XAPP869 (v1.0) October 4, 2007

Point-to-Point Connectivity Using Integrated Endpoint Block for PCI Express Designs

Authors: Sunita Jain and Guru Prasanna

Summary

This application note provides a reference design for point-to-point (FPGA to FPGA) high-speed serial packet transfer functionality using the integrated Endpoint block for PCI Express® designs in a Virtex™-5 LXT FPGA.

Two integrated Endpoint blocks for PCI Express functionality are used in the design; one of them (master) is equipped with the ability to configure the other (slave). Data can flow in both directions between these two integrated Endpoint blocks, i.e., in full duplex mode. The user interface to this design is provided through a LocalLink (LL) interface.

The design is capable of operating in x1, x2, x4, and x8 lane configurations.

Hardware Requirements

The design can be demonstrated on an ML523, a Virtex-5 RocketIO™ characterization platform or an ML505, a Virtex-5 RocketIO evaluation platform. The following hardware is also required:

- JTAG cable or platform USB cable
- SMA connector cables
- Super clock module for ML523 boards

Software Requirements

Software requirements include the following:

- ISE™ software, v9.2i SP2 (9.2.02i) or later
- Modelsim 6.1e
- ChipScope™ Pro analyzer, v9.1i or later

PCI Express Standard

The PCI Express standard is a high performance, general-purpose interconnect architecture designed for a wide range of computing and communication platforms. It is a packet-based, point-to-point serial interface, which supports a raw bandwidth of 2.5 Gb/s per lane per direction. Its built-in, robust, credit-based flow control eliminates packet discards due to receive buffer overflow, and the retry feature ensures high reliability of data transfer.

Introduction

This reference design provides a high-speed, low cost, reliable, point-to-point or chip-to-chip connectivity solution using the integrated Endpoint block for PCI Express designs available in the Virtex-5 LXT FPGA. Figure 1 depicts an overall block diagram of the design.

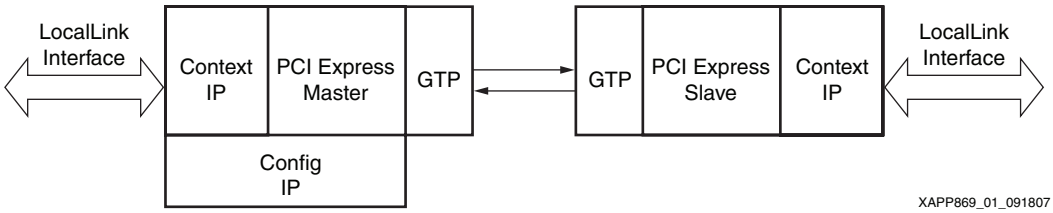

Figure 1: **Block Diagram**

Two integrated Endpoint blocks (one at each end) are utilized in the design. One end acts as a master and configures itself locally. The other end acts as a slave and is configured by the master over the PCI Express link. After configuration, the setup is ready to operate in full-duplex mode for data transmission.

The user interface is implemented to match the LocalLink protocol. During transmission, the user-provided LocalLink frames are converted to transaction layer packets (TLP) before moving to the integrated Endpoint block for transfer. During reception, the TLPs from the integrated Endpoint block are converted to LocalLink frames, and the data, along with the other control signals, is passed to the user through the LocalLink interface.

Design Features and Functionality

- LocalLink framing interface for user interface
  - 64-bit data bus width and 3-bit remainder bus
  - Packetized interface with start of frame (SOF) and end of frame (EOF) to delineate packets
- Packet discontinue feature for the user interface in both transmit and receive directions
- Frame error detection support
- Multi-lane configuration support: x1, x2, x4, and x8
- 1.62 Gb/s or greater throughput per lane
- Automatic initialization, recovery, and channel maintenance, which is transparent to the user application
- Integrated Endpoint block in Virtex-5 LXT device
  - The integrated Endpoint block uses three or six block RAMs for its internal TX, RX, and RETRY buffers
  - All data transfers through the integrated Endpoint block are in the form of memory write TLPs
  - The built-in packet retry feature in the integrated Endpoint block enables error correction at the link layer
- Auto negotiation feature in the integrated Endpoint block allows continued operation of the design with lowered bandwidth if specific lanes become non-operational
- GTP transceivers in the Virtex-5 device provide high-speed transceiver capability
  - 8B/10B encoding and decoding used in the physical layer
  - Clock compensation
  - Automatic clock data recovery
  - Uses up to 8 transceivers
- The Config IP block configures the two integrated Endpoint blocks
- Context IP logic performs TLP generation/decode
- Built-in FIFOs in the Virtex-5 device are used for buffering in the FPGA

Configurable Parameters

Table 1 lists the user-configurable parameters for generating the design.

Table 1: **Parameter List**

Parameter	Allowed Values	Description
Board	ML523 (default) ML505	The user can select either the ML523 (XC5VLX110T-1-FF1136 device) or the ML505 (XC5VLX50T-1-FF1136 device) board. The ML505 board provides SMA connectors for GTP_X0Y4 (GTP1) only. Hence, the ML505 board can support only x1 lane configuration.
Device	XC5VLX110T-1-FF1136	The user enters the target device. The device name must be entered in full as shown.
Number of Lanes	1	The user selects the number of lanes for the design operation. Multi-lane designs are supported only on the ML523.
	2	
	4 (default)	
	8	
Silicon Sample Version	PS (default)	The user selects the silicon version: PS for production device and ES for engineering sample version. The design supports the ES sample for XC5VLX110T device for the GTPs specified in the user constraints file (UCF) for each lane width. For details on how to use different GTP locations or a device for the ES silicon version, see "Generating the Design."
	ES	
Transceiver Locations	Set in UCF	The user can set the GTP locations in the UCF. When using one lane for the ML523 board, GTP0 is always used; selection of GTP1 for one-lane designs in ML523 is currently not supported. For the ML505 board, the GTP location is fixed to GTP_X0Y4 (GTP1). Moving GTP locations farther from the integrated Endpoint block can impact timing.
Design Option	1 (default)	The user selects the design option depending on: the number of block RAMs that the user can spare, throughput required, and user frame size (in bytes). See Table 2 for more details.
	2	

Table 2 details the design options allowed.

