0 was the first publication of this document.

2 Implementing SpaceWire Clock and Data Recovery in RTG4 FPGAs

RTG4™ field programmable gate arrays (FPGAs) contains built-in clock recovery circuits for SpaceWire applications. The clock recovery circuit and the FPGA fabric allow easy implementation of the SpaceWire receiver block in the RTG4 FPGA device. This application note describes the usage of the clock recovery circuit in the SpaceWire receiver implementation with a reference design. The reference design is implemented in the RTG4 Development Kit.

The SpaceWire protocol is widely used to handle the payload data on-board a spacecraft. It uses data-strobe (DS) encoding, which encodes the transmission clock with the data into data and strobe signals, therefore the clock can be recovered by XORing the data and strobe lines. The recovered clock provides the clock signals used by the receiver. The clock recovery and receiver circuitry design appear to be simple, but could present serious timing closure challenges when targeted onto a generic FPGA architecture. Microsemi RTG4 FPGAs have built-in RX clock recovery blocks with good jitter tolerance for SpaceWire applications. The RX clock recovery block and the FPGA logic element allow easy implementation for the SpaceWire receiver circuit. This application note demonstrates how to use the built-in RX clock recovery block to recover the SpaceWire clock and to reliably capture the input data. It provides a design example showing SpaceWire transmit and receiver block running at 180 Mbps in the RTG4 PROTO device with STAR-Dundee FMC SpaceWire/SpaceFibre board at room temperature. The design example also includes static timing analysis for the RTG4 SpaceWire clock and data recovery block interface, which can be applied to the user design.

This application note refers to information from *UG0574: RTG4 FPGA Fabric User's Guide*, *UG0586 RTG4 FPGA Clocking User's Guide*, *RTG4 Macro Library User's Guide* and *DS0131: RTG4 FPGA Datasheet*. For more information on the SpaceWire protocol, see the ECSS-E-50-12A standard from the *European Cooperation for Space Standardization*.

2.1 Design Requirement

The following table lists the resources required to run the design.

Table 1 • **Design Requirements**

Requirement	Version
Operating System	64-bit Windows 7 and 10
Hardware	
RTG4 Development Kit (Rev B or later): • RTG4 Development Board with one RT4G150 PROTO device in a ceramic package with 1,657 pins • 12 V adapter (provided with the kit) • FlashPro4 programmer (provided with the Kit)	–
STAR-Dundee FMC SpaceWire/SpaceFibre board with 4 SpaceWire ports	–
Host PC or Laptop	
Software	
Libero® System-on-Chip (SoC)	**Note:** Refer to the `readme.txt` file provided in the design files for the software versions used with this reference design.
FlashPro Express	

Note: The STAR-Dundee FMC SpaceWire/SpaceFibre board is available from STAR-Dundee. For more information go to the web-page: *https://www.star-dundee.com/products/fmc-spacewirespacefibre-board*.

Note: The design example shows SpaceWire clock recovery block running at 180 Mbps in the RTG4 PROTO device at room temperature.See, *DS0131: RTG4 FPGA Datasheet* for RTG4 SpaceWire clock recovery block performance numbers across the full military temperature range.

Note: Libero SmartDesign and configuration screen shots shown in this guide are for illustration purpose only. Open the Libero design to see the latest updates.

2.2 Prerequisites

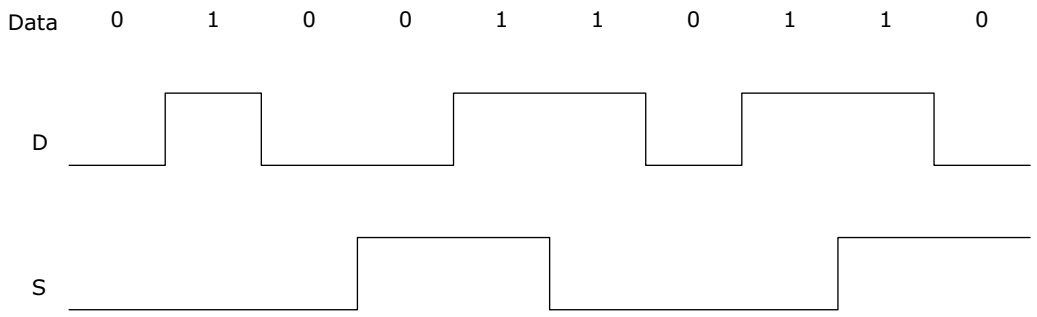
Before you start:

1. Download and install Libero SoC (as indicated in the website for this design) on the host PC from the following location: *https://www.microsemi.com/product-directory/design-resources/1750-libero-soc*
2. For demo design files download link: *http://soc.microsemi.com/download/rsc/?f=rtg4_ac444_df*

2.3 SpaceWire Coding and Signaling Overview

SpaceWire is a high-speed data link standard that provides a unified, high-speed data-handling infrastructure for connecting sensors, processing elements, mass memory units, down-link telemetry subsystems, and electrical ground support equipment (EGSE). SpaceWire links operate from 2 Mbps to 400 Mbps over a full-duplex, point-to-point serial link. SpaceWire uses DS encoding scheme, where data (D) signal follows the data bitstream; that is, high when the data bit is 1 and low when the data bit is 0. The strobe (S) signal changes state whenever the data does not change from one bit to the next. This coding scheme is illustrated in the following figure. The DS encoding and SpaceWire standard is described in ECSS-E-50-12A standard from European cooperation for space standardization. The DS encoding scheme is also used in the *IEEE Standard 1355--1995 [1]* and *IEEE 1394a (Firewire) Standard [6]*.

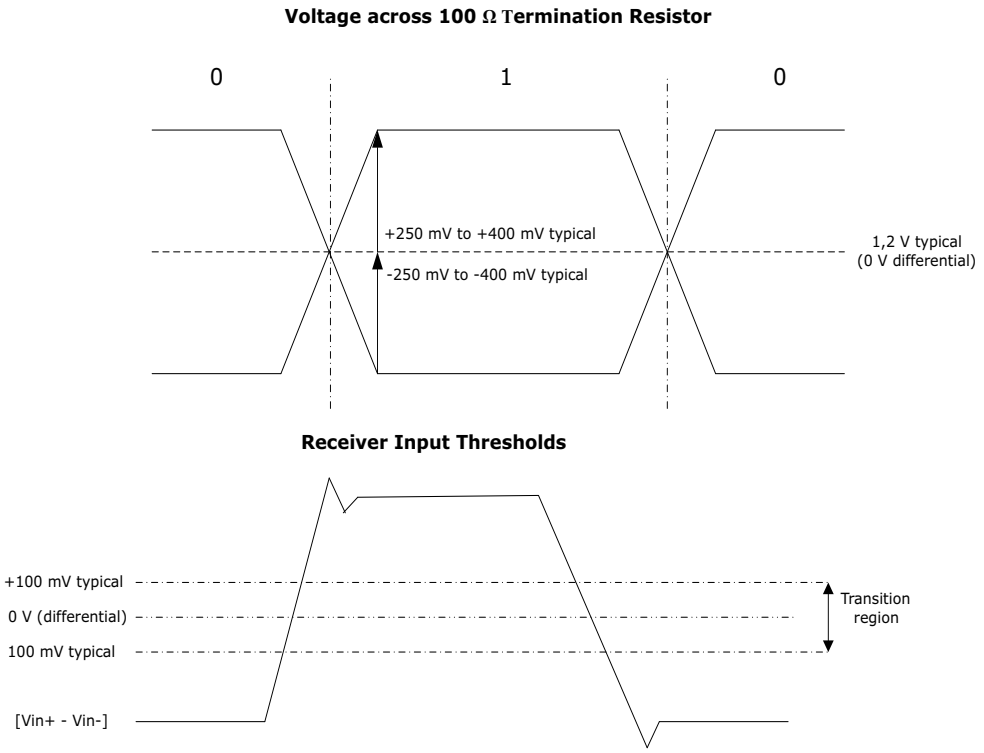
Figure 1 • **DS Encoding**



SpaceWire links use low voltage differential signaling (LVDS) for the D and S signals. LVDS employs balanced signals to provide high-speed interconnection using a low voltage swing of 350 mV typical.

The signaling levels used by LVDS are shown in the following figure. Generally, 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 signals in the opposite direction.

Figure 2 • **SpaceWire LVDS Signaling Levels**

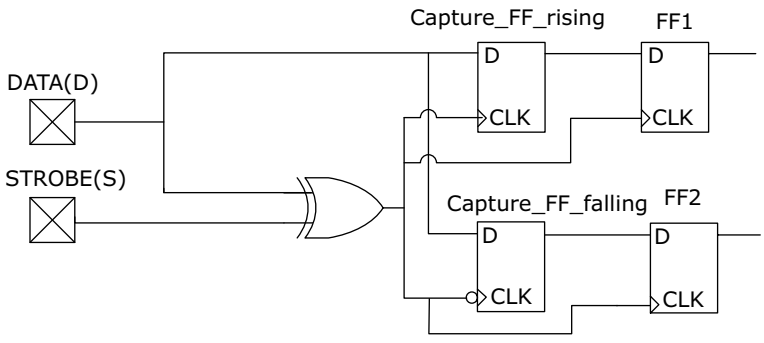


2.3.1 SpaceWire Clock and Data Recovery

SpaceWire interfaces can be implemented in generic FPGA fabric architectures, but the SpaceWire clock recovery circuitry design at the receiver side poses serious challenges specific to the FPGA device architecture. The major challenge is related to the tight timing constraints between data and strobe signals to generate a glitch free and low jitter recovered clock. An additional challenge is caused by the fact that the received data is in the clock domain of the remote SpaceWire transmitter and must be re-timed to the recovered clock. The following figure shows the logic scheme for clock and data recovery. The SpaceWire clock is recovered by XOR-ing D and S signals.

The data signal is recovered by sampling the D input on both edges of the recovered clock. In addition, the capture data is re-timed to the recovered clock domain to avoid any meta-stability issue.

