- THIS IS A DISCONTINUED IP CORE -

Fibre Channel Arbitrated Loop v2.3

DS518 March 24, 2008 **Product Specification**

Introduction

The LogiCORE™ IP Fibre Channel Arbitrated Loop (FC-AL) core provides a flexible, fully verified solution for use in any FC-AL port design. The core handles all link initialization and loop arbitration functions and includes credit management capabilities.

Features

- Dual-speed FC-AL core running at 1 Gbps/2 Gbps or 2 Gbps/4 Gbps (Virtex™-4 devices only)
- Designed to *ISO/IEC FDIS 14165-122 FC-AL-2* specification
- Supports all classes of frames: 1, 2, 3, 4, and F
- Dedicated control interface opens and closes circuits on a loop
- Loop initialization protocol handled by integrated MicroBlaze™ microprocessor
- Old Port State Machine fallback for point-to-point operation in non-loop systems
- Port-independent implementation supports underlying functionality required by all arbitrated loop port types (NL, FL)
- Extensive Hardware Verification (Virtex-II Pro FPGAs) at University of New Hampshire Interoperability Lab (UNH IOL) for FC-1 and FC-AL-2 standards

LogiCORE IP Facts				
Core Specifics				
Supported Device Family	Virtex-II Pro, Virtex-4 FX[1], Virtex-5			
Speed Grades	Virtex-II Pro –6 or faster Virtex-4 FX • –10 or faster (1 Gbps/2 Gbps), • –11 or faster (2 Gbps/4 Gbps) Virtex-5 -2 or faster (any SXT or LXT part)			
Performance	1.0625 Gbps, 2.215 Gbps, 4.250 Gbps			
Core Resources				
Resources Used (estimate)	Slices	LUTs	FFs	Block RAMs
Virtex-II Pro FPGAs	5070	7180[2]	3900	22
Virtex-4 FPGAs	5200	7400[2]	3950	22
Virtex-5 FPGAs	2500	6100[3]	3800	14
Provided with Core				
Documentation	Product Specification, Getting Started Guide, User Guide			
Design File Formats	NGC netlist			
Constraints File	UCF (User Constraints File)			
Verification	VHDL Test Bench			
Example Design	VHDL, Verilog			
Design Tool Requirements				
Supported HDL	VHDL, Verilog			
Xilinx Implementation Tools	ISE™ v10.1			
Verification	Mentor Graphics® ModelSim® v6.3c			
Simulation[4,5]	ModelSim PE/SE v6.3c Cadence® IUS v6.1			
Synthesis	XST v10.1			
Support				
Provided by Xilinx, Inc. at www.xilinx.com/support.				

1. Virtex-4 FX solutions require the latest silicon stepping, and are pending hardware validation.
2. Four LUTs
3. Six LUTs
4. Virtex-5 device designs require either a Verilog LRM-IEEE 1364-2005 encryption-compliant simulator or a SWIFT-compliant simulator. For a Verilog LRM-IEEE 1364-2005 encryption-compliant simulator, ModelSim is currently supported; for a SWIFT-compliant simulator, Cadence IUS and Synopsys are currently supported. See the Facts table for supported versions.
5. Virtex-4 and Virtex-II Pro device designs require a SWIFT-compliant simulator. ModelSim, Cadence IUS, and Synopsys are currently supported. See the Facts table for supported versions.

Functional Overview

The FC-AL core supports the FC-0, FC-1, and part of the FC-2 functions described in the Fibre Channel standards. Figure 1 shows a high-level block diagram of the core.

The major functional blocks of the core include

- Client Interface Logic, designed for easy attachment of user logic and simple opening and closing of circuits
- Buffer-to-buffer credit management
- Cyclic Redundancy Check (CRC) generation and verification
- FC-1 layer functions, including clock correction and word synchronization
- Old Port State Machine for point-to-point fallback operation
- Arbitrated Loop Port state machine
- Loop Initialization controller
- Management block

