XAPP525 (v2.0) October 15, 2004

SPI-4.2 to Quad SPI-3 Bridge

Summary

This application note describes a reference design used to bridge one 4-channel Xilinx SPI-4.2 (PL4) core (v6.1) to four 1-channel SPI-3 (PL3) Link Layer cores (v3.2). The design is implemented in a Virtex-II™ device.

Introduction

System Packet Interface Level 3 (SPI-3) provides a link-layer interface for transferring packets at the OC48 data rate. System Packet Interface Level 4 Phase 2 (SPI-4.2) provides a link-layer interface for transferring packets at the OC192 data rate. In many applications, it is desirable to move packets from one data rate to another, or bridge between two standard devices supporting different interfaces (SPI-3 and SPI-4.2). This application note presents a design bridging one 4-channel Xilinx SPI-4.2 (PL4) core (v6.1) to four 1-channel POS-PHY Level-3 (SPI-3) Link Layer cores (v3.2).

Design Facts

This section provides key information on the SPI-4.2 to Quad SPI-3 Bridge reference design. Table 1 shows device, performance, and resource utilization information, Table 2 shows information on implementation requirements for the reference design.

Table 1: **Reference Design Specifics**

Name	Value
Device Supported	XC2V3000-5-FF1152
Performance	up to 350 MHz on SPI-4.2 interface; up to 104 MHz on SPI-3 interface; up to 175 MHz on the bridge design
Resource Utilization (includes SPI-4.2 and SPI-3 cores)	7232 Virtex-II slices; 51 block RAM

Table 2: **Design Implementation Requirements**

Name	Tool Used
Synthesis Tool	Synplify 7.2.2
Implementation Tool	Xilinx ISE v6.2i SP3
IP Cores	SPI-4.2 (PL4) v6.1; SPI-3 (PL3) v3.2; SPI-4.2 to Quad SPI-3 Bridge v2.0 (Reference Design)

Design Overview

The XAPP525.zip reference design provides a bridge between a single 4-channel SPI-4.2 core and four 1-channel SPI-3 cores. All necessary functionality to transfer packets between the two sets of interfaces and appropriate flow control information is provided to the SPI-4.2 core and the SPI-3 cores. The reference design works with the Xilinx SPI-4.2 (PL4) core v6.1 and the POS-PHY Level-3 (SPI-3) Link Layer cores v3.2.

The design is divided into two distinct sections. The first section handles passing packets from the single SPI-4.2 sink core to the four SPI-3 transmit cores. The second section handles passing packets in the opposite direction, from the four SPI-3 receive cores to the single SPI-4.2 Source core. Figure 1 depicts a high-level block diagram of the design. Only the portion included in the box labeled *Bridge Reference Design* is included as part of the reference code. The other portions of the design must be purchased from Xilinx.