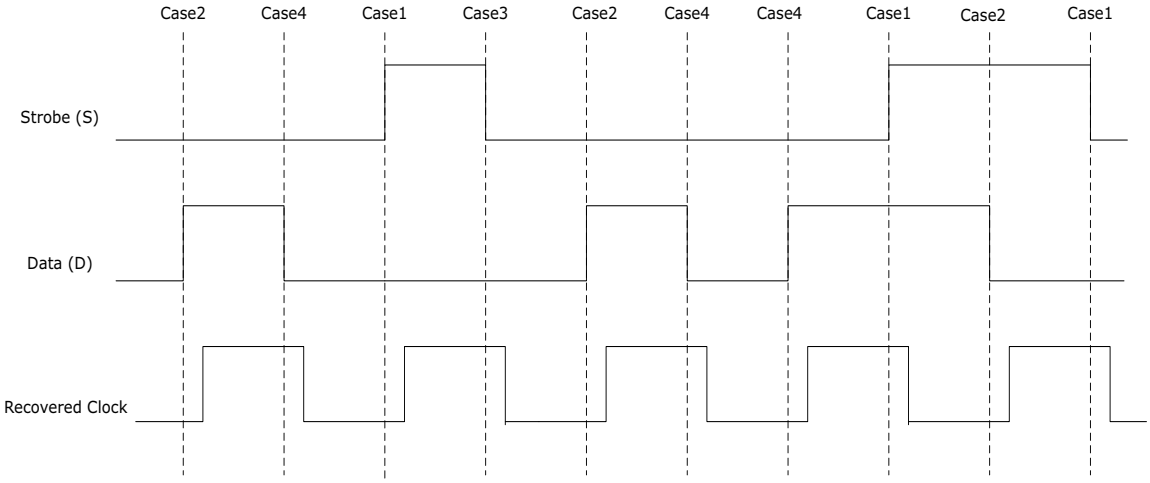
Figure 3 • **SpaceWire Clock and Data Recovery Logic**



When implementing the SpaceWire clock and data recovery block, the following cases must be considered:

- Case1: S transition, D stable
- Case2: D transition, S stable
- Case3: S transition to next S transition, D stable
- Case4: D transition to next D transition, S stable

Figure 4 • **SpaceWire Data and Strobe with Clock Recovery Scenario**



To ensure proper operation of the SpaceWire clock recovery circuitry, the user must comply with the following conditions:

1. When data is changing, the data event arrives before the clock edge. This ensures that the correct data value is sampled.

Longest (D ---> Capture_FF*:D) < Shortest (D ---> Capture_FF*:CK) - Setup (FF)

EQ1

2. When strobe is changing, a strobe event does not generate a clock edge that captures the wrong data.

Bit_Period + Shortest (D ---> Capture_FF*:D) > Longest (S ---> Capture_FF*:CK)

EQ2

The RTG4 device has hardened RX clock and data recovery blocks and includes a hardwired path for SpaceWire data input to an I/O flip-flop (IO-FF) to capture the incoming data. The I/Os include a programmable input delay setting that allows users to control the SpaceWire data path delay and easily meet the conditions listed in the preceding equations, EQ1 and EQ2.

The following sections describe implementing the RTG4 RX clock and data recovery block in detail.

2.4 RTG4 SpaceWire Clock Recovery Block Overview

The RTG4 device has hardened RX clock recovery blocks which reside in the clock conditioning circuits (CCCs). There are two blocks of RX clock recovery in each CCC block, as shown in the following figure. Each RX Clock recovery block has its own multiplexing (MUXing) logic to select two inputs for data and strobe. The SpaceWire data input is also passed to the FPGA fabric for use with SpaceWire receiver logic and allows implementation of SpaceWire protocol IP.

Figure 5 • **RTG4 SpaceWire RX Clock Recovery Block in CCC**

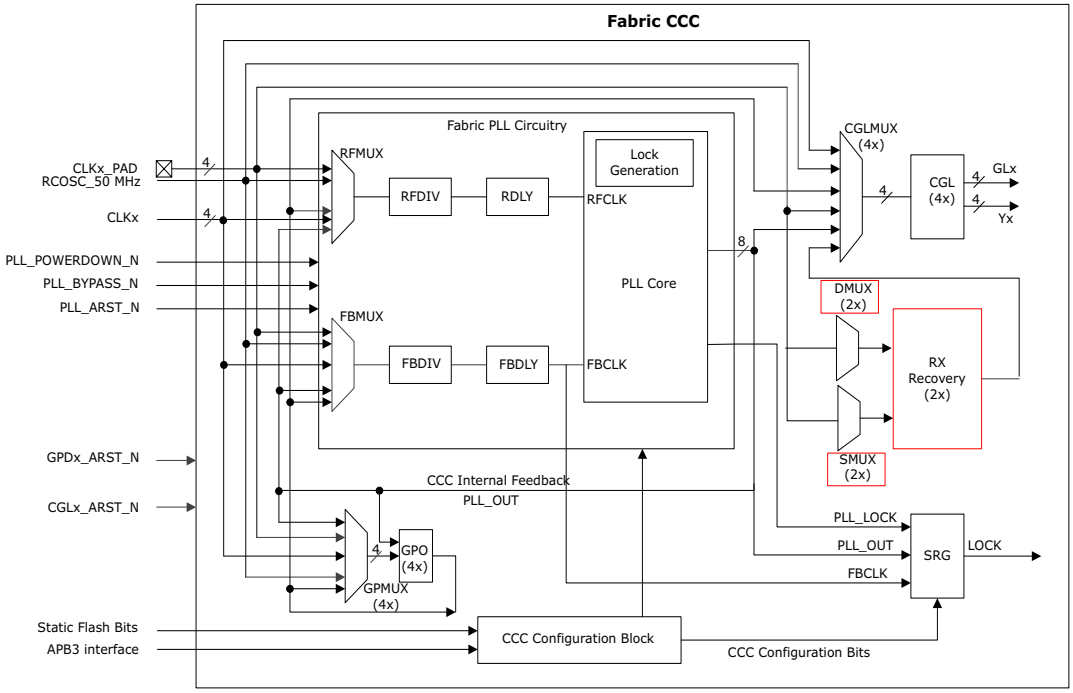
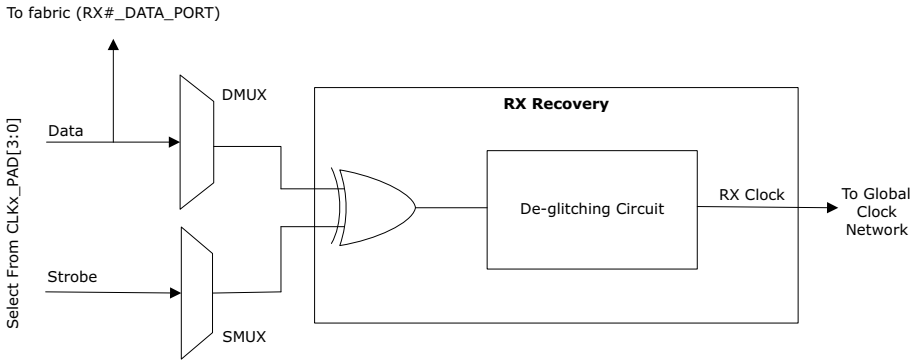


Figure 6, page 8, shows a detailed implementation diagram of the RX clock recovery block in RTG4, which includes the XOR logic. The RX clock recovery block includes a de-glitching circuit which can filter out unwanted narrow clock pulses of 600 ps or less. The de-glitching circuit is used to add filtering for either single event transient (SET) or system-level glitches. The glitch filtering circuit in the SpaceWire clock recovery circuit is always enabled. In contrast, the SET filter for fabric flip-flops are disabled by default and can be enabled in Libero SOC. Enabling the flip-flop SET filter is done globally through an option that is available in the Libero SoC project settings or by using the `set_mitigation` netlist constraint (in `.NDC` constraints) at the instance level.For more information, see the Libero SoC PDC Commands User Guide at *http://coredocs.s3.amazonaws.com/Libero/11_8_0/Tool/pdc_ug.pdf*.

The generated RX clock is radiation hardened and drives the hardened global clock network.

Figure 6 • **RTG4 SpaceWire RX Clock Recovery Block**



The data and strobe dedicated input pads can be configured to use either single ended or differential I/O standards.

The following table shows the SpaceWire pins in the RTG4150-CG1657M device. When the data and strobe dedicated input pads are configured as single-ended I/O, only the P pins are used.When not used for SpaceWire functionality, the unused N pins from the data and strobe inputs can be used by the FPGA fabric design as per their respective pin nomenclature (MSIO/MSIOD/DDRIO) and bank voltage assignment.

Table 2 • **SpaceWire Pins in RTG4 device**

Port	Direction	Description
SPWR_xyz_w_RX_STROBE_[P/N]	Input	Differential Input Strobe signal from I/O pad
SPWR_xyz_w_RX_DATA_[P/N]	Input	Differential Input Data signal from I/O pad

Note: xy represents individual SpaceWire block located at specific chip corner—NE, SE, SW, or NW.

Note: z is CCC number of either 0 or 1 for the corresponding corner of the RTG4 chip.

Note: w refers to one of the two possible input pins associated with each of the two SpaceWire recovery blocks per CCC - SPWR_xyz_[0,1].

RT4G150 production devices support up to 16 sets of SpaceWire data and strobe input pins (14 sets in MSIO/MSIOD and 2 sets in DDRIO banks) to implement SpaceWire data and clock recovery block, while RTG4 ES devices only support up to 12 sets of SpaceWire data and strobe input pins. The MSIO/MSIOD banks support true low-voltage differential signaling (LVDS) receivers, but DDRIO banks do not support true LVDS receivers. Therefore, when using the SpaceWire input pins on the DDRIO banks, the user needs to implement these inputs as single-ended LVCMOS or as pseudo differential (voltage referenced) SSTL and HSTL P/N pairs. The SpaceWire data and clock placed in DDRIO banks have limited performance compared to the SpaceWire data and clock placed in MSIO/MSIOD bank, as shown in the SpaceWire performance data table in *DS0131: RTG4 FPGA Datasheet.* The performance limitation is related to the way the programmable input delay settings are applied globally to the DDRIO Banks. It is not possible to delay the SpaceWire data input relative to the recovered clock since the delay setting is applied to all inputs on the bank. Therefore, on DDRIO banks, the data inputs cannot be delayed without delaying the clock inputs.