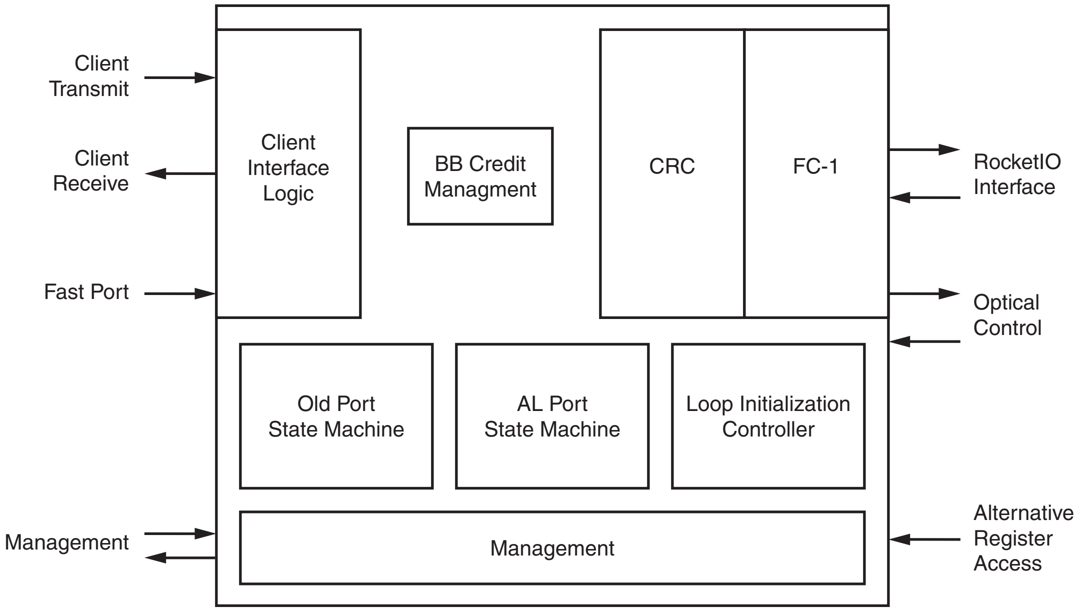

Figure 1: **FC-AL Core Architecture**

Applications

The FC-AL IP core is designed to be used in arbitrated loop systems, with higher-level port and class-specific functions provided by user modules. Figure 2 illustrates a public loop with an FL-port connecting the loop to a fabric switch and NL-ports connecting the other devices on the loop. Figure 3 illustrates an FPGA implementation of an NL-port device.

