ECP5™, ECP5-5G™ – PCI Express Demo Verilog Source Code (Basic, Throughput and Scatter-Gather DMA) User Guide

UG106 Version 1.1, October 2016

Introduction

This user guide provides details of the Verilog code used for the ECP5™ and ECP5-5G™ PCI Express Basic Demo, the PCI Express Throughput Demo, and the PCI Express Scatter-Gather DMA (SGDMA) demo.

Note: The PCI Express reference design for the ECP5-5G Versa Development Board does not currently support the SG-DMA Demo.

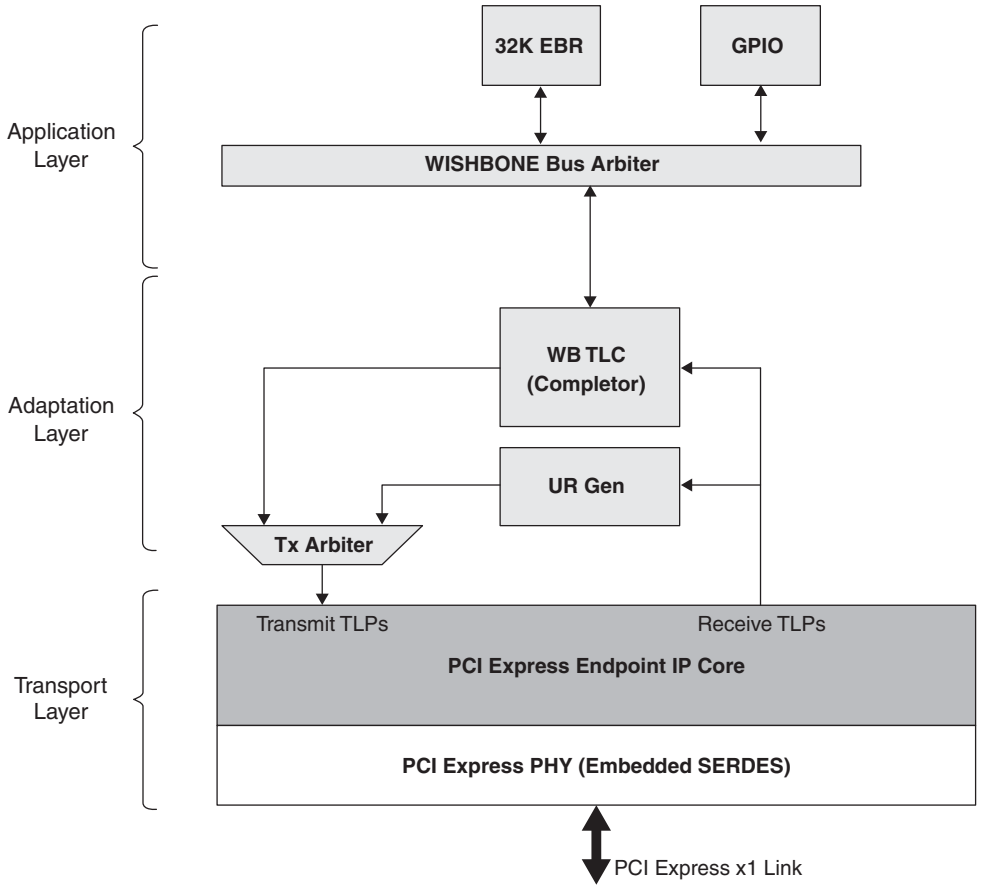
The basic demo is described first, with a block diagram of the design is provided followed by a description for each module in the design. For the Throughput and SGDMA demos, a block diagram is provided followed by a description of each unique module used in the demos. Instructions for building the demo designs using Lattice Diamond® design software are provided as well as a review of the preference files used for the demo. In this document, *ECP5* may be used to generally refer to both ECP5 and ECP5-5G devices.

PCI Express Basic Demo

Top Level

Figure 1 provides a top-level diagram of the demo Verilog design.