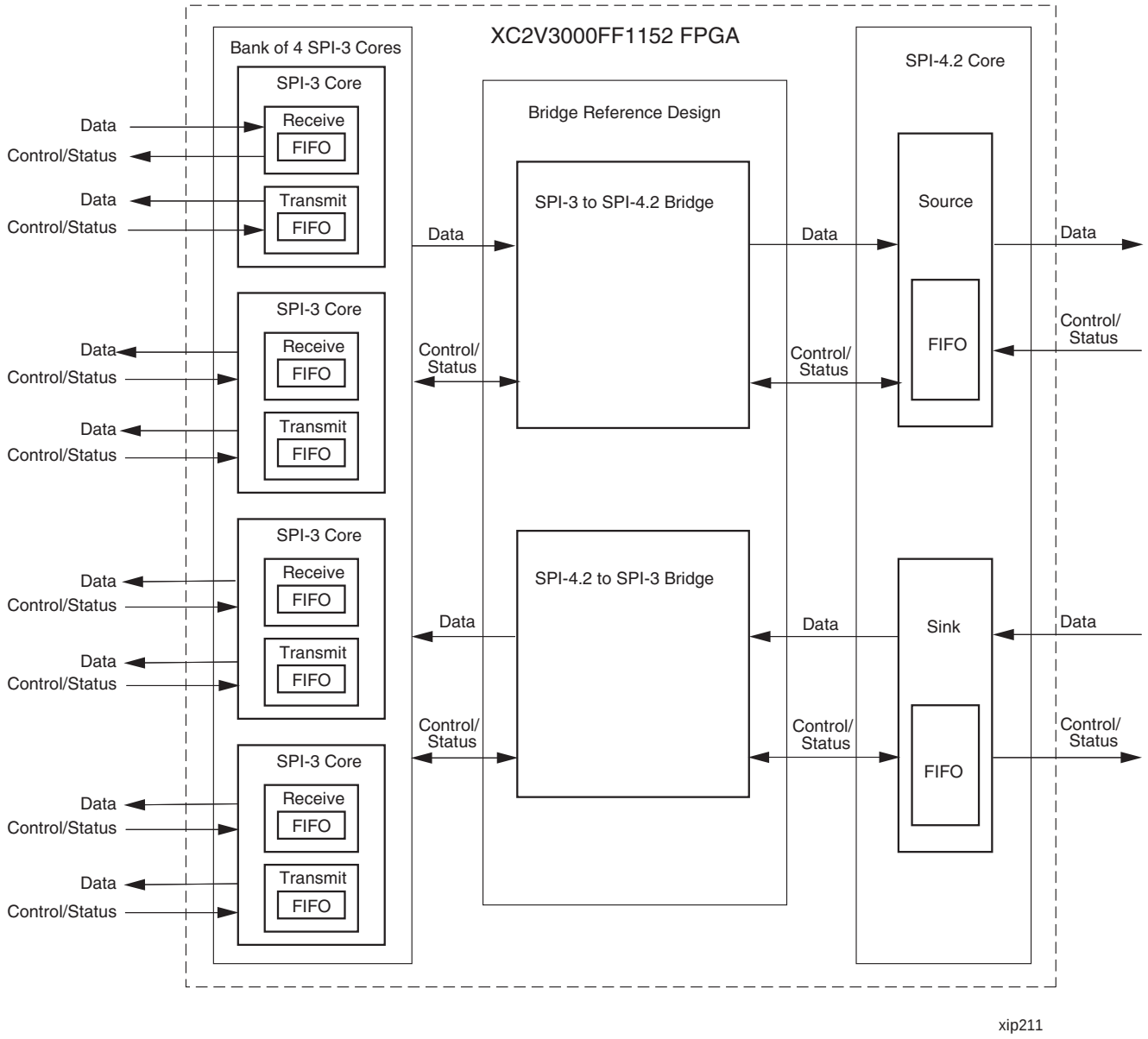


Figure 1: **SPI-4.2 to SPI-3 Bridge Top-level Block Diagram**

SPI-4.2 Core to four SPI-3 Cores Path

Figure 2 is a block diagram of the portion of the design that handles passing packets from the single SPI-4.2 sink core to four SPI-3 transmit cores.

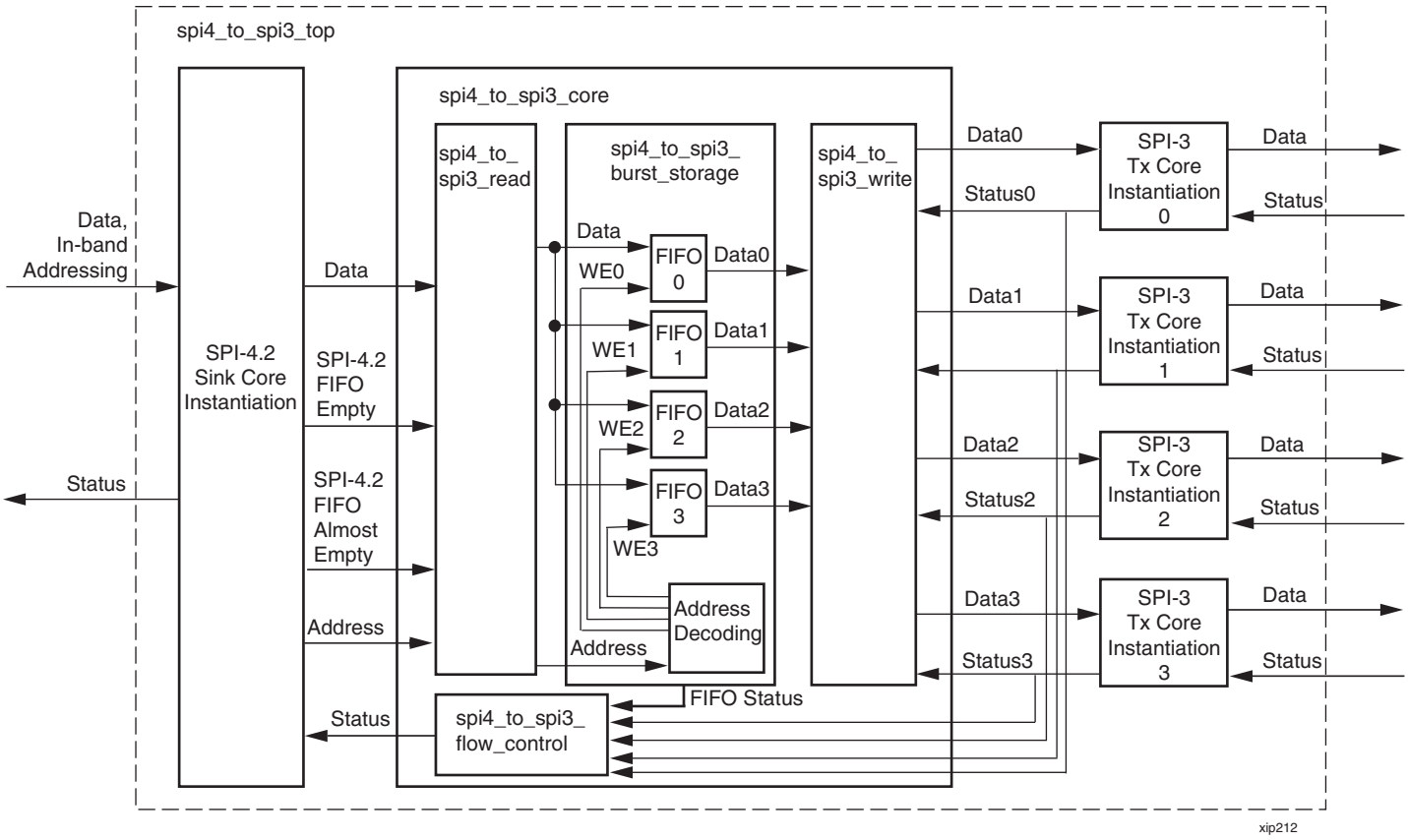


Figure 2: **SPI-4.2 Sink Core to SPI-3 Transmit Core Path Block Diagram**

At the top level, the SPI-4.2 sink core, four SPI-3 transmit cores, and spi4_to_spi3_core are instantiated. SPI-4.2 to SPI-3 core contains the bridging logic. Within the SPI-4.2 to SPI-3 core, there are four sub-modules: the read module, the burst storage module, the write module, and the flow control module.

