4Links Opensource SpaceWire IP: RTG4 Porting Guide

Introduction

This document details the modifications required to build 4Links Opensource SpaceWire Router and standard SpaceWire codec on the RTG4 FPGA by Microsemi.

This document is packaged with the device specific files for the RTG4 FPGA.

This document should be used in tandem with existing instructions located within the relevant IP sub directory. If any conflicting information is present, this document should be taken as the correct source. Note this applies to the RTG4 implementation ONLY.

The RTG4 specific design files will be provided in addition to the 4Links SpaceWire IP repository. 4Links cannot guarantee that future updates to the SpaceWire IP stack do not conflict with the source files provided as part of this porting package.

Should conflicts occur, please email support@4links.space for the latest up-to-date advice.

SpaceWire CoDec

As the CoDec uses device-specific primitives, this IP has undergone the most modification.

The top-level wrapper file, ***spw_wrap_top_level_RTG4.vhd,*** contains new device primitive instantiations for single-ended and differential IO modes on the RTG4.

The clocking-structure of the SpaceWire Codec has also been adjusted. This was done to best-match the fabric capabilities of the RTG4 FPGA. Troublesome paths within the Rx-logic of the core have been optimized to better reflect the capabilities of the RTG4 FPGA fabric.

Note that this is primarily done through the elimination of “clock_b”. Instead of two clock sources at 180-degree phase, the core now requires only a single clock-source. All interface signals are timed with respect to the positive edge of the core clock.

SpaceWire Router

This is a modification of the ***router_top_level.vhd*** file to instantiate the new ***spw_wrap_top_level_RTG4.vhd*** entity as part of the router architecture. Additionally, the separate SpaceWire Port Clock and Router Clock domains have been removed. Now a single clock-domain, Router Clock, is used for the whole design. This removes CDC paths from the core and should make core clocks easier to constrain. This modification is done with the intention that the SpaceWire core frequency does not exceed 200MHz within the RTG4.

The RTG4 has a limited Fmax of ~300MHz on the FPGA (LUT) Fabric. This can make high speed designs challenging. For support on reaching >250Mb SpaceWire on the RTG4, contact James@4Links.space for additional support.

Adding IP to your Project

To add IP to your project, first follow the relevant IP user guides located in the opensource IP directories. Make sure that all packages and context clauses are added to your project before importing any IP design files. More information on adding thew required package files and libraries can be found in the ***read_me.txt*** found in ***.../opensource/src/4links/common_packages***.

Additional information can be found in relevant IP user guides for both the SpaceWire Router and Codec.

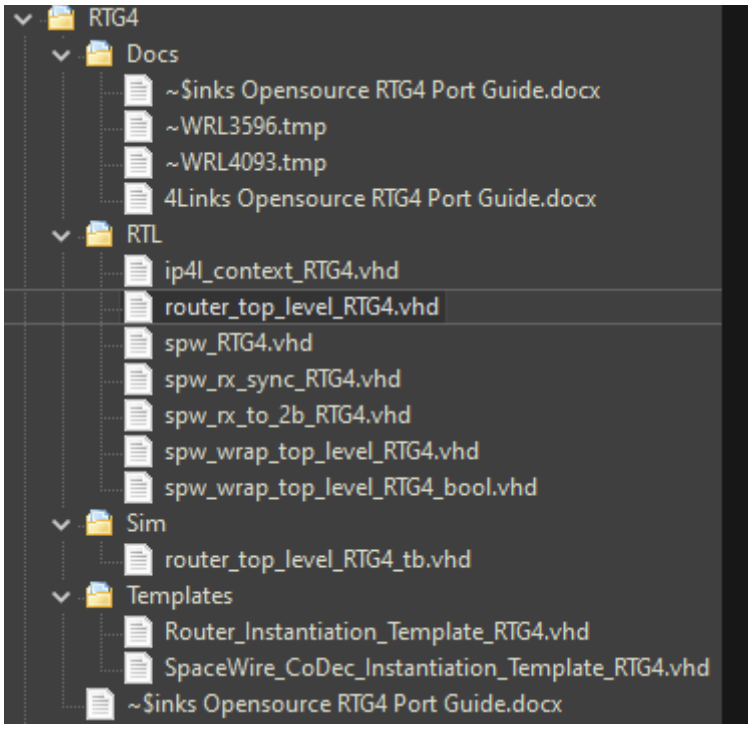

Figure 1: RTG4 Directory Files

Figure 1 shows the files provided as part of the RTG4 Port.