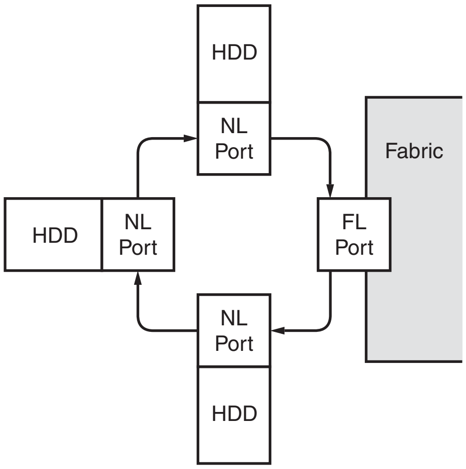

Figure 2: **FC-AL Public Loop**

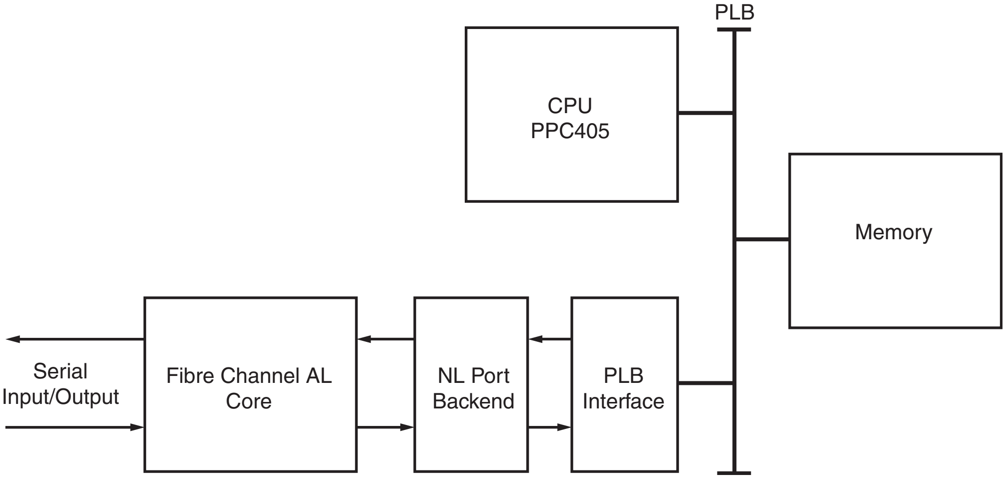

Figure 3: **FPGA Implementation of an NL-port Device**

Core Interfaces

Figure 4 displays a pinout of the FC-AL core netlist with signals grouped according to the functional interfaces.

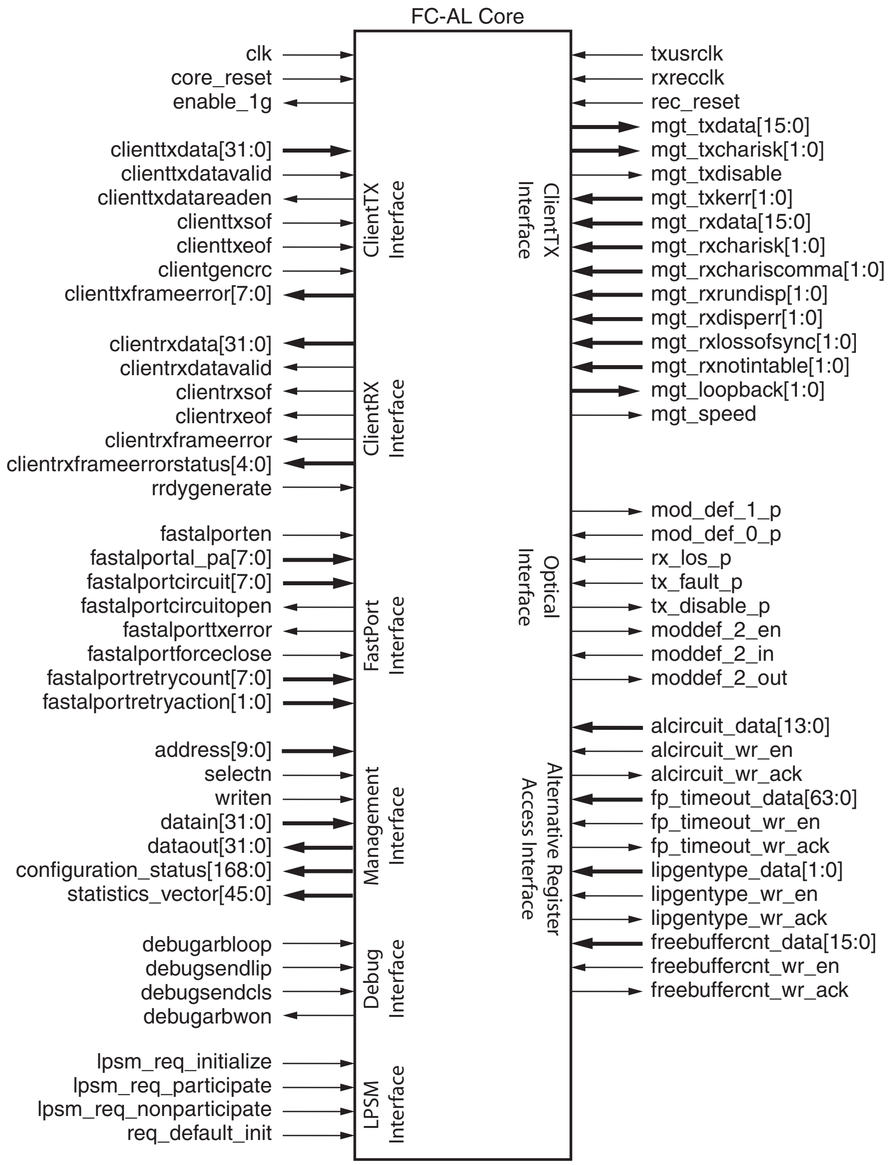

Figure 4: **FC-AL Core Interfaces**

RocketIO Transceiver Interface

The RocketIO™ Transceiver interface signals connect the core to a RocketIO transceiver. Table 1 defines and describes the RocketIO transceiver interface ports.