The read module controls reading packet data and address information from the SPI-4.2 sink core. When data is read from the SPI-4.2 sink core, it is 64-bits wide. Along with data, SOP, EOP, Err, and Mod signals are also pulled from the SPI-4.2 sink core. The address information is 2-bits wide. If the user side of the SPI-4.2 sink core indicates that the core's internal FIFO is empty, then no more words are read from the core. Otherwise, data is read from the core. Therefore, if the device supplying data to the SPI-4.2 core in the user's system is slow to respond to a request to halt data transfer (a status of *satisfied*), then overflow is possible within the bridge. The user of the design must configure system parameters to prevent overflow of the bridge.

The burst-storage module contains four FIFOs for holding packet data and address decoding logic. Each FIFO corresponds to the data for one SPI-3 transmit core. The FIFOs instantiated within these modules are Xilinx LogiCORE FIFOs, provided free with the Xilinx software v6.2i. When data enters the FIFOs within the burst-storage module, it is written as 70-bit data (64 bits of packet data, SOP, EOP, 3-bits Mod signal, and the Err signal). It also comes out as 70-bit data. Each FIFO will hold 4 KBytes of data, corresponding to 512 72-bit words (two of the bits are not used). The address is decoded: address 0 for the SPI-4.2 sink core corresponds to instantiation 0 of the SPI-3 transmit core, address 1 corresponds to instantiation 1, and so on.

The address decoding sub-module provides individual write enable signals to each FIFO sub-module.

The write module controls reading data out of each of the FIFOs contained in the burst storage module, and writing it into the SPI-3 transmit cores. Data is read out of each of the FIFOs of the burst-storage module if: the FIFO is not empty; the corresponding SPI-3 transmit core's FIFO is not almost full; and the status from the SPI-3 bus is not requesting a stop of the flow of source data. After the packet data is read from the burst storage FIFO, the data must be converted from 64 bits to 32 bits. In addition, the Mod signal must be examined to determine if all of the 64-bit data is valid, and to determine the correct time to send EOP, SOP, Mod, and Err signals. The Mod signal must also be converted into a 2-bit signal.

The flow control module controls the status that is sent back to the SPI-4.2 sink core for transmission on the SPI-4.2 bus. The status for a particular SPI-4.2 channel is *starving* unless the corresponding SPI-3 transmit core's FIFO is almost full or the corresponding burst storage FIFO is half full, in which case the status is *satisfied*. The half full level was chosen for the burst storage FIFO module so that more latency can be tolerated before the device supplying data to the SPI-4.2 sink core must respond to the *satisfied* status.

The hierarchy of the SPI-4.2 core to four SPI-3 cores path portion of the design is illustrated in Figure 3.

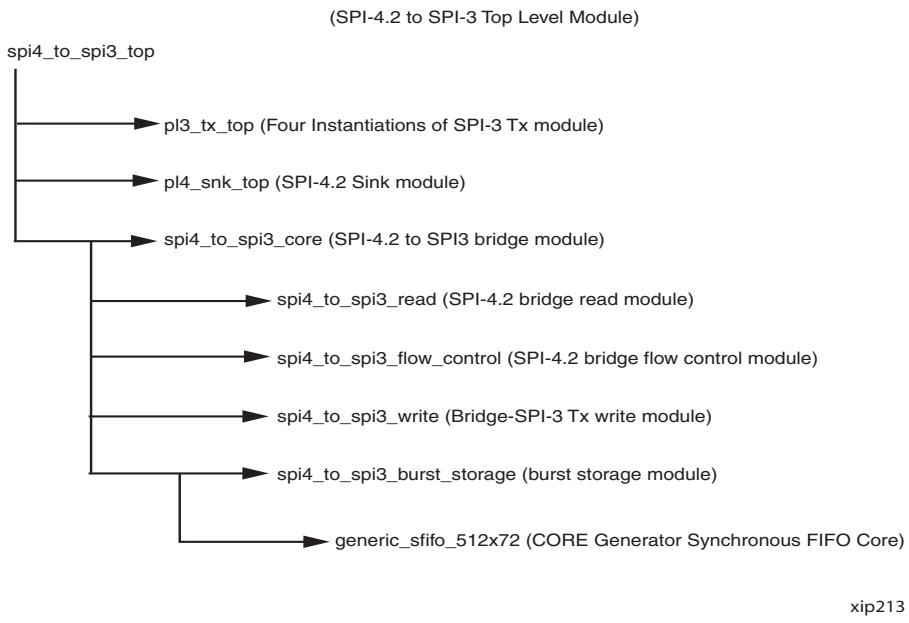


Figure 3: **SPI-4.2-to-SPI-3 Bridge Hierarchy Structure**

Four SPI-3 Cores to One SPI-4.2 Core Path

The block diagram for four SPI-3 cores to one SPI-4.2 core path is shown in Figure 4. This block handles SPI-3 traffic from four single-channel SPI-3 cores and transfers them to a single four-channel SPI-4.2 Source core. Flow control received from the SPI-4.2 core determines how much data is transferred for each channel. This block also provides 4KB intermediate buffering for each channel in the bridge.