Figure 1. PCI Express Basic Demo Block Diagram



The block diagram is separated into three distinct functions. The Transport Layer is used to move TLPs to and from the PCI Express link. This set of features is supported by the Lattice PCI Express Endpoint IP core and embedded SERDES. Moving up the design stack, the next layer is the Adaptation Layer. This layer is responsible for converting PCI Express TLPs into useful data for the Application Layer. The Adaptation Layer uses several soft-IP Verilog modules to extract received TLP contents and repackage transmit data into TLPs for transmission. The final level is the Application Layer. The Application Layer provides the demo capability that is utilized by the PC Demo software.

Basic Project Directory Structure

Figure 2 shows the basic project directory structure. The project files are divided into Clarity, Implementation, and Source directories.

Figure 2. Directory Structure

```
Versa_PCIeBasic/
        Clarity/
                ecp5um[5G]-45F-Versa/
                        pcie[2]_core/
                                pcie[2]_core/
                                        pcie[2]_extref/
                                        pcie[2]_x1
        Implementation/
                ecp5um[5G]-45F_PCIeBasic/
                        impl1/
        Source/
                32kebr/
                ecp5/
                gpio/
                ur_gen/
                wb_arb/
                wb_tlc/
```

Modules

This section discusses the details of each of the modules that make up the basic demo design. Each module listed will be followed by the file name of the Verilog file or Clarity Designer file which includes this module. Verilog files can be found in the *Source* directory of the demo package, while Clarity Designer files are found in the *Clarity* directory.

Top Level – ECP5/top_basic.v

The top level module contains all of the blocks and connections found in Figure 1. The top level demo is unique for each demo version, and designed for use with the ECP5 Versa board.

PCI Express Endpoint IP Core & Embedded SERDES – PCIE_Core.sbx

The PCI Express Endpoint IP Core and the Embedded SERDES settings are all included in the Clarity Designer sbx module. The sbx module is used by the project to bring in the encrypted IP core for the PCI Express Endpoint. The PCIE Express endpoint used in this design is the native Lattice x1 PCI Express Endpoint IP core.

The ECP5 embedded SERDES settings are also included in the sbx module as part of the PCI Express Endpoint IP core.

Tx Arbiter – ip_tx_arbiter.v

This module allows several clients to send TLPs to the PCI Express Endpoint IP core. Using a round-robin scheduler, the tx arbiter waits for the current client to finish sending TLPs before switching to the next client.

Rx Credit Processing – ip_rx_crpr.v

This module monitors receive TLPs coming from the PCI Express Endpoint IP core and terminates credits for Posted and Non-Posted TLPs which are not handled by the wb_tlc module. The number of credits used by each TLP is calculated and using the PCI Express Endpoint IP core ports these credits are terminated. Terminated credits are stored until an UpdateFC DLLP can be sent by the PCI Express Endpoint IP core informing the far end that the credits have been freed.

Unsupported Request Generation – ur_gen.v

All Non-Posted TLPs and memory accesses to unsupported memory must provide a completion. This module will generate unsupported request completions informing the far-end device of these types of memory transactions.

This module receives input from both the PCI Express Endpoint IP core rx_us_req signal as well as TLPs from the receiver. Whenever the rx_us_req signal indicates an unsupported request this module will send an unsupported request completion to the PCI Express Endpoint IP core.

This demo does not support I/O requests. Whenever an I/O request is made this module will send an unsupported request completion to the PCI Express Endpoint IP core.

This demo implements two BARs (BAR0 and BAR1). If a memory request is made to an address other than that serviced by BAR0 and BAR1 an unsupported request completion will be sent to the PCI Express Endpoint IP core.

WISHBONE Transaction Layer Completer (WB_TLC) – wb_tlc.v

The WISHBONE Transaction Layer Completer (WB_TLC) is used as a control plane interface for the endpoint. This module accepts received TLPs from the PCI Express Endpoint IP core. If they are memory transactions to either BAR0 or BAR1, the memory transaction is used by the completer. Other the TLP is dropped and is handled by the unsupported request module.

The WB_TLC is responsible for adapting 1DW TLP memory requests into WISHBONE transactions. 1 DW TLPs are only supported since this is a low throughput control plane interface which reduces logic. There are several modules underneath the WB_TLC top level.