Table 2: **Design Option Details**

Design Option	Number of Block RAMs Used	Maximum Throughput Achievable (Gb/s per lane)	Description
1	3	1.62	This option gives throughput of 1.62 Gb/s per lane for user frame size ≥ 512 bytes and programs the maximum payload size (MPS) of the integrated Endpoint block to be 512 bytes.
2	6	1.74	This option gives throughput of 1.74 Gb/s per lane for user frame size ≥ 1024 bytes and a throughput of 1.72 Gb/s per lane for frame size ≥ 512 bytes. Design Option 2 programs the MPS to be 1024 bytes.

User Interface

The user interface is compliant with the LocalLink specification and also supports the optional discontinue signals in LocalLink. Table 3, Table 4, and Table 5 describe the system interface ports, LocalLink transmit, and LocalLink receive ports.

Table 3: **System Interface Ports**

Port	Direction	Description
RST_N	Input	Global Reset - Resets the integrated Endpoint block and the FPGA logic.
GTPRESET_N	Input	GTP Reset - Resets the entire transceiver portion of the FPGA.
USRCLK	Output	User Clock - Provided as output for use with the FPGA logic, it is 125 MHz for x1/x2/x4 designs and 250 MHz for x8 designs. User LocalLink must be run at this clock.
CORECLK	Output	Core Clock - This is the 250 MHz core clock from the integrated Endpoint block.
LINK_READY	Output	Link Ready - Indicates that the link is ready for data transfer. Asserted after the configuration is complete after link up.
LINK_UP	Output	Link Up - Indicates that link training between the two integrated Endpoint blocks was successful and that the link is up.
LINK_WIDTH[3:0]	Output	Link Width - Indicates the negotiated lane width: `0001` : One lane `0010` : Two lanes `0100` : Four lanes `1000` : Eight lanes
CLOCK_LOCK	Output	Clock Lock - Indicates that PLL (phase-locked loop) has achieved clock lock.
SYSTEM_ERROR	Output	System Error - This output indicates that the link is no longer reliable and the system requires a reset.
L0RXMAC_LINK_ERROR[1:0]	Output	Link Error - Bit 1 asserted indicates a receiver error. Bit 0 asserted indicates a link training error.

Table 4: **User Interface Transmit Ports**

Port	Direction	Description
LLUI_TX_DATA[63:0]	Input	Transmit Data from User - Data from the user application to be transmitted.
LLUI_TX_SRC_RDY_N	Input	Source Ready - When asserted, indicates that the data presented on LLUI_TX_DATA input is valid.
LLUI_TX_DST_RDY_N	Output	Destination Ready - When asserted, indicates that the context IP is ready to receive data from the user application.
LLUI_TX_SOF_N	Input	Start of Frame - The current data present on LLUI_TX_DATA port is the first byte(s) of a new frame.
LLUI_TX_EOF_N	Input	End of Frame - The current data present on LLUI_TX_DATA port is the last byte(s) of the current frame.

Table 4: **User Interface Transmit Ports *(Cont'd)***

Port	Direction	Description
LLUI_TX_REM[2:0]	Input	Remainder Bus - Used primarily to indicate the position of the last byte of a frame in a word that is accompanied by an asserted LLUI_TX_EOF_N. The following shows REM field mapping to data: 000: one data byte [63:56] valid 001: two data bytes [63:48] valid 010: three data bytes [63:40] valid 011: four data bytes [63:32] valid 100: five data bytes [63:24] valid 101: six data bytes [63:16] valid 110: seven data bytes [63:8] valid 111: all data bytes [63:0] valid Data byte is defined with respect to LLUI_TX_DATA[63:0].
LLUI_TX_SRC_DSC_N	Input	Source Discontinue - Indicates the cancellation of the current frame being transmitted. Should be asserted in conjunction with LLUI_TX_EOF_N, LLUI_TX_SRC_RDY_N, and LLUI_TX_DST_RDY_N to signal the end of transfer.
LLUI_TX_DST_DSC_N	Output	Destination Discontinue - From the system, indicates the current frame has been discontinued. This can happen either due to system error or the remote end asserting a reset.
FRAME_ERROR	Output	Frame Error - Indicates a LocalLink frame error has occurred. The following conditions are considered framing errors, which result in the assertion of FRAME_ERROR: • Two SOFs without an intervening EOF • Two EOFs without an intervening SOF • SRC_DSC_N without EOF

Table 5: **User Interface Receive Ports**

Port	Direction	Description
LLUI_RX_DATA[63:0]	Output	Received Data to User - Data received by the user application.
LLUI_RX_SRC_RDY_N	Output	Source Ready - When asserted, indicates that the context IP has valid data presented on LLUI_RX_DATA.
LLUI_RX_DST_RDY_N	Input	Destination Ready - When asserted, indicates that the user application is ready to receive data from the context IP.
LLUI_RX_SOF_N	Output	Start of Frame - Current data present on LLUI_RX_DATA port is the first byte(s) of a new frame.
LLUI_RX_EOF_N	Output	End of Frame - Current data present on LLUI_RX_DATA port is the last byte(s) of the current frame.

Table 5: **User Interface Receive Ports *(Cont'd)***

Port	Direction	Description
LLUI_RX_REM[2:0]	Output	Remainder Bus - Used primarily to indicate the position of the last byte of a frame in a word that is accompanied by an asserted LLUI_RX_EOF_N. The following shows REM field mapping to data: 000 : one data byte [63:56] valid 001 : two data bytes [63:48] valid 010 : three data bytes [63:40] valid 011 : four data bytes [63:32] valid 100 : five data bytes [63:24] valid 101 : six data bytes [63:16] valid 110 : seven data bytes [63:8] valid 111 : all data bytes [63:0] valid Data byte is defined with respect to LLUI_RX_DATA[63:0]
LLUI_RX_SRC_DSC_N	Output	Source Discontinue - Indicates the cancellation of the current frame being transmitted. Asserted in conjunction with LLUI_RX_EOF_N, LLUI_RX_SRC_RDY_N, and LLUI_RX_DST_RDY_N to signal the end of transfer.
LLUI_RX_DST_DSC_N	Input	Destination Discontinue - From the user, indicates cancellation of the current frame. LLUI_RX_EOF_N is asserted the next cycle by the context IP in compliance with LocalLink protocol.

Architecture

The entire design has been divided into three main architectural blocks: Config IP, Context IP-transmit, and Context IP-receive.

Config IP

The Config IP block is responsible for configuring the master locally and configuring the slave end via the PCI Express protocol link. After configuration is completed, LINK_READY is asserted, indicating that the link is ready for data transfer. During configuration, the maximum payload size is set on both the master and slave ends.