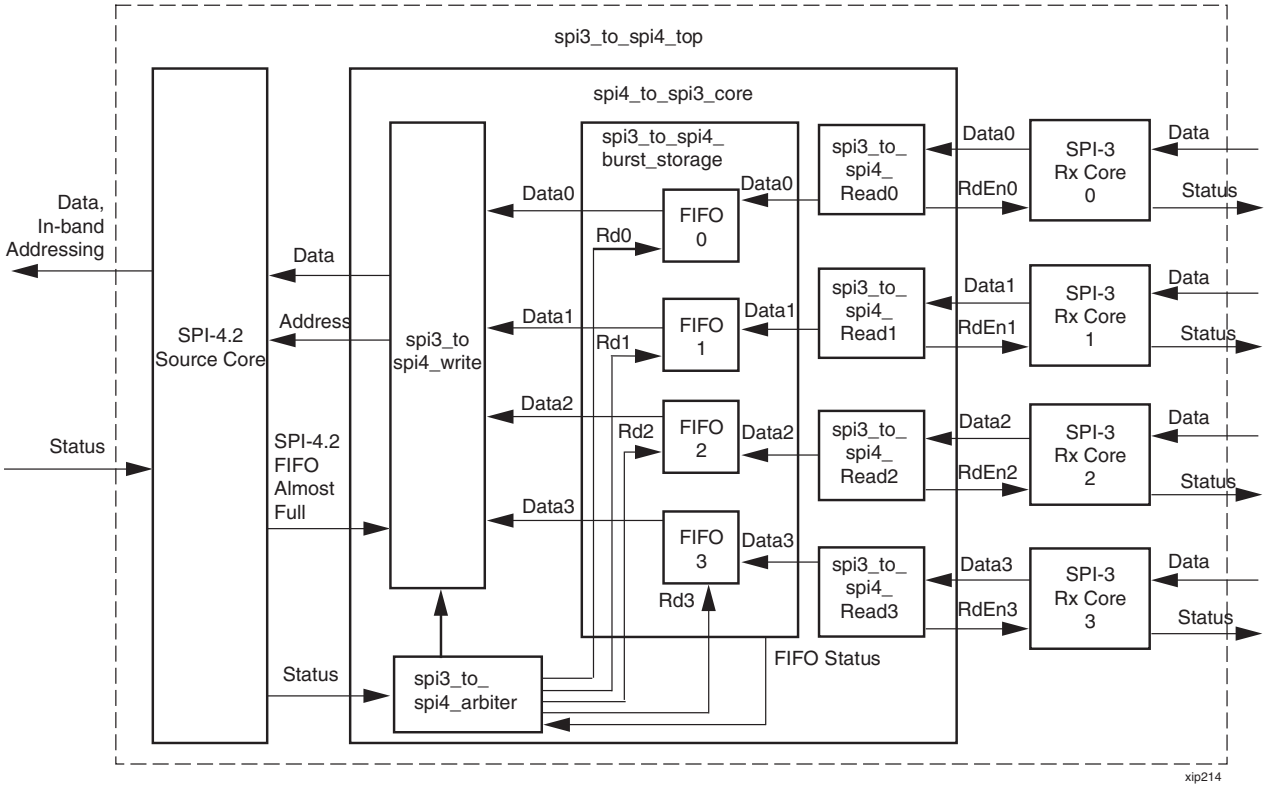


Figure 4: **SPI-3 to SPI-4.2 Bridge Block Diagram**

The top-level block is spi3_to_spi4_top. It has instantiations of spi3_to_spi4_core, four SPI-3 Rx cores and one SPI4.2 Source core. All the bridge functionality is performed by the spi3_to_spi4_core block with four main sub blocks: spi3_to_spi4_read, spi3_to_spi4_burst_storage, spi3_to_spi4_arbiter, and spi3_to_spi4_write.

The read module spi3_to_spi4_read is instantiated four times, one for each SPI-3 Rx core. These read modules read from a SPI-3 Rx core and write into burst storage FIFOs. If the SPI-3 Rx FIFO is not empty, the read module reads from the SPI-3 FIFO and transfers to burst storage FIFO as long as the burst storage FIFO does not fill up. This read module stops reading from SPI3 Rx FIFO as soon as the burst storage FIFO is almost full.

The burst storage modules provide intermediate buffering for each SPI-3 core's packet data. There are four instantiations of this module, one for each SPI-3 Rx core. The FIFOs instantiated within these modules are Xilinx LogiCORE FIFOs, provided free with the Xilinx software ISE v6.2i. These burst storage modules have 4KB capacity and store all the packet info including SOP, EOP, Err, Mod and packet data from the SPI-3 Rx core. The burst storage module merges two 32-bit SPI-3 data words into one 64-bit FIFO word and converts the 2-bit SPI-3 Mod signals to 3-bit SPI-4.2 Mod signals. The burst storage module also provides FIFO status information indicating whether the FIFO is full or almost empty. The burst storage FIFOs' Almost Empty threshold is defined in the package file `spi_pkg.v/vhd` and this value has to be greater than the MaxBurst1 and MaxBurst2 parameters. The MaxBurst1 and MaxBurst2 parameters are defined in the **Configuration Parameters** section later in this document. Finally, the burst storage module keeps track of the number of EOPs in the FIFO and is used during arbitration.

The arbiter module `spi3_to_spi4_arbiter.v/vhd` arbitrates between four SPI3 channels to transfer packet data from burst storage to the SPI-4 Source FIFO. Each SPI-3 channel is eligible for transfer when it has at least one EOP in its burst FIFO or if its burst storage FIFO has more data than *Almost Empty Threshold* words. The arbiter always transfers a multiple of SPI-4.2 credits (1 credit = 16 bytes) as per the SPI-4.2 specification. The arbiter monitors SPI-4.2 Source status; the data transferred for each channel depends on the status of each channel.

The arbiter:

- Transfers *MaxBurst1* credits when Starving (*00*) status is received for a channel;
- Transfers *MaxBurst2* credits when Hungry (*01*) status is received for a channel; and
- Terminates transfer immediately after EOP is transferred from burst storage to SPI-4.2 Source FIFO.