TLP Decoder – wb_tlc_dec.v

The WB_TLC TLP decoder is responsible for decoding the type of TLP that enters the WB_TLC. The WB_TLC is only capable of supporting MRd and MWr TLPs that are accessing BAR0 or BAR1. All other TLPs are dropped and presumably handled by other modules in the design. After leaving the decoder, MRd and MWr TLPs write into a FIFO.

Memory Request TLP FIFO – wb_tlc_req_fifo.v

The request FIFO is used to store accepted TLPs until they can be converted into WISHBONE transactions on the WISHBONE bus. This request FIFO provides two clock domains (write and read), however this demo design connects both ports to the single 125MHz clock domain.

TLPs are read from the FIFO when the FIFO is no longer empty under control of the WISHBONE interface module. This module terminates write credits when the TLPs are pulled from the FIFO. Read credits are terminated when the completion is sent.

Credit Processor – wb_tlc_cr.v

This module converts credits terminated in the WISHBONE clock domain to the PCI Express domain.

WB_TLC WISHBONE Interface – wb_intf.v

The WISHBONE interface of the WB_TLC is responsible for reading TLPs from the request FIFO and creating WISHBONE transactions. When the request FIFO is not empty, the WISHBONE interface will read the TLP from the FIFO until the end of the TLP. As the TLP is read, the address and data will be converted into a WISHBONE transaction. The address is adjusted to use the least significant 18 bits.

For read transactions the transaction ID is passed out of the module to be used by the completion generation module.

WB_TLC Completion Generation – wb_tlc_cpld.v

The WB_TLC completion generation module is responsible for accepting data from a read request on the WISHBONE bus and creating a CpID TLP. As data is returned from a read on the WISHBONE bus the CpID TLP is filled with this data. The CpID also uses this transaction ID and length from the wb_intf module to fill the remaining fields in the CpID. Once the CpID TLP is created it is stored in the CpID FIFO.

WB_TLC Completion FIFO – wb_tlc_cpld_fifo.v

The completion FIFO stores the CpID TLPs from the completion generation module until the TLP can be sent to the PCI Express Endpoint IP core.

WISHBONE Arbiter – wb_arb.v

The WISHBONE arbiter is responsible for arbitrating between the WB_TLC and the Unsupported Request module for access to the WISHBONE bus. It is also responsible for the slave select based on an address decode from the master selected. To account for the demo applications features, the arbiter does not support all masters transacting with all slaves. The WB_TLC, however, can request data from all slaves on the WISHBONE bus.

GPIO – wbs_gpio.v

The General Purpose Input Output (GPIO) module is responsible for several housekeeping functions. It provides an ID register used by the software to identify the feature set and version of the design. It also provides access to control the 16-segment LED on the board. There is a section dedicated to interrupt control logic as well as other maintenance type functions. Table 1 is a memory map for the GPIO module.

Table 1. GPIO Module Memory Map

Address	Bits	Description
0x0	[0:31]	ID register
0x4	[0:31]	Scratch pad
0x8	[0:15]	DIP switch value
	[16:31]	14 segment LED
0xc		Generic down counter control
	[0]	Counter run
	[1]	Counter reload
0x10	[0:31]	Counter value
0x14	[0:31]	Counter reload value
0x18		
	[0:4]	DMA Request (per channel)
	[5:9]	DMA Ack (per channel)
0x1c	[0:31]	DMA Write Counter – The number of clock cycles from the DMA request to the DMA ack of channel 0
0x20	[0:31]	DMA Read Counter – The number of clock cycles from the DMA request to the DMA ack of channel 1
0x24	[0:15]	Root complex Non-Posted Buffer size
	[16:31]	Root complex Posted Buffer size
0x28	[0:31]	EBR Filter value used for triangle manipulation
0x2c		Not used
0x30	[0]	ColorBar reset
0x100	[0:31]	Interrupt Controller ID

Address	Bits	Description
0x104	[0]	Current status of interrupt
	[1]	Test mode
	[2]	Interrupt enable to pass interrupt onto the PCI Express Endpoint IP core
	[3:7]	Not used
	[8:15]	Interrupt Test value 0
	[16:23]	Interrupt Test value 1
	[24:31]	Not used
0x108	[0:15]	If in normal mode: [0:4] dma_ack [5] down counter = 0 If in test mode: [0:7] Test value 0 [8:15] Test value 1
0x10c	[0:31]	Interrupt mask for all sources

32K EBR – wbs_32kebr.v

The 32K EBR is used to store data on the WISHBONE bus. The WISHBONE slave is 64 bits wide and supports burst operations on the bus.