Depending on the user design option selected, the Config IP programs the integrated Endpoint block maximum payload size (MPS) to be 512 bytes for Design Option 1 and 1024 bytes for Design Option 2.

Context IP Transmit

The Context IP Transmit block uses the concept of the store-and-forward technique to transmit user frames across the protocol link as memory-write TLPs. The TAG field in the memory-write TLP header is undefined and can contain any value; consequently, it is used to store indicators like start of frame, end of frame, discontinue etc., which is helpful in building the LocalLink frame back at the receive end.

The transmitter logic stores the incoming data from the user in a FIFO. The FIFO capacity is equal to the MPS set for a selected design option. A length counter counts the length of the payload that is used to build the length field in the TLP header. The counter wraps around after reaching the highest count value (511 in the case of MPS = 512, and 1023 in the case of MPS = 1024). A length buffer stores the length of the frames.

The built-in FIFO in the Virtex-5 device is used for storage. If the link goes down in between the data transfer (LINK_UP goes Low), then all data currently in the buffer will be lost. The link

going down in between the data transfer is considered to be a reset because the configuration has to be redone.

Figure 2 shows an overall view of the transmit logic.

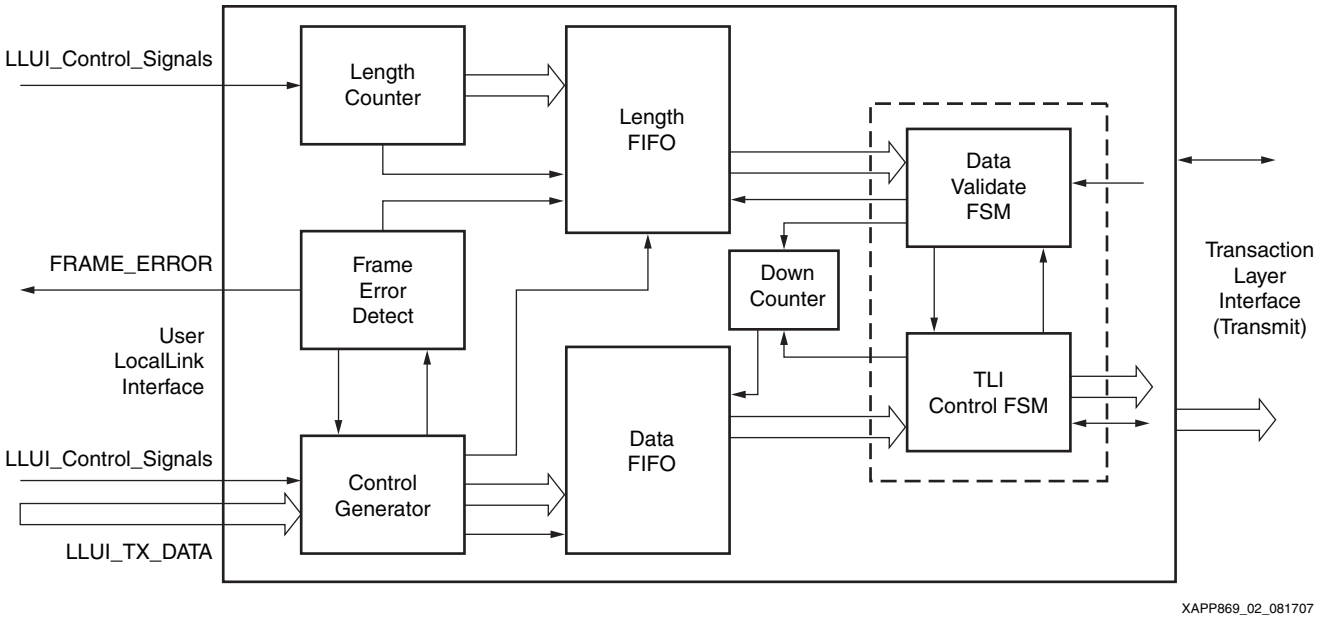

Figure 2: **Context IP Transmit Block Diagram**

For user frame size lesser than the MPS programmed in the integrated Endpoint block, the entire frame is transmitted as a single TLP, as depicted in Figure 3. Signals beginning with "llui_" show the user LocalLink signals, and those prefixed with "tli_" indicate signals at the transmit side of the transaction layer interface.

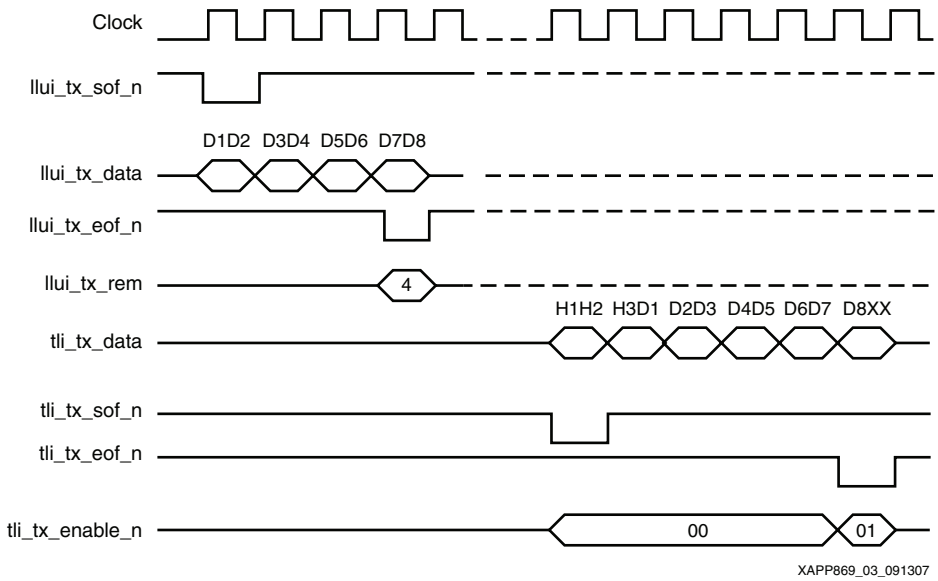

Figure 3: **Frame Transfer across Transaction Layer Transmit Interface**

The user can choose not to transmit frames when the Context IP is ready to receive frames by deasserting LLUI_TX_SRC_RDY_N input.