The write module *spi3_to_spi4_write* writes the current channel's data from its burst storage FIFO to the SPI-4.2 Source FIFO. This writes SOP, EOP, Err, Mod, Addr and data into SPI-4.2 Source FIFO.

The hierarchy of the four SPI-3 cores to one SPI-4.2 core path portion of the design is illustrated in Figure 5.

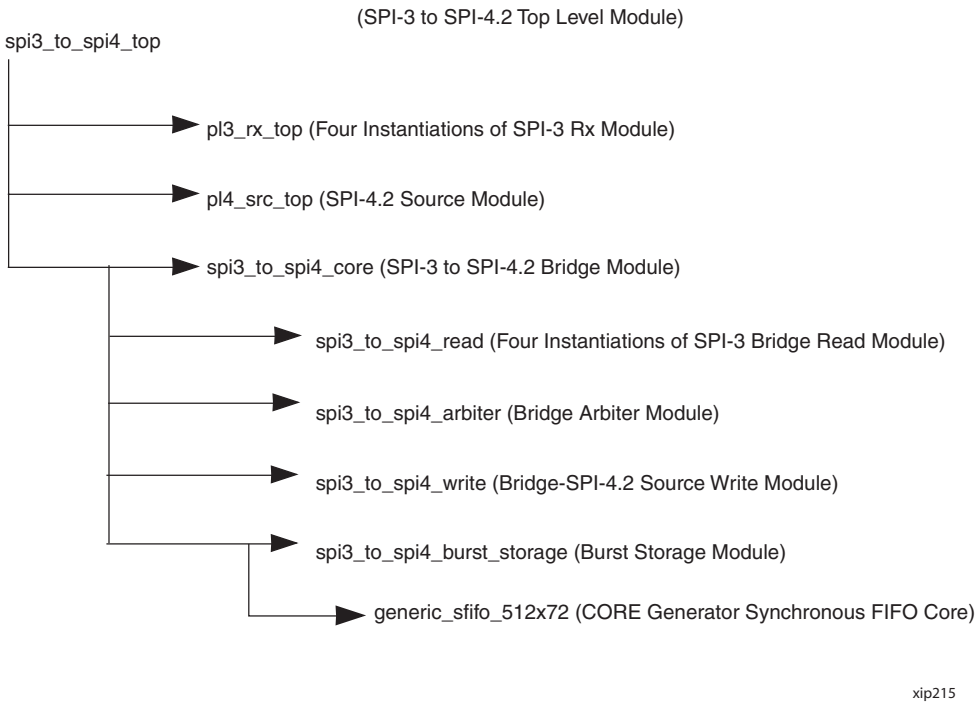


Figure 5: **SPI-3 to SPI-4.2 Bridge Hierarchy Structure**

Initialization Sequencing

Also included at the top level of the reference design is a module called spi_clk_startup. This module assures that the SPI-4.2 core Source and Sink modules and the DCMs within these modules are initialized in the proper sequence (as explained in solution record 16176).

Configuration Parameters

Table 3 shows the bridge configuration parameters defined in the `spi_pkg.v/vhd` file.

Table 3: **SPI-4.2 to SPI-3 Bridge Configuration Parameters**

Configuration Signal	Description and Range	Range	Default Value
MaxBurst1	**MaxBurst1 for Starving Channels:** When a SPI4.2 channel receives Starving status, (status = *00*), it is able to accept a burst length of MaxBurst1 blocks. SPI3-to-SPI4 bridge will transfer MaxBurst1 SPI4.2 credits (1 credit = 16 bytes) to the Starving channel. Units are 16-byte credits.	1 to 255 MaxBurst1 < (SPI3_to_SPI4_AEThres parameter)/ 2	8
MaxBurst2	**MaxBurst2 for Hungry Channels:** When a SPI4.2 channel receives Hungry status, (status = *01*) it is able to accept a burst length of MaxBurst2 blocks. SPI3-to-SPI4 bridge will transfer MaxBurst2 SPI4.2 credits (1 credit = 16 bytes) to the Hungry channel. Units are 16-byte credits.	1 to 254 MaxBurst1 < (SPI3_to_SPI4_AEThres parameter)/2	4
SPI3-to-SPI4_AEThresh	**SPI3-to-SPI4 Burst Storage Almost Empty Threshold:** This parameter specifies the number of FIFO words (64 bit) that must be present in the Burst storage FIFO for a given channel before FIFO almost empty signal is asserted. This value must be greater than 2*MaxBurst1.	4 to 504	255

Synthesis and Implementation

This reference design has all the required VHDL and Verilog files for the bridge logic between four SPI-3 cores and one SPI-4.2 core. Before implementing the reference design, the user must purchase the SPI-4.2 and SPI-3 cores from Xilinx. The netlist files and simulation models of the SPI-4.2 and SPI-3 cores are required for implementation and verification of this bridge.

The reference design directory structure is made up of three sub-directories: the `hdl/` directory, the `implement/` directory, and the `test/` directory. The `hdl/` directory provides all Verilog | VHDL files for the bridge design (excluding the SPI-3 and SPI-4.2 core). The `implement/` directory allows users to implement the design.

Note: The synthesis tool used for the reference design was Synplify Pro 7.2.2. The version of Xilinx implementation tools used was v6.2i Service Pack 3.