LED Status – led_status.v

This module provides the control of the LEDs on the demo board for the LTSSM states.

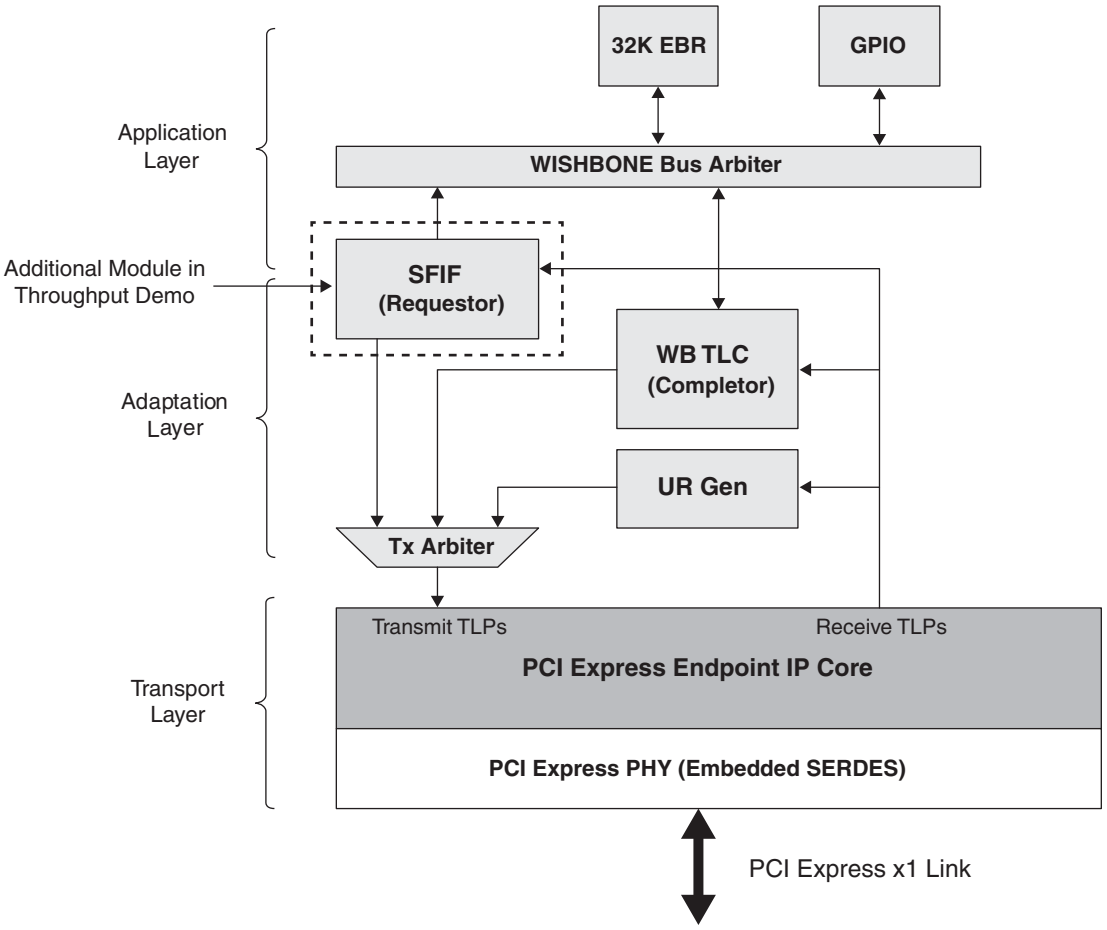
PCI Express Throughput Demo

This section of the user's guide provides details of the Verilog code used for the Lattice PCI Express Throughput demo (also known as the SFIF demo). A block diagram of the entire design is provided (with updated / additional blocks from the basic demo highlighted). A description of each updated or additional module is provided.