Segmentation Scheme for User Frame Size Greater Than MPS

For user frame size greater than the MPS, the frame is segmented, with each segment equal to the MPS bytes in size, and then transmitted. For example, in Design Option 1, the MPS equals 512 bytes. If the user frame is greater than 512 bytes in size, after receiving 512 bytes from the user, the data FIFO becomes full, resulting in LLUI_TX_DST_RDY_N deassertion. The length counter reaching the terminal count (511 in this case) triggers a write to the length FIFO. The user segment buffered until this point is sent as a TLP (TLP length = 512 bytes) across the transaction layer interface (TLI). As the data in buffer gets transmitted across the TLI, the data FIFO is ready to accept more data from the user. This readiness is indicated by the assertion of the LLUI_TX_DST_RDY_N signal. Hence, when the user frame size is greater than the MPS, data gets transmitted across the TLI as multiple TLPs.

Discontinue Feature Behavior

The user can discontinue LocalLink frames by asserting the LLUI_TX_SRC_DSC_N signal at the transmit end. The assertion of this signal should be accompanied with LLUI_TX_EOF_N, as shown in Figure 4. If there is no LLUI_TX_EOF_N, FRAME_ERROR is asserted, and the frame is discontinued.

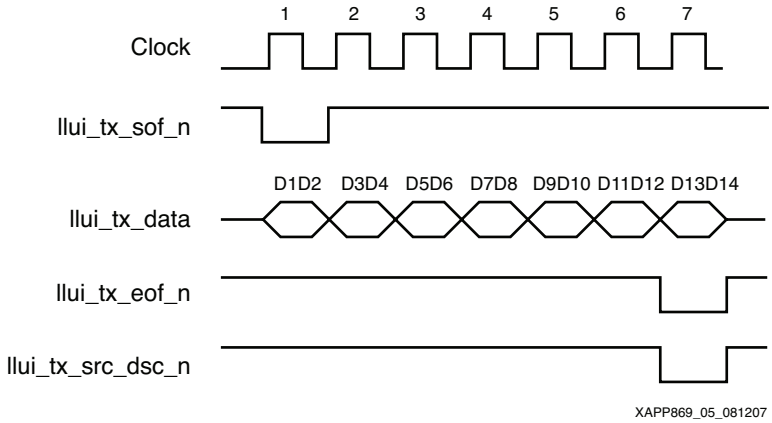

Figure 4: **Discontinue at LocalLink Transmit**

The discontinue scenario from the user is explained below for various cases.

User Frame Less Than or Equal to the MPS: Discontinued

When a user frame that is less than or equal to the MPS is discontinued, the data corresponding to that frame is discarded by the context IP transmit logic. No TLP is transmitted in this case.

User Frame Greater Than MPS: Discontinued

Source Discontinue Received during Second Segment (after First Set of 512 Bytes)

In this scenario, the MPS equals 512 bytes and the user discontinues the frame after sending 600 bytes. The context IP buffers the first 512 bytes and initiates TLP transfer across the TLI for the first segment. As the data for the first segment is being transmitted across the TLI, data for the next segment is being buffered into the data FIFO. When the user asserts LLUI_TX_SRC_DSC_N during the second segment, the TLP under transmission across the TLI is discontinued, and both FIFOs are flushed. LLUI_TX_DST_RDY_N is deasserted during the FIFO flush process. No TLP is transmitted in this scenario.

Source Discontinue Received after a Segment is Transmitted across the TLI

In this discontinue scenario, the user asserts LLUI_TX_SRC_DSC_N when a segment from the user has already been sent across the TLI. For example, the user asserts LLUI_TX_SRC_DSC_N during the third segment. This means the first segment has already been sent across the TLI; the second segment from the user is in the data buffer; and its transmission across the TLI is in progress. During the buffering of the third segment, if the user discontinues the frame, the second segment's transmission across the TLI is discontinued and the FIFOs are flushed by the Context IP logic. Additionally, a one double-word (DW) TLP (a TLP with length = 1DW (32-bits)) with a discontinue indicator bit set in the TAG field is sent across the TLI; this TLP indicates to the receive end that the frame, which previously received a TLP, has been discontinued. At the receive end, the reception of a 1-DW TLP with the discontinue indicator bit set is translated to LLUI_RX_SRC_DSC_N, as shown in Figure 5.

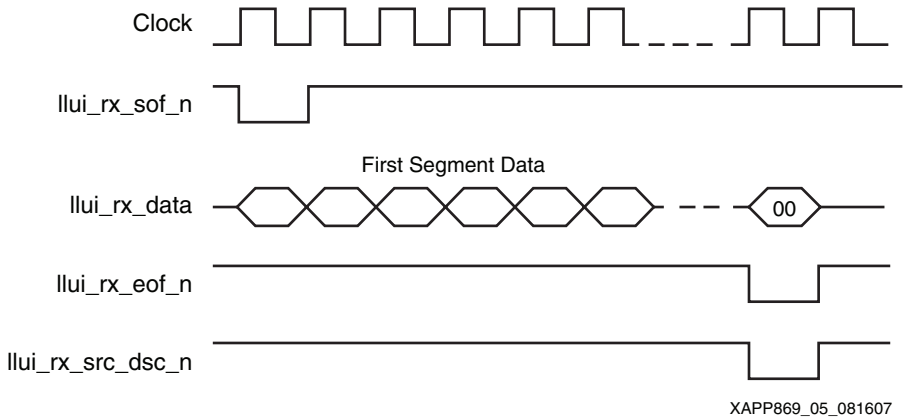

Figure 5: **Discontinue at LocalLink Receive**

Hence, the segment that is part of a discontinued frame and gets transmitted across the TLI is discontinued at the receive end by the assertion of LLUI_RX_SRC_DSC_N. The user should discard the frame for which LLUI_RX_SRC_DSC_N is received.

Context IP Receive

This block receives TLPs from the transaction layer receive interface and then extracts payload out of it. The indicator bits extracted from the TAG field of the TLP are written into the data parity input field of the FIFO at the receive end.

The extracted data along with the proper LocalLink framing signals is sent to the user at the receive end. See Figure 6.