To synthesize the bridge design, go to the `implement/synthesis` directory and run the following command:

from a UNIX prompt: `source run_synthesis`

from a DOS prompt: `run_synthesis.bat`

This command creates the following files and copies them to the `implement/netlists` directory.

```
spi4_to_spi3_top.edf
spi3_to_spi4_top.edf
bridge_top.edf
```

Before implementing this design, the following files need to be placed in the `implement/netlists` directory. Purchase the SPI-4.2 and SPI-3 cores from Xilinx.

```
pl3_rx_top.edf
pl3_tx_top.edf
pl4_snk_top.edf
pl4_snk_top.ncf
pl4_src_top.edf
pl4_src_top.ncf
```

To implement the design, go to the `implement` directory and run the following command.

from a UNIX prompt: `source build_bridge_top`

from a DOS prompt: `build_bridge_top.bat`

This script deletes all contents of the `implement/fpga` directory. Backup any data from this directory before executing this command.

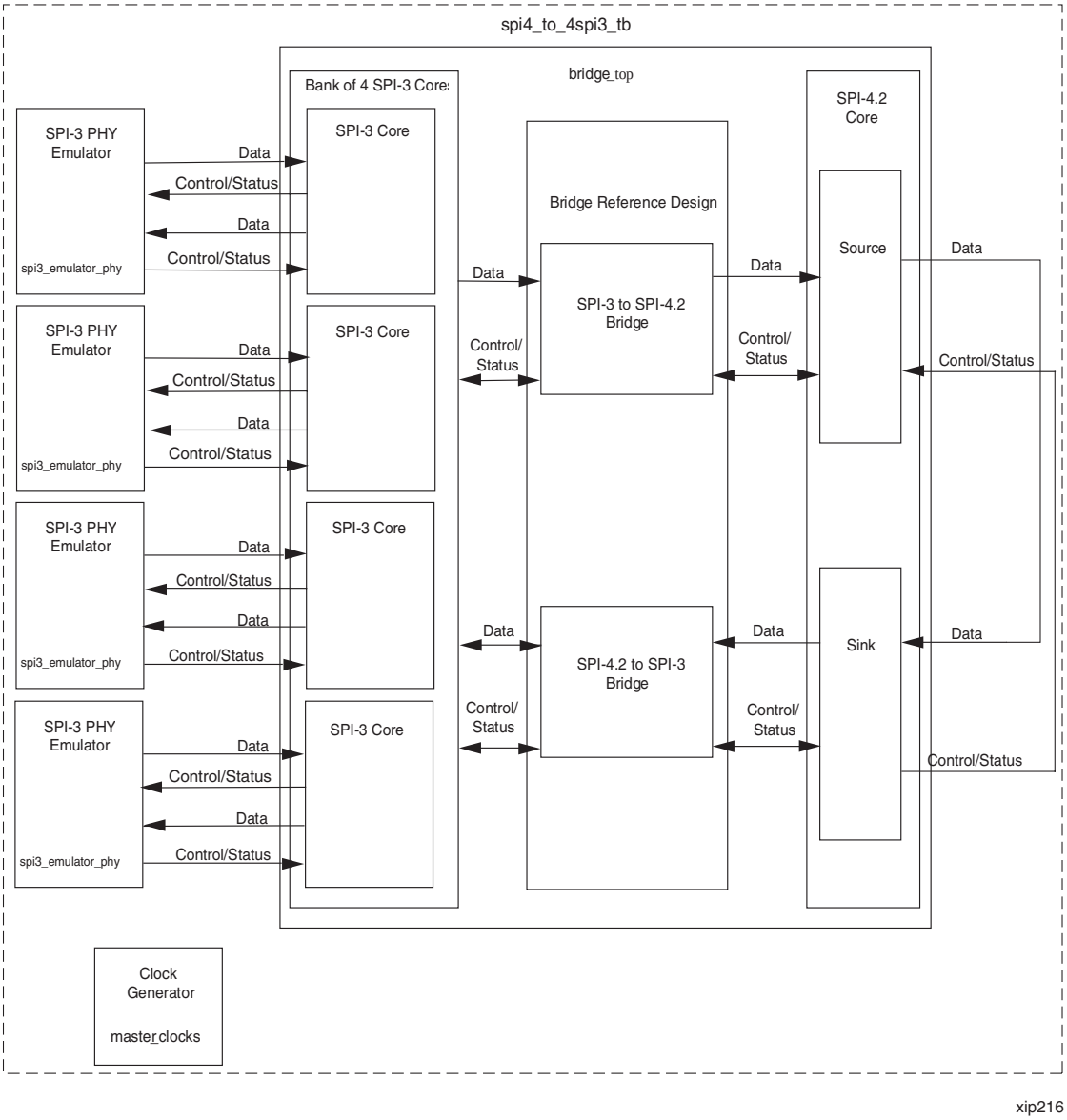
Example report files are provided in the `implement/example_reports` directory.

A UCF (User Constraints File) file is provided in the `implement/constraints` directory. All constraints that are required for the implementation of the SPI-3 and SPI-4.2 cores are included in the provided UCF file. If the user has a different version of the cores than the one used when implementing this design (see the application note for core version numbers), then these constraints must be replaced.

Finally, a wrapper file for the entire design is included in the directory `implement/<verilog|vhdl>`.

Testbench

Figure 6 shows the demonstration test suite provided with this reference design.



xip216

Figure 6: **Demonstration Test Suite**

The demonstration test suite consists of three files:

`master_clocks.v/vhd,`

`spi3_emulator_phy.v/vhd`

`spi4_to_4spi3_tb.v/vhd`

The hierarchical structure of files used in the test bench is shown in Figure 7.