Top Level

Figure 3 provides a top level diagram of the Lattice PCI Express Throughput Demo (also known as the SFIF demo).

Figure 3. Top-Level Block Diagram



The block diagram is separated into three distinct functions. The Transport Layer is used to move TLPs to and from the PCI Express link. This set of features is supported in the same fashion as the basic demo, using the Lattice PCI Express Endpoint IP core and embedded SERDES. Moving up the design stack, the next layer is the adaptation layer. This layer is responsible for converting PCI Express TLPs into useful data for the Application layer. The Adaptation layer uses several soft-IP Verilog modules to extract received TLP contents and repackage transmit data into TLPs for transmission. The final level is the Application layer. The Application Layer provides the demo capability that is utilized by the demo software. The SFIF module (unique to this demo) is loaded with TLPs to be sent via the PCI Express link.

SFIF / Throughput Project Directory Structure

Figure 4 shows the basic project directory structure. The project files are divided into Clarity, Implementation, and Source directories.

Figure 4. Directory Structure

```
Versa_PCIeThruput/
        Clarity/
                ecp5um[5G]-45F-Versa/
                        pcie[2]_core/
                                pcie[2]_core/
                                        pcie[2]_extref/
                                        pcie[2]_x1
        Implementation/
                ecp5um[5G]-45F_PCIeThroughput/
                        impl1/
        Source/
                32kebr/
                ecp5/
                gpio/
                sfif/
                ur_gen/
                wb_arb/
                wb_tlc/
```

Modules

This section will discuss the details of each of the modules that are updated or added to the basic demo in order to build the SFIF / Throulghput demo. For reference on all other modules, see the Basic Demo Modules. The Verilog files can be found in the *Source* directory of the demo package.

Top-Level – top-sfif.v

The top-level module contains all of the blocks found in Figure 3. These are the same blocks as the basic demo, with the key addition of the SFIF module.

SFIF – sfif.v

The SFIF is used to move data across the PCI Express link as fast as possible. The design uses a transmit FIFO that is loaded by the control plane via the WB_TLC. When the software starts the SFIF the FIFO contents are sent to the PCI Express Endpoint IP core as fast as the credits available and tags available will allow. For read requests, the CpID is stored into a shallow receive FIFO. The software can analyze the receive FIFO for the correct contents of the FIFO via the WB_TLC.

The SFIF uses several modules to provide the functionality described.

WISHBONE Slave – sfif_wbs.v

The WISHBONE slave interface of the SFIF is a slave on the WISHBONE bus. This interface is used to load the tx FIFO, read the rx FIFO, and set the parameters of the SFIF run.

Transmit FIFO – sfif_tx_fifo.v

The transmit FIFO is used to store the TLPs to be sent by the SFIF. The entire contents of the TLP are written into Tx FIFO in TLP form.

Credit Available – sfif_ca.v

This module is used to compare if enough credits are available to send a TLP to the PCI Expresss Endpoint IP core interface. The number of credits of the next TLP to be sent is compared to the number of credits available from the PCI Express Endpoint IP core.

Tag Available – sfif_tag.v

When performing a read operation a tag needs to be available before a read request can be sent. For each read TLP, the SFIF increments the tag used up to 31 tags. After 31 tags, the tag rolls over to 1. If tag 1 is not available then the SFIF must wait until tag 1 is available.

Receive FIFO – sfif_rx_fifo.v

The receive FIFO module is used to store CpID TLPs from the PCI Express core. These CpID TLPs must match the tag of the outstanding request otherwise the CpID is dropped.

Receive Credit Processing – sfif_cr.v

Credits for the received CpIDs are terminated as they come into the SFIF. This module accepts the CpIDs and frees up the credits on the PCI Express Endpoint IP core interface.

Control – sfif_ctrl.v

The SFIF control module is responsible for controlling the read of the transmit FIFO and sending TLPs to the PCI Express Endpoint IP core. It is responsible for making sure that the credits are checked before sending and handling the requests and ready indications from the core. It also handles all of the cycles and inter cycle gap parameters than can be set by the user.