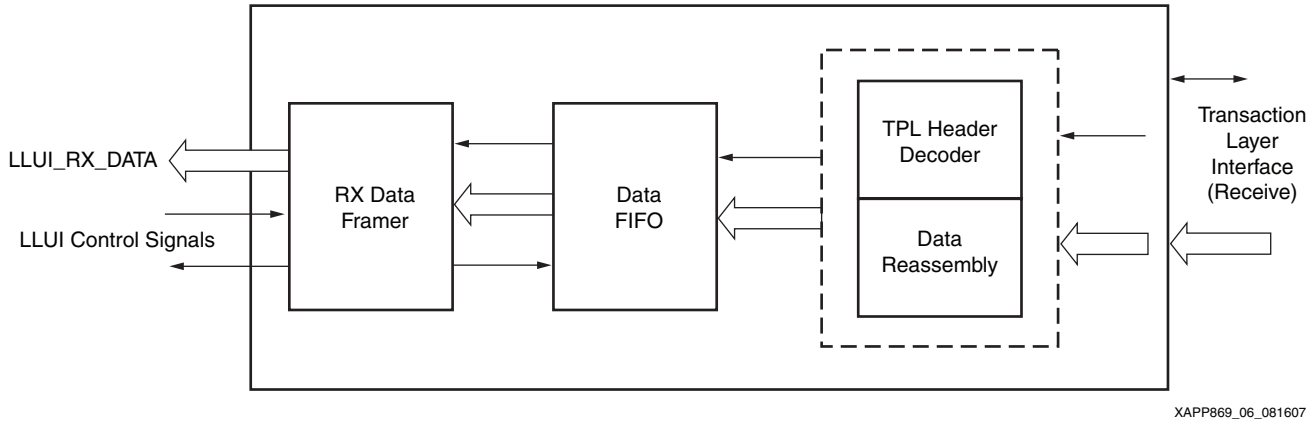

Figure 6: **Context IP Receive Block**

Figure 7 depicts how a TLP received at TLI-receive interface (signals indicated by "tli_rx*") translates into a LocalLink frame at the user-receive interface (signals indicated by “llui_rx*”).

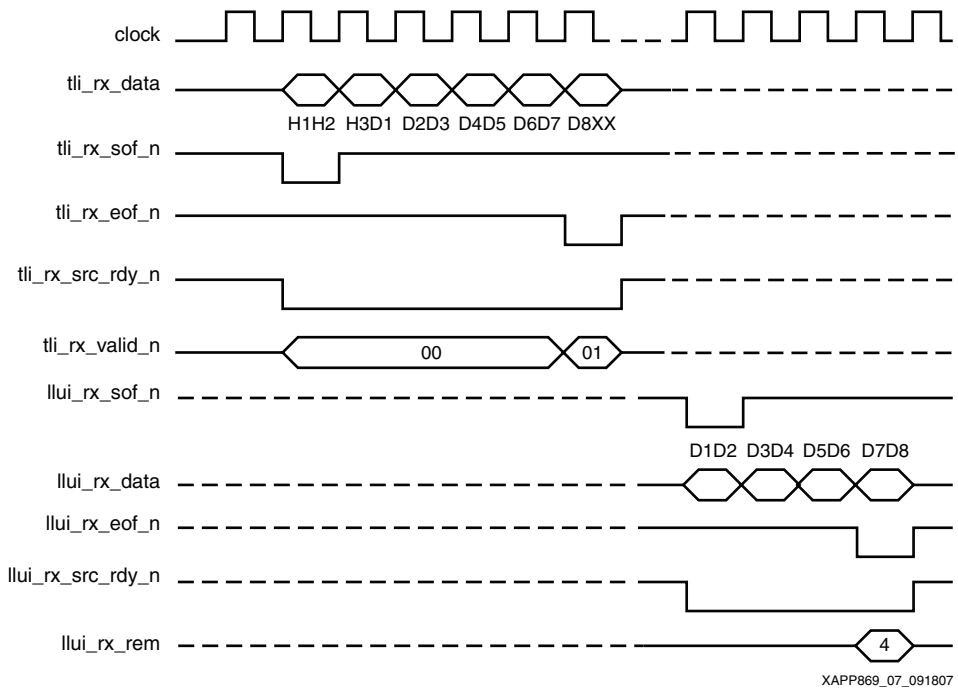

Figure 7: **Frame Transfer across Transaction Layer Receive Interface**

Destination Discontinue at the Receive End

The user at the receive end can also discontinue frames by asserting LLUI_RX_DST_DSC_N. LLUI_RX_EOF_N will be asserted during the next cycle by the Context IP receive logic, in compliance with LocalLink protocol, and the rest of the data in the discontinued frame is discarded. See Figure 8.

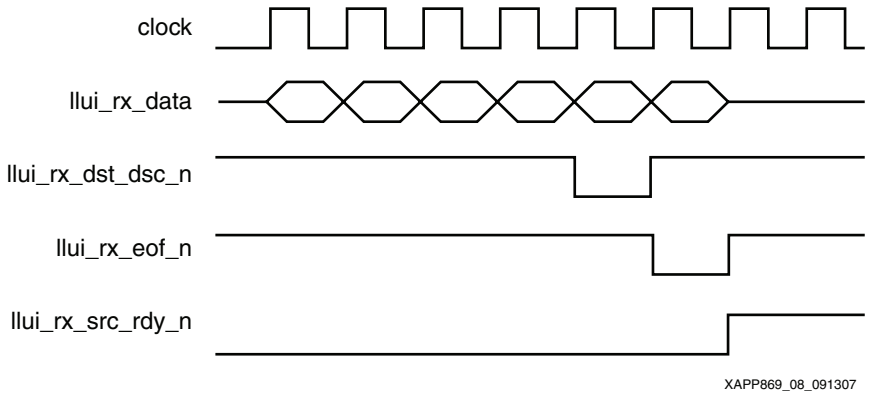

Figure 8: **LLUI_RX_DST_DSC_N Behavior**

Auto-Negotiation

Auto-Negotiation allows the design to recover in a lower lane configuration in the event of a link cable disconnect in a higher lane width design.

This means that a x8 design is capable of recovering in a x1, x2, or x4 configuration; a x4 design is capable of recovering in x1or x2 mode, depending on the lane for which the cable is disconnected.

The auto-negotiation feature allows continued operation of the design with lowered bandwidth if specific lanes become non-operational.

Clocking Requirements

The integrated Endpoint block for PCI Express designs works at an internal core clock frequency of 250 MHz for a line rate of 2.5 Gb/s.

The integrated Endpoint block also provides a clock for operating across the TLI for various lanes. All x8 designs must run the TLI at a clock of 250 MHz to maintain full bandwidth. The TLI can run at a lower frequency of 125 MHz for x1, x2, and x4 lane configurations.

This reference design runs TLI at a USRCLK of 125 MHz for x1, x2, and x4 configurations and at 250 MHz for x8 configuration. The same clock (USRCLK) is provided as an output. The user is required to run LocalLink at the USRCLK provided.