Note: The DDRIO buffers do not have built-in SET mitigation like the MSIO/MSIOD buffers.

*S*ome of the pin assignments are changed in the production devices, See“*SpaceWire Pin Mapping from RTG4 ES/MS Silicon to PROTO/Flight Silicon* for more information. Pin assignment must be carefully done, when migrating from RT4G150 ES silicon to RT4G150 production silicon.

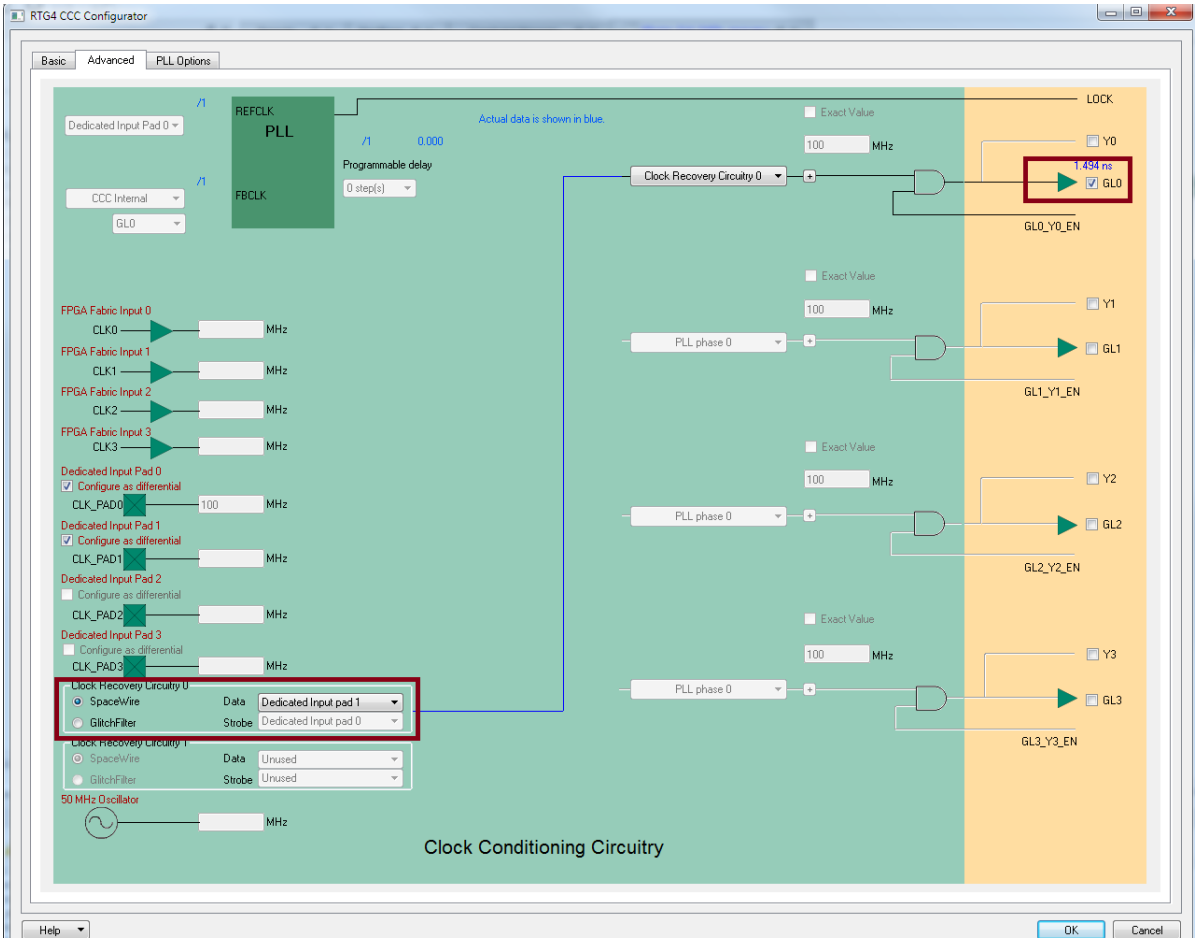
See the Appendix 4: SpaceWire Pin list, page 27 for the 16 sets of SpaceWire data and strobe input pins in RTG4 production device.

2.4.1 Using RTG4 SpaceWire Clock Recovery Block

Designers need to use the RTG4 CCC configurator in the Libero SoC software to configure the RX clock recovery block. The RX clock recovery block generates the RX clock that can drive any of the four global outputs from the CCC global outputs. The following steps describe how to configure the RTG4 SpaceWire Clock Recovery block, design flow proceeds from right to left in the GUI:

1. Open the RTG4 CCC configurator in the Libero SOC and click the **Advanced** tab, as shown in the following figure.

Figure 7 • **SpaceWire RX Clock Recovery Block Selection in CCC Configurator**



2. Select the desired output clocks from the four different global clocks—GL0, GL1, GL2, and GL3. In addition, user can also select four core clocks—Y0, Y1, Y2, and Y3. It is recommended to select the global clock when implementing the SpaceWire RX clock recovery block, as the global networks are radiation hardened.
3. For each selected output clock, select the reference input clock, Spacewire data input pad: Dedicated Input pad1 or Dedicated Input pad3.
4. Select the data pad as “**Dedicated Input Pad 1" or** “**Dedicated Input Pad 3”**. The strobe pad is automatically selected.
5. If differential inputs are used for STROBE and DATA pins, the corresponding “**Configure as differential”** option beside the **Dedicated Input Pads** is selected.
6. Click **OK** to generate the RTG4 CCC configurator.

The dedicated input pad for data and strobe can also be configured as either single-ended or differential I/O standard. For differential option, selecting one dedicated pad for the data signal automatically selects the corresponding pad as it is a pair of P and N pads. See, *RTG4 Clock Conditioning Circuit with PLL Configuration Guide* for more information on the RX clock recovery options.

2.4.2 Implementing the SpaceWire Receiver using the RTG4 Recovery Block

The following figure, RTG4 SpaceWire Clock and Data Recovery Block Implementation, shows a basic SpaceWire data recovery block implementation in the RTG4 device. The implementation requires using the hardened RX clock recovery block to recover the clock and the DDR_IN macro to capture data on both the rising and falling edges.The recovered clock and recovered data can then be used with a SpaceWire controller IP to implement the rest of the SpaceWire receiver logic. The clock and data recovery block use hardwired delay and the respective delays from the data and strobe input to clock generation are almost the same. Implementing the SpaceWire receiver requires the data arrival event, before the clock edge is generated from the same data and strobe events, to prevent incorrect capture of data, as shown in EQ1 and EQ2. Users need to use the I/O delay settings based on their design performance requirement and the guidance shown in Table 3, page 12, to satisfy EQ1, and EQ2 as described in I/O Delay Adjustment, page 11.

Figure 8 • **RTG4 SpaceWire Clock and Data Recovery Block Implementation**

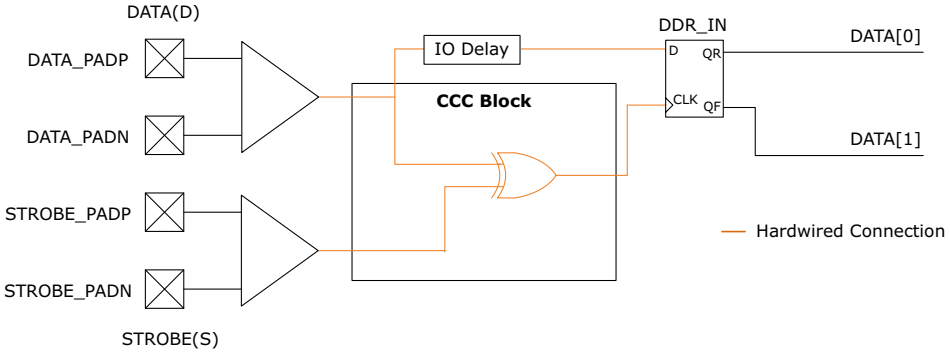
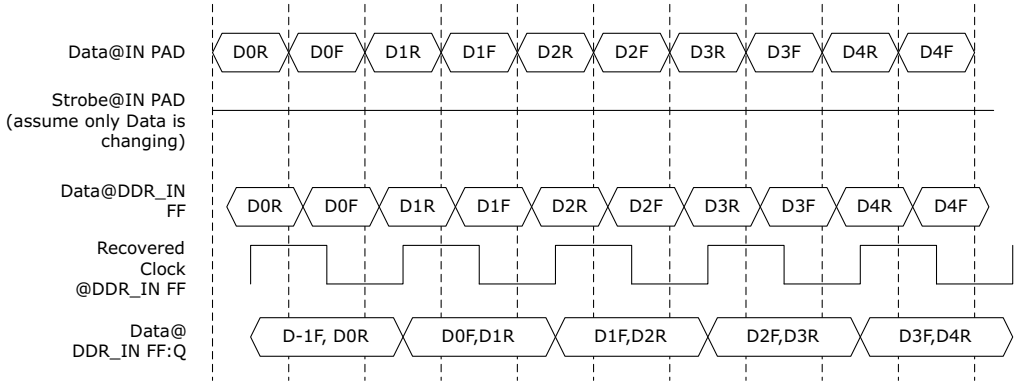


Figure 9 • **RTG4 SpaceWire Clock and Data Recovery Waveforms**



The following figures show data recovery block and waveforms with the added registers to recover the original data.