PCI Express Scatter-Gather DMA (SGDMA) Demo

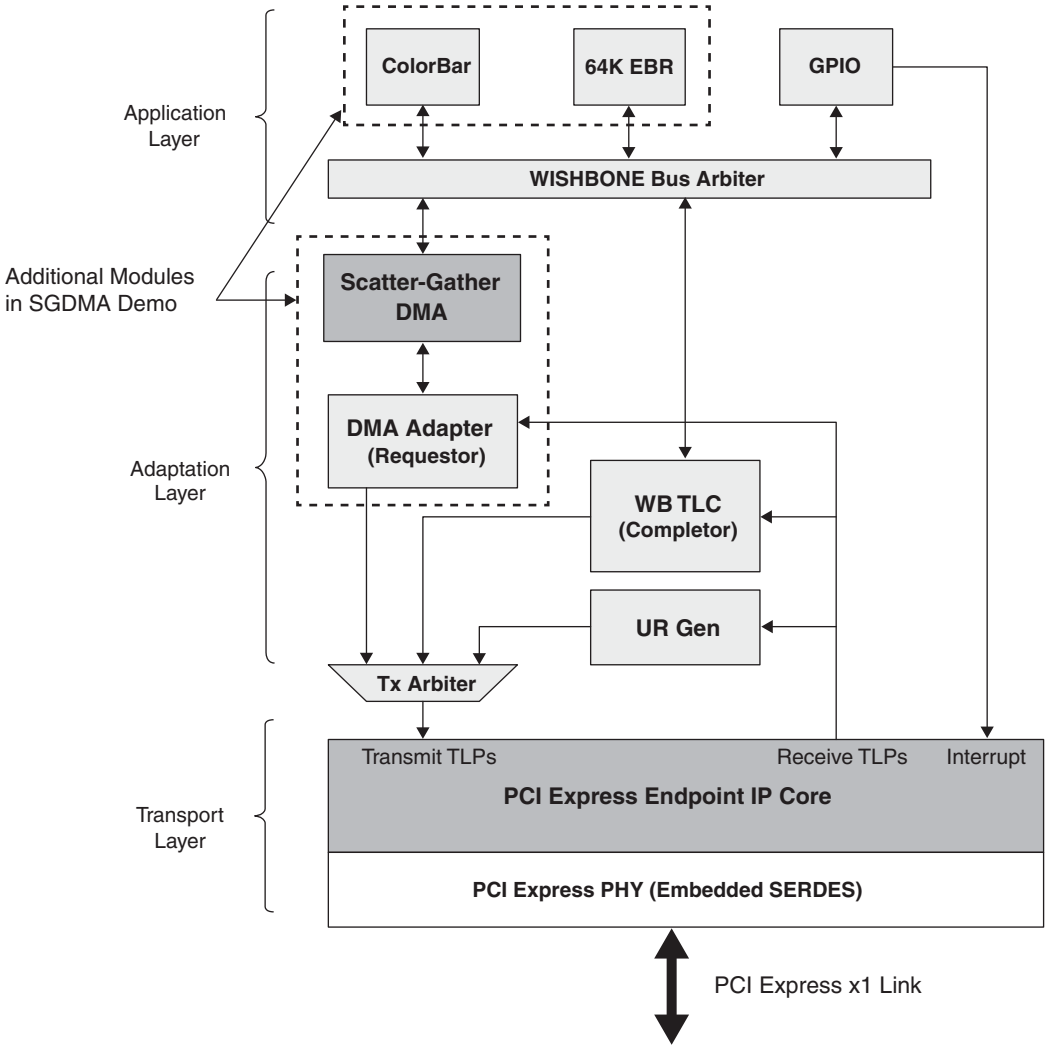
This section of the user's guide provides details of the Verilog code used for the Lattice PCI Express SGDMA demo. A block diagram of the entire design is provided (with updated / additional blocks from the basic demo highlighted). A description of each updated or additional module is provided.

Note: The PCI Express reference design for the ECP5-5G Versa Development Board does not currently support the SG-DMA Demo.

Top Level

Figure 5 provides a top level diagram of the Lattice PCI Express SGDMA Demo.