The user is required to input a differential reference clock of 100 MHz for proper operation of the design. The differential clock inputs must be connected to GTP clock inputs as per the UCF.

Example Design

The data transfer capability of the design is demonstrated by an example design. The LocalLink frame generator and frame checker are used for traffic generation and for checking the data received.

The frame generator is deterministic in nature and provides LocalLink user frames with incrementing payload size and various remainder values. The frame checker examines the data received and flags an error output if received data is found to be erroneous. See Figure 9.

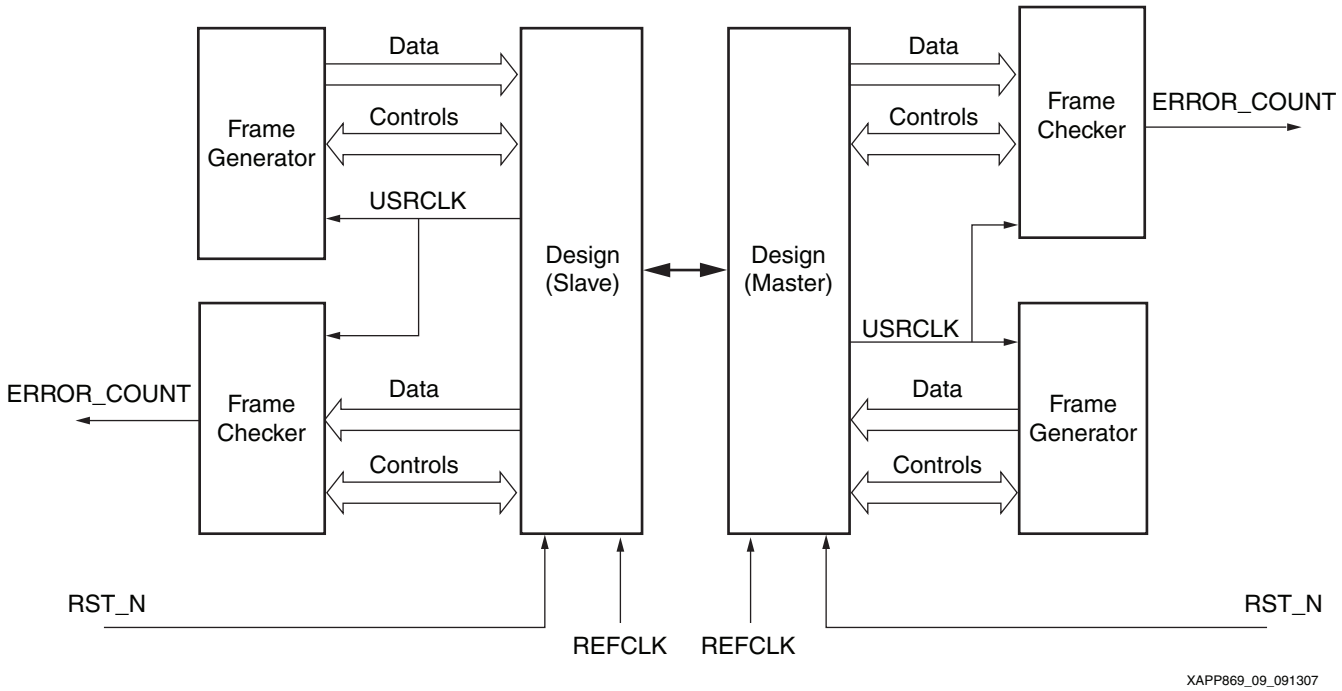

Figure 9: **Example Design to Test Data Transfer**

Design Directory Structure

Figure 10 depicts the design directory structure this reference design.

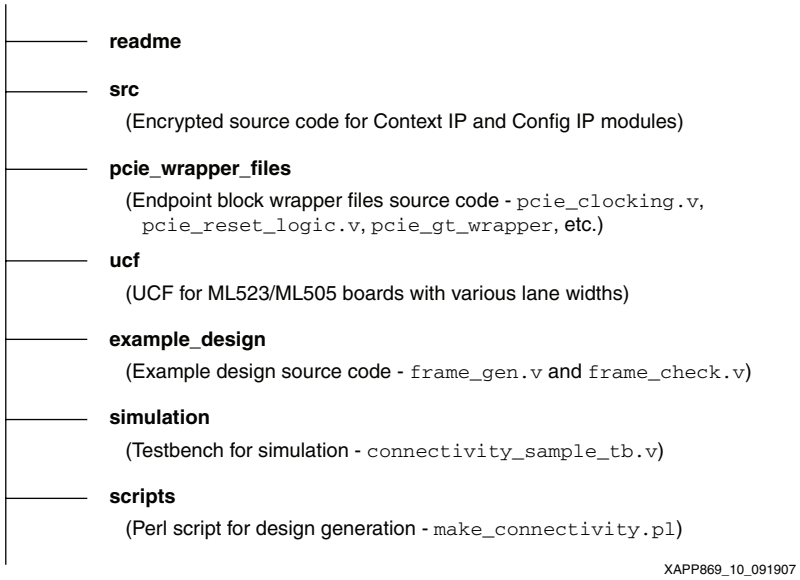

Figure 10: **Design Directory Structure**

Figure 11 depicts the design directory structure after design generation (see "Generating the Design" for details). It shows the new files that get generated.