Figure 10 • **Data Recovery Block**

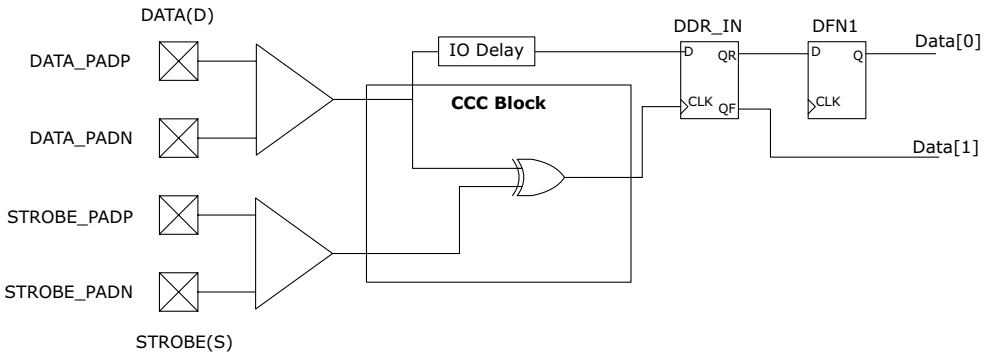
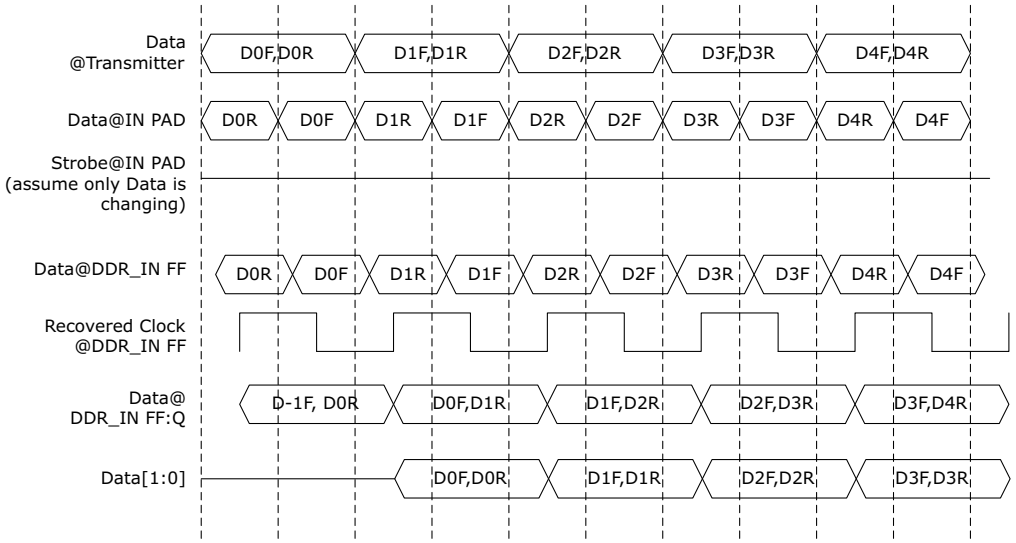


Figure 11 • **Data Recovery Waveforms**



2.4.3 I/O Delay Adjustment

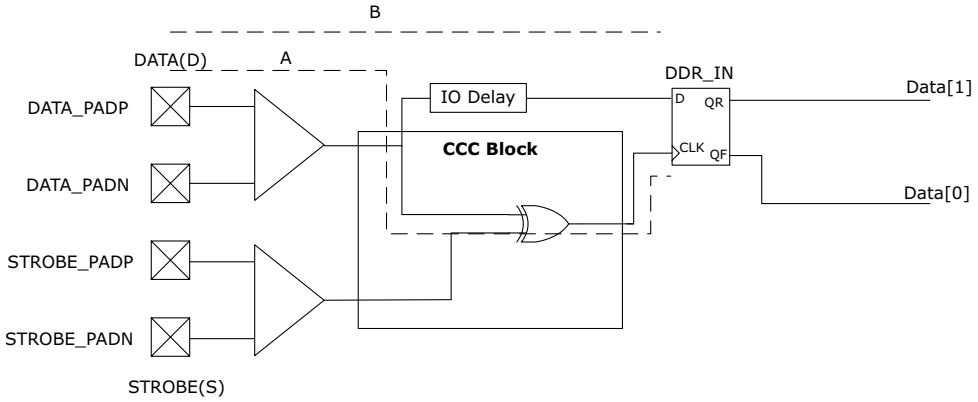
The RTG4 SpaceWire recovered clock is routed to DDR_IN macro through the global network. The user may need to adjust the I/O delay setting to match the clock insertion delay, so that the SpaceWire recovered clock can sample the correct data. The I/O delay adjustment option is only available for SpaceWire clock recovery circuits connected to MSIO/MSIOD banks. Programmable input delay settings can be made by editing the I/O PDC file using set_io constraints with the -IN_DELAY <##> argument or by using the I/O editor in the Libero SoC constraints manager. For an example of these constraints, see the associated demo design project. For the SpaceWire clock recovery circuits connected to DDRIO bank, the maximum speed which can be achieved is listed in the *DS0131: RTG4 FPGA Datasheet*.

Table 3, page 12 provides the recommended I/O delay settings to achieve the SpaceWire performance rates listed in, *DS0131: RTG4 FPGA Datasheet*. Figure 12, page 12, shows the I/O delay calculation for reference.

The user must ensure that the data arrives at the DDR_IN register before the recovered clock arrives and that the recovered clock does not sample the wrong data. The user can compute the data path and clock path delay and then use the recovered clock period to choose the I/O delay.

The I/O delay settings provided in the following table achieve the SpaceWire performance rates listed in *DS0131: RTG4 FPGA Datasheet*.

Figure 12 • **SpaceWire I/O Delay Calculation**



Diff_Delay = (DATA-to-DDR_IN:CLK Delay) - (DATA-to-DDR_IN:D Delay) - Tsu = A-B- Tsu

IO_Delay_Attribute = Diff Delay % (recovery clock period)

The recommended SpaceWire I/O Delay Settings are listed in Table 3, page 12.

The recommended I/O Delays have been chosen such that the recovered SpaceWire clock edges are centered within the SpaceWire data window. These I/O delay settings have also been validated in silicon testing across process, voltage, and temperature (PVT) variations and thus used to calculate the SpaceWire frequency and data rate limits in the RTG4 Device datasheet. These programmable input delay settings limit the data rate of the SpaceWire receiver interface, when implemented with the hard recovery block, to the published datasheet limits for STD and -1 speed grade devices.

Note: Designers may find that these I/O delay settings are conservative, and that it's possible to further tune the I/O delays, using the methodology above, to achieve faster performance as reported by SmartTime 4-corner timing analysis. With this faster performance, the data transitions will move closer to the recovered clock edges. Microsemi strongly recommends using the published SpaceWire I/O delay settings in the following table. Customers are responsible for validating whether I/O delay settings other than the following settings will work in their system.

Table 3 • **Recommended SpaceWire Programmable Input Delay Settings**

CCC Location	CCC Dedicated Clock Input #	SpaceWire Recovery Circuitry	I/O Bank Type	SPWR DATA Input Pin # CG1657 (P / N)	SPWR DATA I/O Delay Setting SET Off	SPWR DATA I/O Delay Setting SET On
SW0	1	0	MSIOD	N10 / N11	28	11
SW1	1	0	MSIOD	N8 / P8	28	14
NW1	1	0	MSIOD	V11 / V10	45	25
SE0	1	0	MSIOD	N34 / P34	29	14
SE1	1	0	MSIOD	N32 / M32	27	12
NE0	1	0	MSIOD	V31 / V32	41	23

Table 3 • **Recommended SpaceWire Programmable Input Delay Settings *(continued)***

NE1	1	0	MSIOD	AA34 / AB34	39	23
NW0	1	0	MSIO	AA8 / AB8	40	22
SW0	3	1	MSIO	H10 / J9	28	12
SW1	3	1	MSIO	K15 / K14	31	14
NW1	3	1	MSIOD	AB5 / AB6	39	20
SE0	3	1	MSIO	F27 / G27	34	17
SE1	3	1	MSIO	G29 / H29	25	13
NE0	3	1	MSIOD	AB37 / AB36	35	19
NE1	3	1	DDRIO	AC39 / AD39	0	0
NW0	3	1	DDRIO	AC3 / AD3	0	0

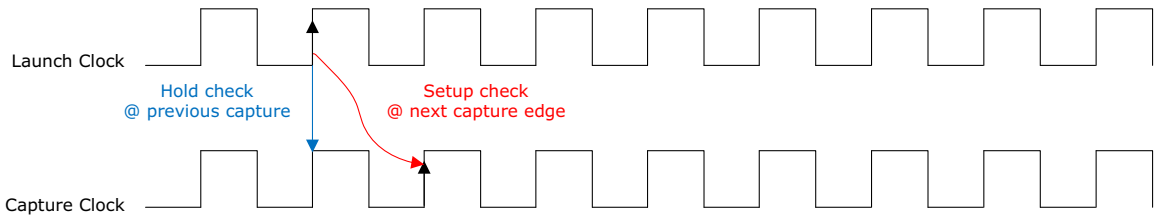
During Place and Route with programmable input delay enabled on the SpaceWire DATA input pins, Libero SoC will report the warning shown below. This warning can be safely ignored for this application. The warning is generated because the SpaceWire data input is branched into two nets. One branch is sent into the SpaceWire clock recovery block (XOR), so it can be used along with the strobe input to recover the SpaceWire clock. The other branch is passed out of the CCC and sent to local FPGA fabric routing for data capture by user logic in the FPGA fabric or I/O Flip-Flop (IOFF). The warning tells the user that the SpaceWire DATA input branch sent to the CCC's SpaceWire clock recovery circuit via a dedicated hardwired connection would not be delayed by the user's programmable input delay setting. Nets using a hardwired connection from the input buffer physically bypass the input delay taps and thus can not be affected by the programmable input delay. In contrast, the branch which is routed into the fabric on local routing will be delayed based on the user's programmable input delay setting. This aligns well with the goal of Figure 10, page 11, where the I/O Delay is only applied to the fabric data input path to compensate for the clock recovery through the CCC and global buffer insertion delay. Furthermore, the SpaceWire clock recovery circuit must XOR the aligned STROBE and DATA inputs because a delayed version of the DATA input would make it unsuitable for the clock recovery operation. Therefore, this warning can be ignored for the SpaceWire clock and data recovery application.

```
Warning: When MSIO/MSIOD port 'CLK3_SPWR_DATA_PADP' directly drives a global
or CCC using a hardwired connection, the programmable input delay taps
(IN_DELAY) are ignored on this specific path because the hardwired connection
physically bypasses the delay element. The delay taps will still affect
routed connections from this port to the fabric.
```

2.4.4 Timing Analysis of RTG4 Data Recovery Block

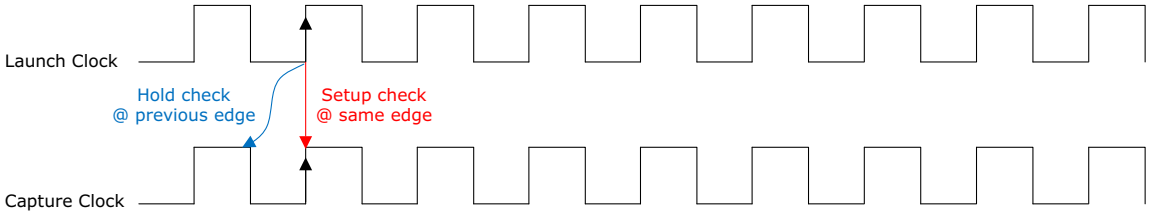
SpaceWire uses a DS encoding scheme that encodes the transmission clock with the data into data and strobe. The clock is recovered by XOR-ing the data and strobe lines together and is used to capture the data input. To ensure proper operation of the SpaceWire clock recovery circuit, the launch and capture of data must be on the same clock edge. The following figure shows a typical single clock cycle path setup and hold check window.