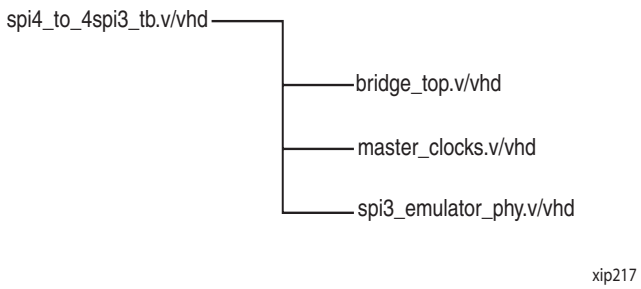


Figure 7: **Testbench File Structure**

The top-level file is `spi4_to_4spi3_tb.v/vhd`. In this file, the two other components of the test suite and the bi-directional bridge design under test (`bridge_top.v/vhd`) are connected together. The test suite instantiates four versions of the SPI-3 PHY Emulators (`spi3_emulator_phy.v/vhd`) and connects them to the SPI-3 Rx and Tx interfaces on the bridge. The SPI-4.2 interfaces on the bridge are connected together in a looped back manner to allow data to flow through both directions of the bridge. In its supplied form the test executed is very simple and only designed to show data flow through the bridge. Initially, the emulators and the bridge design are reset. They are brought out of reset and the data is allowed to flow from the SPI-3 emulators through the SPI-3 to SPI-4 bridge. As the SPI-4.2 interface is looped back, the data that was passed through the SPI-3 to SPI-4 bridge passes back through the SPI-4 to SPI-3 bridge into the SPI-3 emulators.

The file `master_clocks.v/vhd` generates the external clocks necessary to drive the bridge design and the emulators. Three clocks are required, one for the SPI-3 Rx interfaces, one for the SPI-3 Tx interfaces and one for the SPI-4.2 interface.

The `spi3_emulator_phy.v/vhd` file provides a simplified behavioral simulation model of a SPI-3 PHY. The model is configured as a single-channel PHY device that transmits sets of eight DWORD (nominally 32 byte) packets into an SPI-3 port. The Start of Packet and End of Packet signals are driven as expected to send an eight_DWORD packet. The values the Modulus (Mod) and Err signals are driven with during an End-of-Packet condition are changed throughout the simulation so that the packet lengths vary between 29 and 32 bytes and some packets get sent with the Err signal asserted to indicate a packet error. The emulator uses a generic value to seed the starting data value for the packets to be sent to the link layer SPI-3 port, thus allowing different instances of the emulator to send packets containing different data. The data sent in each packet is the same. For the Tx part of the PHY interface, polled status information is provided back to the Tx interface on the link layer SPI-3 port that varies with time. As the emulator is only designed to be a simple demonstration, no error checking is performed on the data it receives on its SPI-3 Tx interface.

Some additional files need to be copied into the simulation directory to use the test suite. These files are the gate-level simulation models for the SPI-3 and SPI-4.2 cores used in the bridge design. These models are provided with the cores from Xilinx, separate from the reference design. These simulation models use alternative names for SPI-3 and SPI-4: POS-PHY Level 3 (PL3) and POS-PHY Level 4 (PL4). Once the files `pl3_rx_top.v/vhd`, `pl3_tx_top.v/vhd`, `pl4_snk_top.v/vhd`, and `pl4_src_top.v/vhd` are copied to the simulation directory, run the test suite.

Running a Simulation

To run the example, start Modelsim from the `test/<verilog|vhdl>`. In the Modelsim main window execute `do simulate.do`. This compiles the appropriate source files, runs the simulation, and displays the waveforms at the end of the simulation run.

Data is visible at all 4 SPI-3 interfaces and the looped back SPI-4 interface, as seen in the waveform when running the sample MTI scripts provided with the demonstration test suite. Only the external interfaces are shown to reduce the complexity of the displayed waveforms.

Modifying the Reference Design

Minor modifications can be made to the SPI-4.2 to SPI-3 bridge reference design RTL. This section describes how to make a few of these modifications. Once the modifications are made, the design may not meet timing using the constraints file provided.

Performance Requirements

This design is specified to work with the SPI-4.2 core at an operating frequency of 350 MHz DDR (175 MHz internal frequency), and the SPI-3 core at an operating frequency of 104 MHz. The constraints file can be modified to support additional data rates, up to the maximum operating frequencies of the SPI-4.2 and SPI-3 cores.

Clocking Requirements

The SPI-4.2 user interface, bridge logic, and SPI-3 user interfaces are currently all clocked synchronously from the SPI-4.2 output clock SysClkDiv_GP. This design can be modified to clock these interfaces with a different clock. To clock the SPI-4.2 and SPI-3 user interfaces at different clock frequencies, refer to the '**Changing to an Asynchronous FIFO**' section later in this document.

To simplify clocking within the design, the bridge design assumes that all four SPI-3 cores use the same RFClk and TFClk clocks. The bridge design can be modified to support independent clocks for the SPI-3 interfaces.

Device and Package Requirements

This reference design has been implemented in a Virtex-II XC2V3000FF1152-5. However, the provided constraints file can be modified to support additional device/package combinations. Any new device/package combination for this design must be supported by the SPI-4.2 v6.1 core. See the SPI-4.2 v6.1 core data sheet for a list of supported device/package combinations.

To modify the UCF file provided with the reference design, lines 1-531 should be replaced with information from the UCF files provided with the SPI-4.2 v6.1 core for the particular device/package combination. Where hierarchy is used, an additional top-level of hierarchy must be added to the path (*spi3_to_spi4_top0*). In addition, lines 814 through the end of the file must be changed to reflect the new device/package combination. This section of the file includes the DCM, BUFGMux, and BRAM placement of the bridge design and pin locations for the SPI-3 core. These may be changed to locations of the user's requirements. The section with pin locations for the SPI-3 core also includes constraints to make the design meet timing. These constraints may not be suitable in a new device/package combination.