Figure 5. Top Level Block Diagram



The block diagram is separated into three distinct functions. The Transport Layer is used to move TLPs to and from the PCI Express link. This set of features is supported in the same fashion as the basic demo, using the Lattice PCI Express Endpoint IP core and embedded SERDES. Moving up the design stack, the next layer is the adaptation layer. This layer is responsible for converting PCI Express TLPs into useful data for the Application layer. The Adaptation layer uses Lattice's Scatter-Gather DMA IP core as well as several soft-IP Verilog modules to extract received TLP contents and repackage transmit data into TLPs for transmission. The final level is the Application

layer. The Application Layer provides the demo capability that is utilized by the demo software. In this particular demo application a color bar and modified floating triangle are displayed on the monitor for the user.

SGDMA Project Directory Structure

Figure 6 shows the basic project directory structure. The project files are divided into Clarity, Implementation, and Source directories.

Figure 6. Directory Structure

```
Versa_PCIeSGDMA/
        Clarity/
                ecp5um[5G]-45F-Versa/
                        pcie[2]_core/
                                pcie[2]_core/
                                        pcie[2]_extref/
                                        pcie[2]_x1
                pll/
                sgdma/
        Implementation/
                ecp5um[5G]-45F_PCIeSGDMA/
                        impl1/
        Source/
                64kebr/
                colorbar/
                dma_adapter/
                ecp5/
                gpio/
                ur_gen/
                wb_arb/
                wb_tlc/
```

Modules

This section will discuss the details of each of the modules that are updated or added to the basic demo in order to build the SGDMA demo. For reference on all other modules, see the Basic Demo Modules. The Verilog files can be found in the *Source* directory of the demo package.

Top-Level – top_sgdma v

The top-level module contains all of the blocks found in Figure 5. These are the same blocks as the basic demo, with the key additions of the SGDMA, DMA Adapter, ColorBar and 64K EBR modules. One additional difference is the clocking of the Memory Request TLP FIFO (see the Basic Demo description). This FIFO uses a 125 MHz write clock (from PCIE module) and a 100 MHz read clock (Wishbone domain).

Scatter-Gather DMA – sgdma_top.v

The Scatter-Gather DMA module is a wrapper to encapsulate the Scatter-Gather DMA IP core and the buffer descriptor memory used by the DMA core.

The Scatter-Gather DMA IP core (sgdma_ip.sbx) is an encrypted IP core which uses a Verilog black box model for synthesis and an obfuscated Verilog simulation model. Clarity Designer is used to create this module. The file sgdma.sbx is located in the Clarity/SGDMA directory. This file can be used to load the setting into Clarity Designer to recreate or modify the module.

The buffer descriptor memory is used to hold all of the buffer descriptors used by the SGDMA. A PMI-based Embedded Block RAM (EBR) is used for the buffer descriptor memory. This demo uses all 256 buffer descriptors as an example. This requires an EBR with a 10-bit address.

DMA Adapter – dma_adapter.v

The DMA Adapter is responsible for interfacing the SGDMA to the PCI Express Endpoint IP core. Like the SGDMA, the DMA adapter uses channels. Channel 0 is used for writing data from the WISHBONE bus (via the SGDMA) to the PCI Express core. Channel 1 is used for reading data from the PCI Express core to the WISHBONE bus (via the SGDMA). The remaining channels of the SGDMA are not used by this adapter implementation.

When the SGDMA does a write operation (from the WISHBONE to the PCI Express) channel 0 is used to write data into the adapter. The burst length of the WISHBONE transaction will be no larger than 128 bytes (the largest size supported by the demo software). Using the burst length information from the SGMDA a TLP is created and stored in the transmit FIFO. When this FIFO is not empty the adapter requests to send this TLP to the PCI Express Endpoint IP core. First the number of available Posted credits is checked and then the TLP is requested to be sent and finally sent.

When the SGDMA does a read operation (from the PCI Express to the WISHBONE) channel 1 is used to terminate a read transaction from the SGDMA. The adapter will immediately issue a WISHBONE retry informing the SGDMA that this transaction will take some time and to wait until a request comes back from the adapter to gather the data.