Figure 13 • **Typical Single Clock Cycle Path - Setup and Hold Check Window**



The following figure shows the SpaceWire cycle path setup and hold check window.

Figure 14 • **SpaceWire Clock Cycle Path Setup and Hold Check Window**



The following SDC constraint example demonstrates how to setup the user timing constraints for SpaceWire clock and data recovery logic before running 4-corner timing analysis with SmartTime. Designers must follow this format and substitute the timing parameters to match their clock period. For a complete timing constraint example, refer to the SDC constraints provided in the design files linked in Appendix 3: Design and Programming Files, page 26.

Note: The following SDC constraint example uses SpaceWire recovery clock of 100 MHz to align with the maximum frequency listed for a -1 speed grade device. For more information, see the *DS0131: RTG4 FPGA Datasheet*. The example design files contain the SDC constraints for a 90 MHz clock to align with the published limit for a STD speed grade device.The clock period needs to be adjusted based on the design requirements.

#SpaceWire clock and data recovery block constraint file

```
set spwr_clock_frequency 100
set clock_period 10
set half_clock_period 5
```

#Clock Constraints

Define clock constraint on STROBE PIN

```
create_clock -name { CLK0_SPWR_STROBE_PADP } -period 10 \
-waveform { 0.000 5 } { CLK0_SPWR_STROBE_PADP  }
create_clock -name { CLK0_SPWR_STROBE_PADN } -period 10 \
-waveform { 5 0.000 } { CLK0_SPWR_STROBE_PADN  }
```

#Input Delay Constraints

Identify clock associated with the DATA pins

```
set_input_delay 0.00 -clock { CLK0_SPWR_STROBE_PADP } \
[get_ports { CLK1_SPWR_DATA_PADN CLK1_SPWR_DATA_PADP }]
set_input_delay 0.00 -clock { CLK0_SPWR_STROBE_PADN } \
[get_ports { CLK1_SPWR_DATA_PADN CLK1_SPWR_DATA_PADP }]
```

#Max / Min Delay Constraints

```
# Max Delay = 0 for same edge setup check
set_max_delay 0.0 -from \
[get_ports { CLK1_SPWR_DATA_PADN CLK1_SPWR_DATA_PADP }]\
-to [all_registers]
```

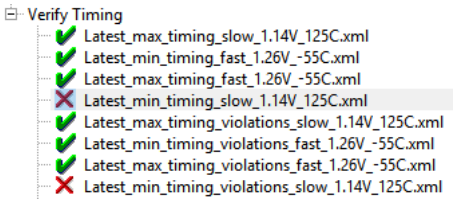
 # Min Delay = -(Clock Period/2)

```
set_min_delay -5 -from \
[get_ports { CLK1_SPWR_DATA_PADN CLK1_SPWR_DATA_PADP }]\
-to [all_registers]
```

When performing timing analysis of the SpaceWire clock and data recovery block shown in Figure 10, page 11, the designer must ensure that 4-corner timing analysis is performed. This ensures that timing violations at all corners are captured.

The following figure shows an example of the Libero SoC reports window indicating minimum delay timing violations in worst case conditions.

Figure 15 • **Libero SoC Reports Window with Timing Violations**



The typical timing violation observed with the SpaceWire clock and data recovery block is an external hold violation on the SpaceWire data input path as shown in the following figure.

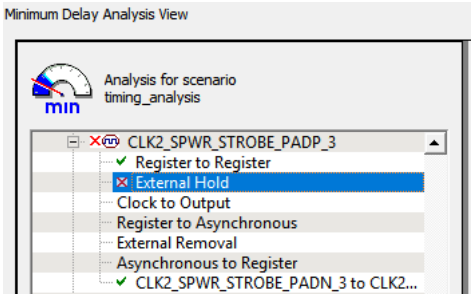
Figure 16 • **External Hold Violation on the SpaceWire Data Input Path**

SET External Hold

	From	To	Delay (ns)	Slack (ns)	Arrival (ns)	Required (ns)	Hold (ns)	External Hold (ns)
Path 1	CLK3_SPWR_DATA_PADP_3	Verify_Block_9/DDR_IN_0:D	5.536	-0.512	5.536	6.048	-0.091	5.512
Path 2	CLK3_SPWR_DATA_PADN_3	Verify_Block_9/DDR_IN_0:D	5.536	-0.512	5.536	6.048	-0.091	5.512

In the SmartTime GUI, the timing violation can also be seen by opening the corresponding analysis view (min or max delay) at the corresponding corner / operating conditions as shown in Figure 15, page 15. The following figure shows the SmartTime GUI with an external hold violation. If the user design constrains the required clock frequency to be within the published SpaceWire recovery block performance limits, and uses the recommended programmable input delay settings, the static timing analysis should be met at all four corners.

Figure 17 • **SmartTime GUI**



2.5 Design Example

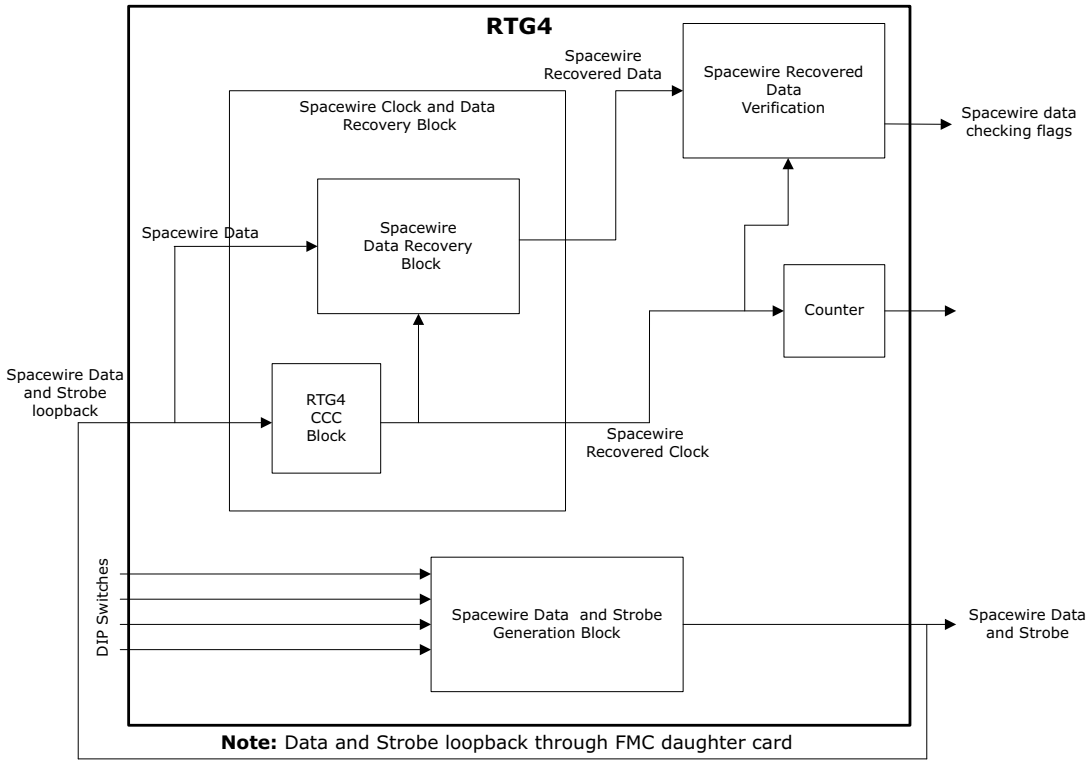
This application note provides the following design example, which shows RTG4 SpaceWire RX Clock Recovery block running at 180 Mbps.

This design has SpaceWire signal generator that generates the SpaceWire data and strobe. The SpaceWire signals are looped back using a SpaceWire FMC daughter card connected to the RTG4 Development Kit. The RX clock recovery block uses the transmitted SpaceWire signals and generates the clock. This clock is used to sample the incoming data. The sampled data is verified against the expected data.

Figure 18, page 16 shows the top-level block diagram. The top-level design example includes the following sub-blocks:

- **SpaceWire data and strobe generation block**: Generates SpaceWire data and strobe signals. The data is a counter pattern. A CCC is used to generate data at the desired rates. It also includes a switch sync and register block sub-block, that synchronizes the dual in-line package (DIP) switch inputs and configures the register block to send control signals to SpaceWire data and strobe generation block.
- **SpaceWire clock and data recovery**: This block includes the SpaceWire RX clock recovery block to generate the SpaceWire clock from the SpaceWire signals. It also samples the incoming data using SpaceWire recovered clock.
- **SpaceWire recovery data verification block**: This block compares the sampled incoming data against the expected SpaceWire data.
- **Counter**: A 28-bit counter clocked by the SpaceWire recovered clock.

Figure 18 • **Top-Level Block Diagram**



The following table shows the top-level interface signals.