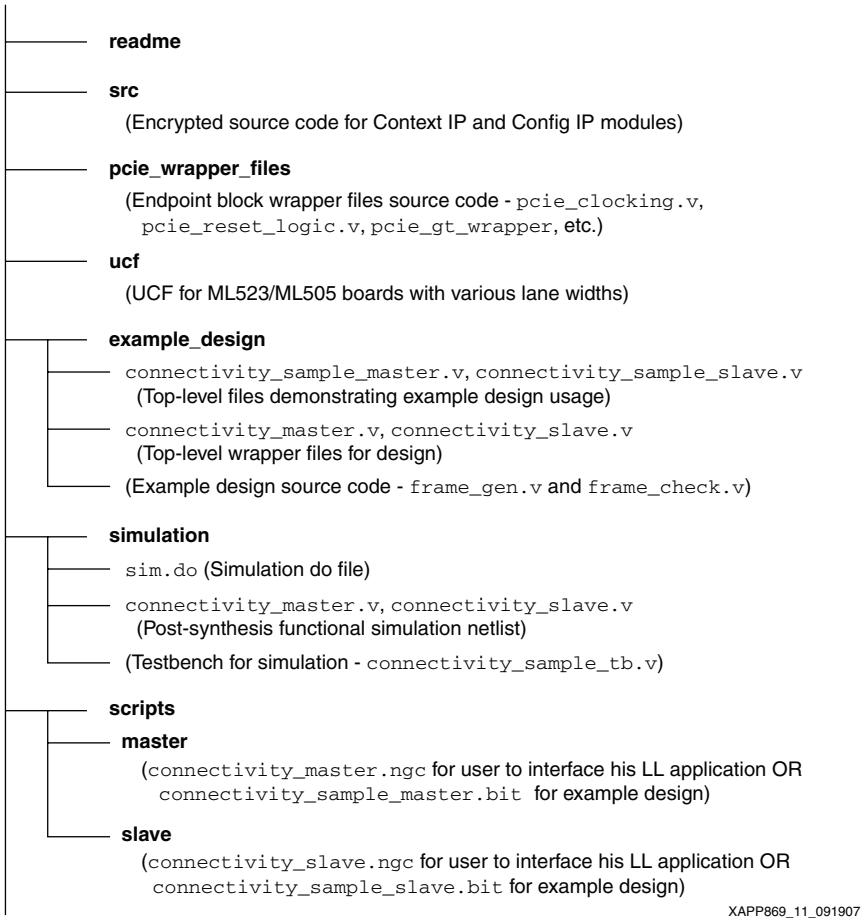

Figure 11: **Design Directory Structure after Design Generation**

Generating the Design

The design source files are delivered in an encrypted fashion. The wrapper files (e.g., clocking module, reset logic, GT-wrapper, etc.) required for the integrated Endpoint block are delivered as source code. An associated `perl` script provided with the design generates a post-synthesis netlist for the user depending on the parameters selected. The `perl` script usage is as shown (it requires the Xilinx environment to be set based on user's installation of ISE software):

```
xilperl make_connectivity.pl -design_option=2 -lanes=2 -board=ML523 -example -ps
```

The parameters, e.g., board, design option and number of lanes, etc., can be entered through command line interface.

The `perl` script parameters are:

- `design_option`: When set to `1`, this sets the MPS value for the integrated Endpoint block at 512 bytes, which uses three block RAMs for various transmit, receive, and retry buffers in the PCI Express protocol. When set to `2`, this sets the MPS at 1024 bytes and uses six block RAMs. Default value of this parameter is set to `1`.
- `lanes`: This selects the number of lanes for the design. Allowed values are 1, 2, 4, and 8. Default value of this parameter is set to 4.
- `board`: This selects the target board for implementation. Allowed options are ML505 (supports only x1) and ML523 (supports all lane configurations). Default value is set to support the ML523 board.
- `example`: This switch generates a bitstream for the example design provided with the ChipScope Pro analyzer and is used to observe the LocalLink frames. Two bitstreams are generated: one for the master end and another for the slave end.
- `device`: This parameter is for the target device. The default value is set to XC5VLX110T-1-FF1136 for the ML523 and XC5VLX50T-1-FF1136 for the ML505.
- `es/ps`: This switch selects either an engineering sample (ES) silicon version or a production device. The ES support is provided for the XC5VLX110T device with the GTP locations mentioned in the UCF for various lane configurations.

To create a design for a different GTP location for ES versions or for a different device for ES versions, the user must generate a `pcie_gt_wrapper.v` file for the specific GTP location or device (and not use the `pcie_gt_wrapper.v` supplied by the reference design). The following steps are required:

1. Generate a design with the required GTP locations and device from the LogiCORE™ IP *Endpoint Block Wrapper for PCI Express,* delivered as CORE Generator™ software (IP).
2. Use the `pcie_gt_wrapper.v` file from the generated IP.
3. Open the `make_connectivity.pl` script and set the path and filename of the variable "$gt_wrapper" to the `pcie_gt_wrapper.v` file generated above. Now the script picks up the `pcie_gt_wrapper` file set by the user.
4. Enter the *full device name* and change the UCF to reflect the correct GTP locations. Run the script and generate the design.

If required, the user can obtain the latest `pcie_gt_wrapper.v` file (for production device in case of a GTP attribute change) from the Endpoint block wrapper for PCI Express designs as explained in steps 1–4 and include it instead of the gt_wrapper file provided with the reference design.

The user can also use the reset logic and clocking modules from the Endpoint block wrapper for PCI Express designs instead of using the modules provided under the `pcie_wrapper_files` folder. To do so, the user has to update the path of these files in the `make_connectivity.pl` script.

However, the user is advised against modifying any parameter values in the wrapper files.

Without the `-example` option, netlists for the design (`connectivity_master.ngc` & `connectivity_slave.ngc`) with given options are generated under the folders titled `master` and `slave` in the `scripts` directory. For ease of use within the user application, the design provides two wrapper files for the top modules. They are named `connectivity_master.v` and `connectivity_slave.v` and are available under the `example_design` folder. Users can use these two files and interface their application to the LocalLink ports provided.

Additionally, UCFs are provided for both the ML505 (XC5VLX50T-1-FF1136 device) and ML523 (XC5VLX110T-1-FF1136 device) boards.

The `perl` script also generates post-synthesis functional simulation `Verilog` files.

To run simulation, the following must be run with the Modelsim (v6.1e) environment set in the `simulation` directory:

```
vsim -do sim.do &
```

Resource Utilization

Table 6 and Table 7 depict the resource utilization by the master and slave designs.

The designs are tested on the ML523 board with an XC5VLX110T device and on the ML505 board with an XC5VLX50T device.

Table 6: **Resource Utilization for the Design**

Number of Lanes	Master Resource Usage LUT/FIFO	Slave Resource Usage LUT/FIFO
1	589/656	551/661
2	591/691	553/728
4	596/761	558/862
8	608/901	569/1130

Table 7: **Resource Utilization including the Example Design**

Number of Lanes	Master Resource Usage LUT/FIFO	Slave Resource Usage LUT/FIFO
1	672/831	636/837
2	674/866	638/904
4	679/936	643/1038
8	716/1074	680/1304

Other resources used by the design include the following:

- Integrated Endpoint block for PCI Express designs (one at each end)
- PLL (one at each end)
- Block RAM
  - 3 at each end for Design Option 1
  - 6 at each end for Design Option 2
- Built-in FIFO in the Virtex-5 device
  - 3 at each end
    - 2 of size 512 x 72 (FIFO36_72)
    - 1 of size 512 x 36 (FIFO18_36)
- Transceivers (1–8) depending on number of lanes selected by the user

Performance

Resource Utilization against Lane Width

Figure 12 shows the resource utilization figures against lane width. The increase in resource utilization is marginal with increase in lane width.