The files located in the ***RTG4/RTL*** directory should be added to your project after adding the required opensource IP files.

These files contain the modified design entities for porting the SpaceWire Router and CoDec to the RTG4. When building a project for the RTG4, these entities should be instantiated into your project as required.

The Templates folder in Figure 1 contains VHDL Instantiation Templates for both the router and codec RTG4 Port. Note that these files are non-synthesizable and should not be added to your Libero project.

IMPORTANT*:* ***ip4l_context_RTG4.vhd* must **REPLACE** ***ip4l_context.vhd*** in your project. ***DO NOT HAVE BOTH IN YOUR PROJECT AT THE SAME TIME!***

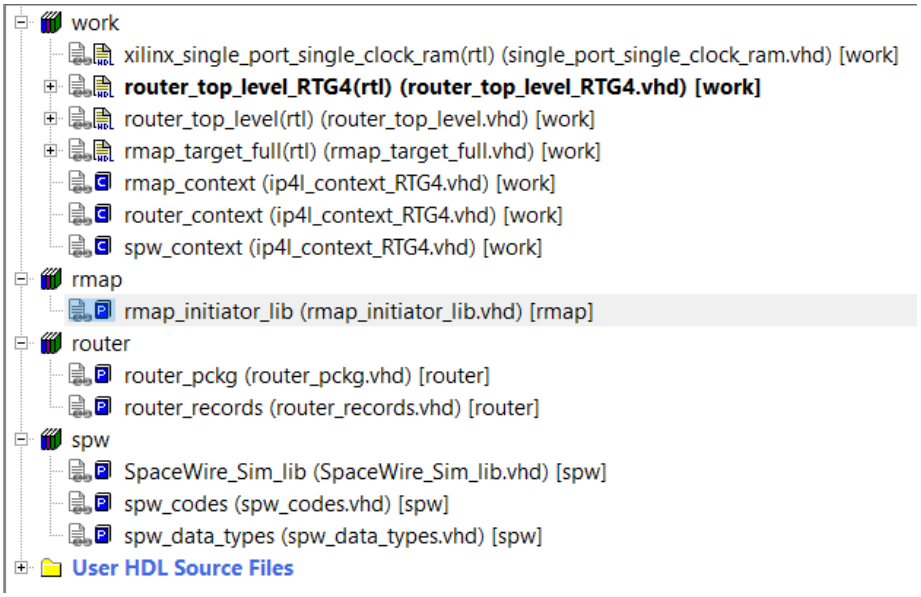

Figure 2: Libero Design Hierarchy

Figure 2 shows the Libero design hierarchy after all the relevant project files have been added to the design. Note that ***router_top_level*_RTG4** has been set as the root design entity.

The ***rmap***, ***router*** and ***spw*** libraries were created with the appropriate packages added as required. Unlike Vivado, Libero automatically recognizes VHDL-2008 files and uses the appropriate language features such as context clauses.

Where direct entity instantiation is used, entities are set as part of the default VHDL working library (work). Should you move the SpaceWire IP files into a separate library for project organization, then you must modify the entity instantiations to reference the new library name.

Port Limitations

Simulation via Modelsim packaged with Libero, using the provided testbench files does not work. This is due to Modelsim's poor support for protected-type based stimuli in VHDL. We have developed a "class-based" simulation set for generating SpaceWire/RMAP data. This is loosely based on practices used by OSVVM. Unfortunately, Modelsim support for these methods is not adequate.

Even after extensive modifications to remove any language conflicts, the packaged version of Modelsim ran out of memory when attempting to start the simulation. At this time, we cannot guarantee the working of testbenches outside of Xilinx Vivado XSim.

The RMAP_periph_packg will also not work in Modelsim. This is down to modelsim having poor support for interaction between integer and real values. XSim does not have this problem. As a result, the rmap_perhipheral package has been omitted from the project source list.