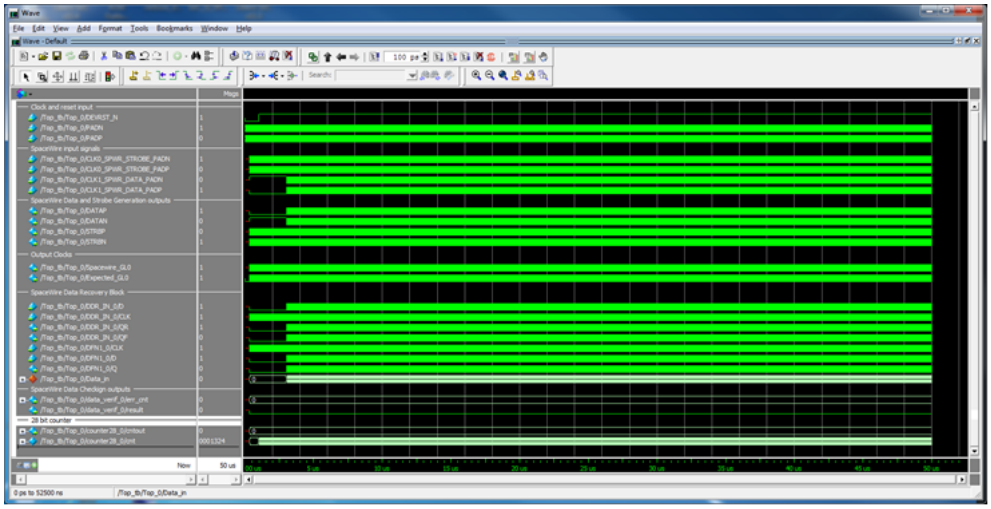
Table 4 • **Top-Level Interface Signals**

Signal	Direction	Description
CLK0_SPWR_STROBE_PADP	Input	Differential Input Strobe signal from I/O pad (PADP)
CLK0_SPWR_STROBE_PADN	Input	Differential Input Strobe signal from I/O pad (PADN)
CLK1_SPWR_DATA_PADP	Input	Differential Input Data signal from I/O pad (PADP)
CLK1_SPWR_DATA_PADN	Input	Differential Input Data signal from I/O pad (PADN)
Cntout	Output	Counter using SpaceWire recovered clock
DATAP	Output	SpaceWire Differential Data output (PADP)
DATAN	Output	SpaceWire Differential Data output (PADN)
STRBP	Output	SpaceWire Differential Strobe output (PADP)
STRBN	Output	SpaceWire Differential Strobe output (PADN)
DEVRST_N	Input	Device reset Input
DIP1	Input	DIP switch1, used to initiate SpaceWire signal counter generation
DIP2	Input	DIP switch2, used to insert error pattern in SpaceWire signal counter generation
DIP3	Input	DIP switch3, used to insert same data and strobe pattern in SpaceWire signal
DIP8	Input	DIP switch8, start the APB state machine
Expected_GL0	Output	Expected SpaceWire clock
CCC_0_CLK3_PADP	Input	Differential clock input space data and strobe generation block (PADP)
CCC_0_CLK3_PADN	Input	Differential clock input space data and strobe generation block (PADN)
Recovered_data	Output	SpaceWire recovered data
rd_err	Output	APB read error
result	Output	Compare SpaceWire recovered data and expected data
Spacewire_GL0	Output	Actual SpaceWire clock
wr_err	Output	APB write error
err_cnt	Output	Non-sticky error counter that toggles whenever data verification fails after a successful link up.

2.5.1 Simulating the Design

The reference design includes testbench to simulate the design in the Libero SoC software. The testbench provides the main reference clock for SpaceWire data and strobe generation block. It also loopbacks the SpaceWire data and strobe signals. The design waits until DIP1 switch input level is high. Once it is high, it sends SpaceWire data and strobe. The SpaceWire clock and data recovery block recover the clock and data. The SpaceWire data checking block does the final verification.

Figure 19 • **SpaceWire Design1 Simulation Waveform**

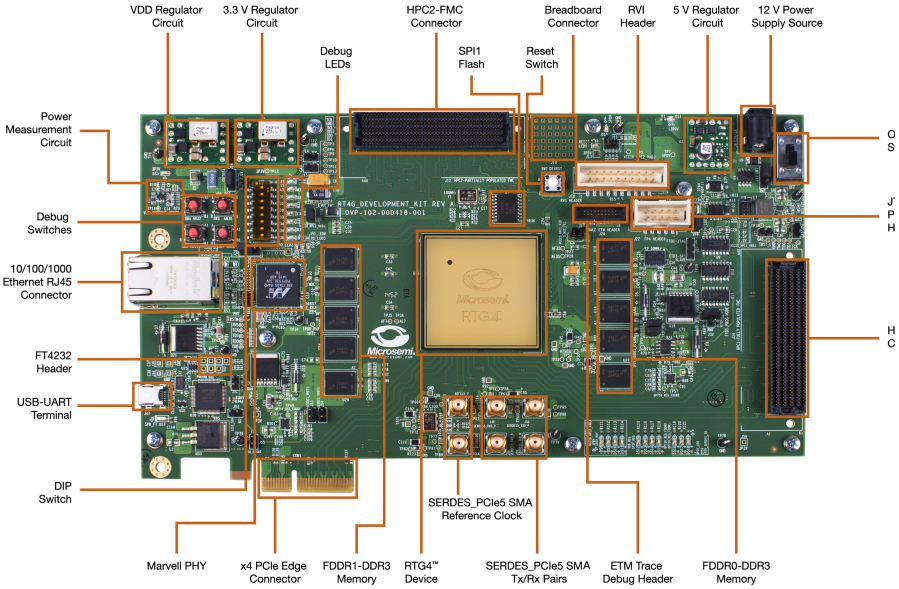


2.5.2 Running the Design

The following steps describe how to run the design in the RTG4 Development Kit. Design1 and design2 examples use the same steps.

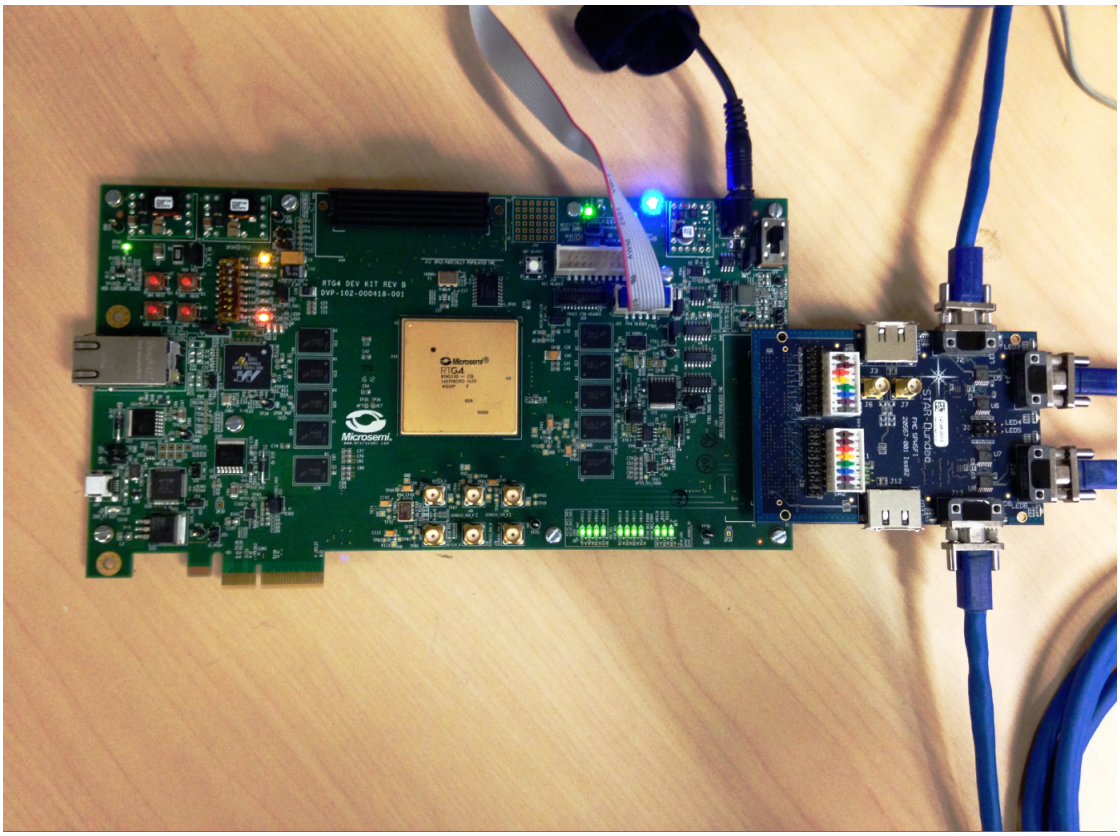
1. Connect the “STAR-Dundee SpaceWire/SpaceFibre FMC” daughter card with loop-back cable to FMC connector HPC1 on the RTG4 Development Kit board.
2. A cable should be connected between SPW1 and SPW2 ports on the daughter card.
3. Set DIP switch settings for “STAR-Dundee SpaceWire/SpaceFibre FMC” board. For Production/PROTO development kits, use: 00011011 00011011. For ES Development Kits, use: 0000 0000 1011 0000.

Figure 20 • **RTG4 Development Kit**



The following figure shows SpaceWire daughter card connected to the RTG4 Development Kit board.

Figure 21 • **RTG4 Development Kit Connected to SpaceWire Daughter Card**



4. Open the **RTG4_SpaceWire_90MHz** Libero project. See the Appendix 3: Design and Programming Files, page 26.
5. Connect the RTG4 Development Kit to the PC using FlashPro4, FlashPro5, or FlashPro6.
6. Connect the Power supply to the J9 connector.
7. Switch **ON** the power supply switch, **SW6**.
8. Select **Run PROGRAM Action** in the Libero Design Flow window to program the RTG4 device. Alternatively, the device can be programmed using FlashPro Express and the supplied programming .job file by following the steps in Appendix 1: Programming the Device Using FlashPro Express, page 22.
9. After successful programming, Switch **ON** the DIP1 and DIP8 switches and press DEVRST (SW7) or power cycle the board. A counter pattern on **LED1-4** is displayed.
10. **LED8** is lit when the data compare is passing.
11. Switch **ON** the DIP2 switch to introduce error in data pattern and press DEVRST (SW7) or power cycle the device. An LED8 flickers as data comparison switches between Pass and Fail. LED5, LED6, and LED7 will also flicker whenever the data comparison fails, indicating that the error counter is toggling.
12. Switch **OFF DIP2** switch.
13. Switch **ON DIP3** switch and press DEVRST (SW7) or power cycle the device to send same data and strobe signal.
14. A counter pattern on LED1-4 stops counting as there is no SpaceWire recovered clock.
15. Switch **OFF** the power supply switch, **SW6**.