The adapter then converts the WISHBONE read transaction into a memory read TLP to be sent to the PCI Express Endpoint IP core. This TLP is stored in the transmit FIFO. The TLP is read out of the FIFO when there are enough Non-Posted credits available to send the TLP. A request is made to send the TLP followed by the actual TLP.

A memory read TLP may come back with several CpID TLPs until all of the data has been returned. The adapter will wait and store data until all of the data has been returned. Once all of the data has been returned the adapter will issue a dma_req to the SGDMA alerting the SGDMA that all of the data for that channel has been collected. The SGDMA will then issue a read to gather all of the data. As the data has been read from the FIFO to the SGDMA the credits for these CpIDs are released back to the PCI Express Endpoint IP core.

The DMA adapter uses several modules to provide the functionality described.

WISHBONE Slave – dma_wbs.v

The WISHBONE slave interface of the dma adapter is a WISHBONE slave which talks directly with the SGDMA master interface. This module converts the WISHBONE transaction into a memory TLP.

Transmit FIFO – dma_tx_fifo.v

The transmit FIFO is used to store both the memory write and memory read transactions before sending them to the PCI Express Endpoint IP core.

Credit Available – dma_ca.v

This module is used to compare if enough credits are available to send a TLP to the PCI Express Endpoint IP core interface. The number of credits of the next TLP to be sent is compared to the number of credits available from the PCI Express Endpoint IP core. If enough credits are not available to send a TLP, then the TLP request is not made until the credits become available.

Receive FIFO – dma_rx_fifo.v

The receive FIFO module is used to store CpID TLPs from the PCI Express core. These CpID TLPs must match the tag of the outstanding request otherwise the CpID is dropped. The receive FIFO will then store all of the CpID data until all of the bytes have been received and then issue a dma_req. When the CpID data is read from the FIFO the credits are released to the PCI Express Endpoint IP core.

Control – dma_Ctrl.v

The DMA adapter control module is responsible for controlling the read of the transmit FIFO and sending TLPs to the PCI Express Endpoint IP core. It is responsible for making sure that the credits are checked before sending and handling the requests and ready indications from the core.

64K EBR – wbs_64kebr.v

The 64K EBR is used to store data on the WISHBONE bus. The WISHBONE slave is 64-bit wide and supports burst operations on the bus.

ColorBar – wbs_colorbar.v

This WISHBONE slave is used to generate the color bar display pattern used in the demo. Each time the colorbar is read the pixel information is incremented ultimately producing the colorbar pattern.

PLL – pll.v

In the ECP5 design, the FPGA PLL is used to create the WISHBONE clock domain (100 MHz).

Building the Designs in Diamond

See the instructions found in UG98, PCI Express Demos for the ECP5 Versa Development Board, for instructions on working with these demos in Lattice Diamond software.

There are two special concerns for working with the demo files in Diamond:

- These demo projects enable the use of the IP Hardware Timer. If the user does not have a license for any of the IP used in the design, the IP Hardware Timer will hold the FPGA in global reset after approximately four hours of operation. Once a valid license is installed and the project rebuilt the Hardware Timer will no longer be used.
- If design changes are made, place and route options may need to be adjusted to run multiple placement iterations in order to meet timing.

Conclusion

This user guide provides a description of the source files and designs used in the PCI Express Demos for the ECP5 Versa Development Board (Basic, Throughput, and SGDMA demos). With this user guide, a user can understand the functions of the different files included with the demo and begin to modify the design to achieve their design goals.

Technical Support Assistance

Submit a technical support case through www.latticesemi.com/techsupport.

Revision History

Date	Version	Change Summary
October 2016	1.1	Revised document title to ECP5™, ECP5-5G™ – PCI Express Demo Verilog Source Code (Basic, Throughput and Scatter-Gather DMA) User Guide
		Updated Introduction section. Added content related to ECP5-5G.
		Updated Basic Project Directory Structure section. Revised Figure 2, Directory Structure.
		Updated SFIF / Throughput Project Directory Structure section. Revised Figure 4, Directory Structure.
		Updated PCI Express Scatter-Gather DMA (SGDMA) Demo section. Added note on SG-DMA Demo support.
		Updated SGDMA Project Directory Structure section. Revised Figure 6, Directory Structure.
October 2015	1.0	Initial release.