In addition to modifying the UCF file, use the NCF file provided with corresponding SPI-4.2 netlist.

Resource Requirements

The slice count for the design is 7232 slices (50% of the Virtex-II 3000 slices) and the block RAM count is 51 (53% of Virtex-II 3000 block RAMs). The block RAM usage can be increased or decreased by adjusting the size of the internal bridge FIFOs (see **Changing FIFO Depth** section later in this document).

SPI-4.2 Core: Static vs. Dynamic Alignment

The SPI-4.2 core implemented with the bridge design is the dynamic alignment core. If dynamic alignment is used with the bridge design, then no modification is required to the provided code. However, if static alignment is desired, modify the file `spi_clk_startup.v/vhd` by commenting out the entire procedure snk_fsm. This procedure uses the PhaseAlignRequest/PhaseAlignComplete signals to ensure the dynamic alignment logic is successfully trained, and is not required in the static alignment implementation. Instead, simply drive the PhaseAlignRequest signal to a constant *0*.

Additionally, modify the UCF file provided with the reference design as described in the Device and Package Requirements section above. Also, use the NCF file provided with the static Alignment netlist.

Changing FIFO Depth

For a different size FIFO:

- Open the Xilinx CORE Generator™ tool
- Click *Create a New Project*
- Browse to the `implement/hdl/<verilog|vhdl>` directory
- Set *Output Options* to *Flow Vendor*, *Design Entry*, *VHDL*, or *Verilog*, the design tool to *Synplify*, the *Target Architecture* to *Virtex2*, and *Overwrite Files* to *true*.
- Click *OK*
- Select the menu item File -> Recustomize core…
- Browse to `hdl/generic_sfifo_512x72.xco`, and click *Open*
- The FIFO depth may now be changed to a different size

A larger FIFO requires more block RAM, this must be considered before implementing a FIFO change.

Changing to an Asynchronous FIFO

Using different clocks for clocking the user side of the SPI-4.2 cores and the user side of the SPI-3 cores, causes the bridge logic to cross clock domains. To make this change, change the `generic_sfifo_512x72.xco` to an asynchronous FIFO, generated using the CORE Generator system.

Since parameter sets *CSET* is different for asynchronous FIFO and synchronous FIFO, we recommend generating a temporary asynchronous FIFO and then modifying async FIFO's XCO file close to synchronous FIFO's XCO file. Finally, regenerate the asynchronous file you need for the bridge design.

In addition, the module `hdl/<verilog|vhdl>/spi3_to_spi4_burst_storage.v/vhd`, the module `hdl/<verilog|vhdl>/spi4_to_spi3_burst_storage.v/vhd`, and the wrapper files must be modified in the following ways:

- Both user-side clocks are entered into the modules. A port is added to the modules. This extra port must be reflected in all levels of wrapper files.
- The generic_sfifo_512x72 instantiations in each module are changed to reflect the new asynchronous FIFO module.
- The empty, almost empty, half full, etc. signals must be appropriately driven by signals from the newly instantiated FIFO.
- In the file `hdl/<verilog|vhdl>/spi3_to_spi4_burst_storage.v/vhd`, a packet count is generated in the process create_pkt_cnt that results in the signals PktCntEq0 and PktCntEq1. This process then compares signals crossing clock domains. Appropriate synchronization design techniques must be followed during this modification to result in minimal glitching and metastability.
- Finally, change all levels of wrapper files so each module is clocked by the appropriate clock.

Changing Channel Mapping

Currently, the SPI-4.2 channel 0 is mapped to SPI-3 core instantiation 0, channel 1 is mapped to instantiation 1, and so on. To change the mapping, only two simple changes are necessary. In the SPI-4.2 to SPI-3 direction, the channel mapping is determined in the file `hdl/spi4_to_spi3_burst_storage.v/vhd` in the sequential block *write_enable_proc*. To change the mapping, modify the address contained in the *if* statement within this sequential block. In the SPI-3 to SPI-4.2 direction, the mapping is determined in the file `hdl/<verilog|vhdl>/spi3_to_spi4_core.v/vhd`, in the spi3_to_spi4_burst_storage instantiations. To change the mapping, change all port mappings of the `SPI3_<port number>` and `Brst_<port number>` to reflect the desired mapping.

Changing Arbitration Schemes

Currently, in the SPI-3 to SPI-4.2 direction, the arbitration to determine the SPI-3 core data to be transferred is determined in a round-robin fashion. If a different arbitration scheme is desired, modify the file `hdl/<verilog|/vhdl>/spi3_to_spi4_arbiter.v/vhd`. The process used is the *schaddr* process.

Conclusion

The SPI-4.2 to Quad SPI-3 Bridge reference design effectively moves packets from one data rate to another and offers a bridge between two standard devices supporting different interfaces.

The SPI-4.2 to Quad SPI-3 Bridge reference design code is available at ftp://ftp.xilinx.com/pub/applications/xapp/xapp525.zip.

References

- POS-PHY Level-3 Link Layer Core v3.2 Product Specification
- SPI-4.2 (PL4) Core v6.1 Product Specification

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
06/24/03	1.0	Initial Xilinx release.
03/26/04	1.1	Updated document to indicate Verilog support implemented in the code and testbench for SPI-4.2 to Quad SPI-3 Bridge reference design.
10/15/04	2.0	Updated document to indicate change in core versions, SPI-3 v3.1 to SPI-3 v3.2 and SPI-4.2 v5.2 to SPI-4.2 v6.1.