2.6 Conclusion

SpaceWire designs can be implemented in RTG4 FPGAs. There are several challenges to overcome, when implementing SpaceWire designs on generic FPGAs. The complex problem is the recovery block implementation. The built-in SpaceWire RX clock recovery circuit in the RTG4 allows easy implementation of SpaceWire receiver interfaces.This application note provides additional guidance to successfully implement and analyze SpaceWire receiver clock and data recovery circuits in RTG4 designs.

3 Appendix 1: Programming the Device Using FlashPro Express

This section describes how to program the RTG4 device with the programming job file using FlashPro Express.

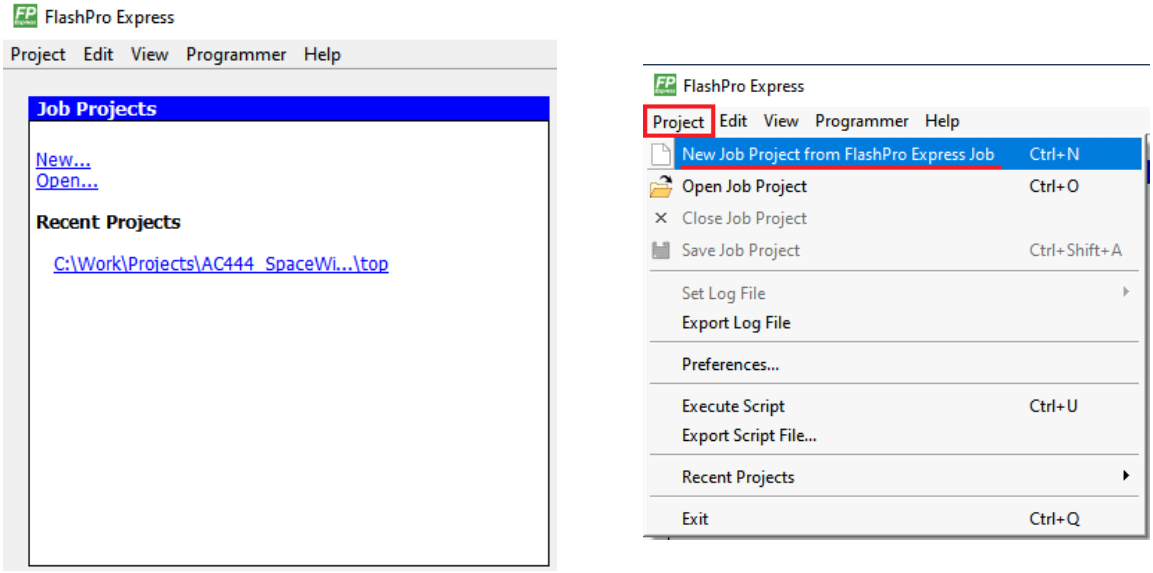
To program the device, perform the following steps:

1. Ensure that the jumper settings on the board are the same as those listed in *Table 3 of UG0617: RTG4 Development Kit User Guide*.
2. Optionally, jumper **J32** can be set to connect pins 2-3 when using an external FlashPro4, FlashPro5, or FlashPro6 programmer instead of the default jumper setting to use the embedded FlashPro5.

Note: The power supply switch, **SW6** must be switched **OFF** while making the jumper connections.

3. Connect the power supply cable to the **J9** connector on the board.
4. Power **ON** the power supply switch **SW6**.
5. If using the embedded FlashPro5, connect the USB cable to connector **J47** and the host PC. Alternatively, if using an external programmer, connect the ribbon cable to the JTAG header **J22** and connect the programmer to the host PC.
6. On the host PC, launch the **FlashPro Express** software.
7. Click **New** or select **New Job Project from FlashPro Express Job** from **Project** menu to create a new job project, as shown in the following figure.

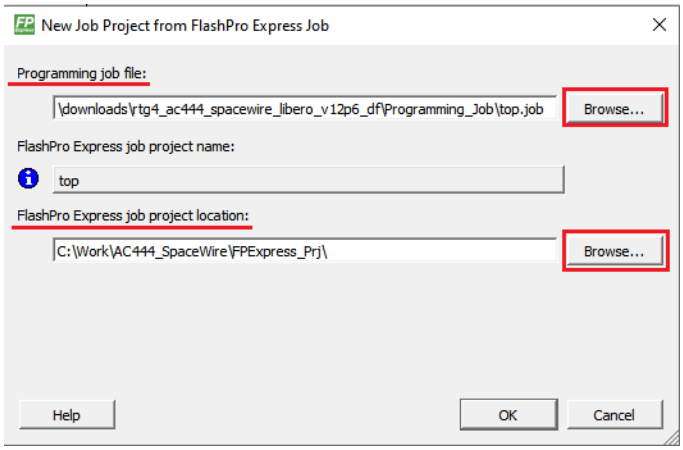
Figure 22 • **FlashPro Express Job Project**



8. Enter the following in the **New Job Project from FlashPro Express Job** dialog box:

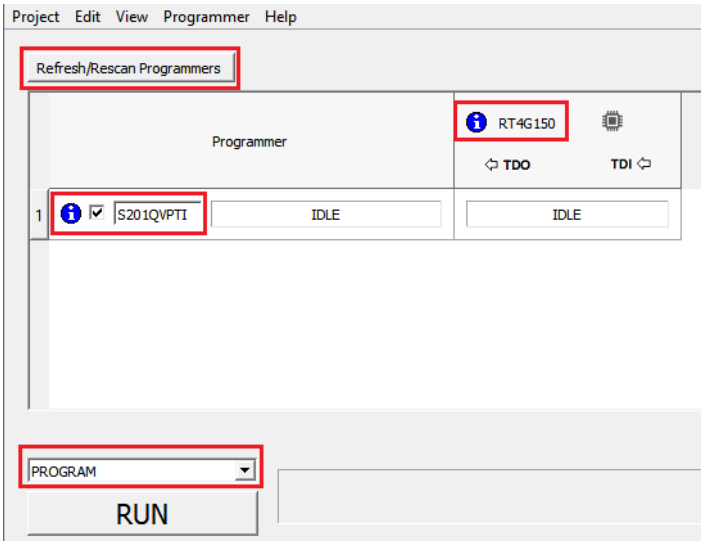
- **Programming job file:** Click **Browse**, and navigate to the location where the .job file is located and select the file. The default location is:
  `<download_folder>/rtg4_ac444_df/Programming_Job/top.job`
- **FlashPro Express job project location:** Click **Browse** and navigate to the desired FlashPro Express project location.

Figure 23 • **New Job Project from FlashPro Express Job**



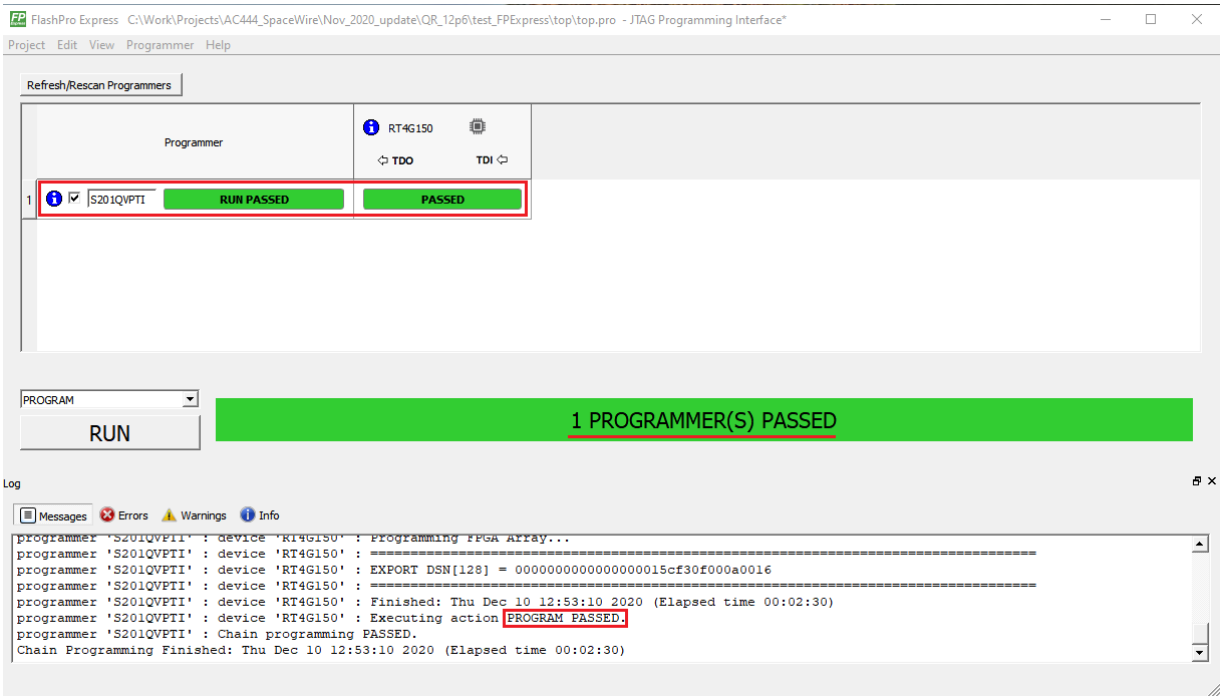
9. Click **OK**. The required programming file is selected and ready to be programmed in the device.
10. The FlashPro Express window appears as shown in the following figure. Confirm that a programmer number appears in the Programmer field. If it does not, confirm the board connections and click **Refresh/Rescan** Programmers.

Figure 24 • **Programming the Device**



11. Click **RUN**. When the device is programmed successfully, a **RUN PASSED** status is displayed as shown in the following figure.

Figure 25 • **FlashPro Express—RUN PASSED**



12. Close **FlashPro Express** or click **Exit** in the Project tab.

4 Appendix 2: Running the TCL Script

TCL scripts are provided in the design files folder under directory TCL_Scripts. If required, the design flow can be reproduced from Design Implementation till generation of job file.