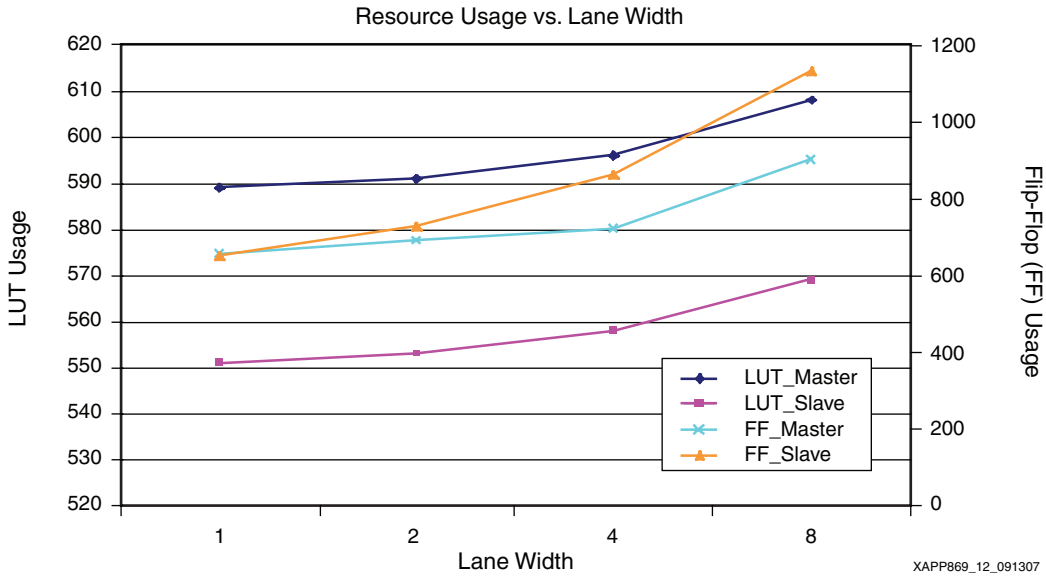

Figure 12: **Resource Utilization vs. Lane Width**

Throughput vs. Frame Size

Figure 13 shows throughput variation (excluding the 8B/10B overhead) against frame size in bytes with different block RAMs.

Selection of Design Option 1 sets the MPS of the integrated Endpoint block to 512 bytes, which uses three block RAMs for various transmit, receive, and retry buffers in the PCI Express protocol implementation. Selection of Design Option 2 sets the MPS to 1024 bytes, which uses six block RAMs.

Throughput results are depicted for the same scenario.

For Design Option 1, any user frame greater than or equal to 512 bytes in size gives a constant throughput of 1.62 Gb/s per lane.

For Design Option 2, any user frame greater than or equal to 1024 bytes in size gives a constant throughput of 1.74 Gb/s per lane. A frame of size 512 bytes gives a throughput of 1.72 Gb/s per lane in this case.

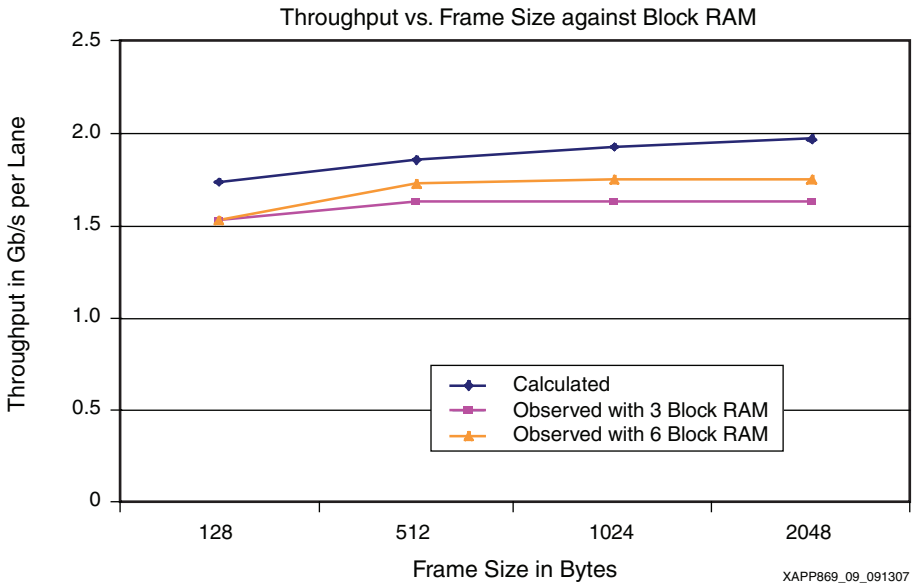

Figure 13: **Throughput against Frame Size**

Reference Design

The reference design is available on the Xilinx website at:
http://www.xilinx.com/bvdocs/appnotes/xapp869.zip

Conclusion

This reference design demonstrates the basic concept of data transfer implementation using the integrated Endpoint block for PCI Express designs.

Two integrated Endpoint blocks (one at each end) are used to achieve a high-speed, low-cost connectivity solution with high data reliability.

The built-in packet-retry feature in the integrated Endpoint block ensures error correction at the link layer, thereby ensuring high reliability of data. The design uses minimal resources (around 600 LUTs) to implement the segmentation and reassembly logic in the FPGA.

The auto-negotiation feature of the PCI Express protocol allows continued operation of the design with lowered bandwidth if specific lanes become non-operational.

The user interface to the design is provided through a 64-bit LocalLink framing interface. An additional LocalLink feature, "discontinue," is supported by this reference design. There is a marginal increase in resource usage with an increase in number of lanes.

References

1. UG197, *Virtex-5 Integrated Endpoint Block for PCI Express Designs User Guide*
2. UG196, *Virtex-5 RocketIO GTP Transceiver User Guide*
3. UG350, *Virtex-5 LogiCORE Endpoint Block for PCI Express Designs User Guide*
4. UG190, *Virtex-5 User Guide*
5. SP006, *LocalLink Interface Specification v2.0*

Revision History

The following table shows the revision history for this document:

Date	Version	Description of Revisions
10/04/07	1.0	Initial Xilinx release.

Notice of Disclaimer