To re-use the supplied script, update the `script.tcl` file with the desired folder path in the **Prj_Location** variable on line 10.

The `common.tcl` script specifies the versions of cores used in the IP Catalog and the Synthesis and Simulation tool profile names. To update the IP core versions, refer to Updating TCL for New Libero Versions and IP Versions, **page 25.**

To run the TCL, follow the steps below:

1. Launch the Libero software
2. Select **Project > Execute Script....**
3. Click Browse and select `script.tcl` from the downloaded AC444_Tcl_Scripts directory.
4. Click **Run**.

After successful execution of TCL script, the Libero project is created at the path specified by the **Prj_Location** variable in the `script.tcl` file.

For more information about TCL scripts, refer to **rtg4_ac444_df/readme.txt.**

Refer to *Libero® SoC TCL Command Reference Guide* for more details on TCL commands. Contact Microsemi Technical Support for any queries encountered when running the TCL script.

4.1 Updating TCL for New Libero Versions and IP Versions

Update the IP Core versions in the `common.tcl` and run the TCL flow in the new Libero version. By default, the given TCL script should run for new Libero and IP releases.

If there are any mismatches in execution, check the following:

1. For Libero related errors, check Libero release notes for changes related to the design components/tools or changes to TCL commands.
2. For IP core related errors, check the IP handbook and release notes for changes in IP configuration parameters, ports and their functionality. If necessary, update the IP configuration parameters, ports and connections in the TCL.

5 Appendix 3: Design and Programming Files

Download the RTG4 SpaceWire design files from the Microsemi Corporation website: *http://soc.microsemi.com/download/rsc/?f=rtg4_ac444_df.* The RTG4 design file consists of a Libero Verilog project and programming files for the RTG4 Development Kit. A TCL script is also included that can be used to rebuild the demo design files from the source files as described in Appendix 2: Running the TCL Script, page 25. See the `Readme.txt` file included in the design file folder for the directory structure and description.

6 Appendix 4: SpaceWire Pin list

The following table lists the types of SpaceWire pin numbers, names and the I/O types.

Table 5 • **Spacewire Pin List**

Pin Number	Pin Name	I/O Type
AD3	DDRIO146NB9/FDDR_W_ADDR12/SPWR_NW0_1_RX_DATA_N	DDRIO
AC3	DDRIO146PB9/FDDR_W_ADDR13/GB0_11/CCC_NW0_CLKI3/SPWR_NW0_1_RX_DATA_P	DDRIO
AC1	DDRIO147NB9/FDDR_W_ADDR14/SPWR_NW0_1_RX_STROBE_N	DDRIO
AB1	DDRIO147PB9/FDDR_W_ADDR15/GB0_11/CCC_NW0_CLKI2/SPWR_NW0_1_RX_STROBE_P	DDRIO
AC41	DDRIO94NB0/FDDR_E_ADDR14/SPWR_NE1_1_RX_STROBE_N	DDRIO
AB41	DDRIO94PB0/FDDR_E_ADDR15/GB12_23/CCC_NE1_CLKI2/SPWR_NE1_1_RX_STROBE_P	DDRIO
AD39	DDRIO95NB0/FDDR_E_ADDR12/SPWR_NE1_1_RX_DATA_N	DDRIO
AC39	DDRIO95PB0/FDDR_E_ADDR13/GB12_23/CCC_NE1_CLKI3/SPWR_NE1_1_RX_DATA_P	DDRIO
E7	MSIO253NB6/SPWR_SW0_1_RX_STROBE_N	MSIO
E6	MSIO253PB6/GB1/CCC_SW0_CLKI2/SPWR_SW0_1_RX_STROBE_P	MSIO
J9	MSIO254NB6/SPWR_SW0_1_RX_DATA_N	MSIO
H10	MSIO254PB6/GB3/CCC_SW0_CLKI3/SPWR_SW0_1_RX_DATA_P	MSIO
C11	MSIO265NB6/SPWR_SW1_1_RX_STROBE_N	MSIO
C10	MSIO265PB6/GB5/CCC_SW1_CLKI2/SPWR_SW1_1_RX_STROBE_P	MSIO
K14	MSIO266NB6/SPWR_SW1_1_RX_DATA_N	MSIO
K15	MSIO266PB6/GB7/CCC_SW1_CLKI3/SPWR_SW1_1_RX_DATA_P	MSIO
F29	MSIO334NB4/SPWR_SE0_1_RX_STROBE_N	MSIO
F28	MSIO334PB4/GB17/CCC_SE0_CLKI2/SPWR_SE0_1_RX_STROBE_P	MSIO
G27	MSIO335NB4/SPWR_SE0_1_RX_DATA_N	MSIO
F27	MSIO335PB4/GB19/CCC_SE0_CLKI3/SPWR_SE0_1_RX_DATA_P	MSIO
E33	MSIO346NB4/SPWR_SE1_1_RX_STROBE_N	MSIO
D33	MSIO346PB4/GB21/CCC_SE1_CLKI2/SPWR_SE1_1_RX_STROBE_P	MSIO
H29	MSIO347NB4/SPWR_SE1_1_RX_DATA_N	MSIO
G29	MSIO347PB4/GB23/CCC_SE1_CLKI3/SPWR_SE1_1_RX_DATA_P	MSIO
G40	MSIOD13NB2/SPWR_SE1_0_RX_STROBE_N	MSIOD
G39	MSIOD13PB2/GB12_23/CCC_SE1_CLKI0/SPWR_SE1_0_RX_STROBE_P	MSIOD
M32	MSIOD14NB2/SPWR_SE1_0_RX_DATA_N	MSIOD
N32	MSIOD14PB2/GB12_23/CCC_SE1_CLKI1/SPWR_SE1_0_RX_DATA_P	MSIOD
AB6	MSIOD167NB8/SPWR_NW1_1_RX_DATA_N	MSIOD
AB5	MSIOD167PB8/GB0_11/CCC_NW1_CLKI3/SPWR_NW1_1_RX_DATA_P	MSIOD
AB3	MSIOD168NB8/SPWR_NW1_1_RX_STROBE_N	MSIOD
AA3	MSIOD168PB8/GB0_11/CCC_NW1_CLKI2/SPWR_NW1_1_RX_STROBE_P	MSIOD
AB8	MSIOD182NB8/SPWR_NW0_0_RX_DATA_N	MSIOD

Table 5 • **Spacewire Pin List *(continued)***

Pin Number	Pin Name	I/O Type
AA8	MSIOD182PB8/GB0_11/CCC_NW0_CLKI1/SPWR_NW0_0_RX_DATA_P	MSIOD
U3	MSIOD183NB8/SPWR_NW0_0_RX_STROBE_N	MSIOD
T3	MSIOD183PB8/GB0_11/CCC_NW0_CLKI0/SPWR_NW0_0_RX_STROBE_P	MSIOD
V10	MSIOD197NB8/SPWR_NW1_0_RX_DATA_N	MSIOD
V11	MSIOD197PB8/GB0_11/CCC_NW1_CLKI1/SPWR_NW1_0_RX_DATA_P	MSIOD
M1	MSIOD198NB8/SPWR_NW1_0_RX_STROBE_N	MSIOD
M2	MSIOD198PB8/GB0_11/CCC_NW1_CLKI0/SPWR_NW1_0_RX_STROBE_P	MSIOD
P8	MSIOD218NB7/SPWR_SW1_0_RX_DATA_N	MSIOD
N8	MSIOD218PB7/GB0_11/CCC_SW1_CLKI1/SPWR_SW1_0_RX_DATA_P	MSIOD
K4	MSIOD219NB7/SPWR_SW1_0_RX_STROBE_N	MSIOD
K5	MSIOD219PB7/GB0_11/CCC_SW1_CLKI0/SPWR_SW1_0_RX_STROBE_P	MSIOD
N11	MSIOD227NB7/SPWR_SW0_0_RX_DATA_N	MSIOD
N10	MSIOD227PB7/GB0_11/CCC_SW0_CLKI1/SPWR_SW0_0_RX_DATA_P	MSIOD
G2	MSIOD228NB7/SPWR_SW0_0_RX_STROBE_N	MSIOD
G3	MSIOD228PB7/GB0_11/CCC_SW0_CLKI0/SPWR_SW0_0_RX_STROBE_P	MSIOD
K38	MSIOD22NB2/SPWR_SE0_0_RX_STROBE_N	MSIOD
K37	MSIOD22PB2/GB12_23/CCC_SE0_CLKI0/SPWR_SE0_0_RX_STROBE_P	MSIOD
P34	MSIOD23NB2/SPWR_SE0_0_RX_DATA_N	MSIOD
N34	MSIOD23PB2/GB12_23/CCC_SE0_CLKI1/SPWR_SE0_0_RX_DATA_P	MSIOD
M41	MSIOD43NB1/SPWR_NE0_0_RX_STROBE_N	MSIOD
M40	MSIOD43PB1/GB12_23/CCC_NE0_CLKI0/SPWR_NE0_0_RX_STROBE_P	MSIOD
V32	MSIOD44NB1/SPWR_NE0_0_RX_DATA_N	MSIOD
V31	MSIOD44PB1/GB12_23/CCC_NE0_CLKI1/SPWR_NE0_0_RX_DATA_P	MSIOD
U39	MSIOD58NB1/SPWR_NE1_0_RX_STROBE_N	MSIOD
T39	MSIOD58PB1/GB12_23/CCC_NE1_CLKI0/SPWR_NE1_0_RX_STROBE_P	MSIOD
AB34	MSIOD59NB1/SPWR_NE1_0_RX_DATA_N	MSIOD
AA34	MSIOD59PB1/GB12_23/CCC_NE1_CLKI1/SPWR_NE1_0_RX_DATA_P	MSIOD
AB39	MSIOD73NB1/SPWR_NE0_1_RX_STROBE_N	MSIOD
AA39	MSIOD73PB1/GB12_23/CCC_NE0_CLKI2/SPWR_NE0_1_RX_STROBE_P	MSIOD
AB36	MSIOD74NB1/SPWR_NE0_1_RX_DATA_N	MSIOD
AB37	MSIOD74PB1/GB12_23/CCC_NE0_CLKI3/SPWR_NE0_1_RX_DATA_P	MSIOD