Table 1: **RocketIO Transceiver Interface Ports**

Name	Direction	Description
mgt_txdata[15:0]	Output	Transmit data to the transceiver in two byte lanes.
mgt_txcharisk[1:0]	Output	Transmit control signals to the transceiver. Indicates which byte lanes are K code-group.
mgt_txkerr[1:0]	Input	Indicates which byte lane(s) have a K code-group to be transmitted which is not a valid K code-group.
mgt_txdisable	Output	Disable the transceiver.
mgt_speed	Output	Indicates the operational speed of the core. For a 1 Gbps/2 Gbps core: • ’0’ denotes 1 Gbps operation • ’1’ denotes 2 Gbps operation For a 2 Gbps/4 Gbps core: • ’0’ denotes 2 Gbps operation • ’1’ denotes 4 Gbps operation
mgt_loopback[1:0]	Output	Selects the two loopback test modes. • Bit 1 is for serial loopback • Bit 0 is for internal parallel loopback • '1' indicates loopback, ’0’ indicates normal operation
mgt_rxdata[15:0]	Input	Received data from the transceiver.
mgt_rxcharisk[1:0]	Input	Indicates which received byte lane(s) are a K code-group. • ’1’ denotes K code-group • ’0’ denotes a D code-group
mgt_rxchariscomma[1:0]	Input	Indicates which received byte lane(s) are a Comma code-group. • ’1’ denotes Comma code-group • ’0’ denotes any other code-group
mgt_rxrundisp[1:0]	Input	Indicates the current running disparity of the incoming code groups in the transceiver.
mgt_rxdisperr[1:0]	Input	Indicates disparity errors in the incoming code groups in the transceiver.
mgt_rxnotintable[1:0]	Input	Indicates incoming code-groups are not recognized code groups.
mgt_rxlossofsync[1:0]	Input	Indicates a loss of synchronization in the transceiver decoder.

Client Interface

The Client Interface signals connect the data path of the core to user logic. In addition to the data signals, the client interface includes signals to frame the output data and to flag errors.Table 2 defines and describes the Client Interface ports.

Table 2: **Client Interface Ports**

Name	Direction	Description
clienttxdata[31:0]	Input	Transmit data from client.
clienttxdatavalid	Input	When High, indicates there is data for transmission as well as indicating clienttxdata word is valid.
clienttxdatareaden	Output	Read Enable for client data. • When ’1,’ clienttxdata, clienttxdatavalid, clienttxsof, and clienttxeof input signals should transition on the rising edge of clk to transfer data to the design. • When ’0,’ all data transitions on these signals should halt.
clienttxsof	Input	Denotes the Start Of Frame word in the frame transfer from the client.
clienttxeof	Input	Denotes the End Of Frame word in the frame transfer from the client.
clientgencrc	Input	Controls CRC generation by the core. • If ’1,’ a CRC is generated for this frame by the core • If ’0,’ the CRC must be supplied by the client This signal is sampled when the input ClientTxSOF signal is asserted.
clienttxframeerror[7:0]	Output	Output error indicator for the transmission data. Each output indicates an error in the process of transmitting a frame. When an error occurs, the output signal pulses for 1 clock cycle.
clientrxdata[31:0]	Output	Receive data output to client.
clientrxdatavalid	Output	’1’ indicates that clientrxdata is a valid data word.
clientrxsof	Output	Denotes the Start Of Frame word in the frame transfer to the client.
clientrxeof	Output	Denotes the End Of Frame word in the frame transfer to the client.
clientrxframeerror	Output	’1’ when the current frame contains an error. Note that this signal can transition High at any point while the frame data is being output. The error type is indicated by the clientrxframerrorstatus output.
clientrxframeerrorstatus[4:0]	Output	Indicates the type of error in the Fibre Channel frame when clientrxframeerror is ’1.’
rrdygenerate	Input	Instructs the core to generate an R_RDY ordered set.

Fast Port Interface

The Fast Port Interface manages the opening and closing of circuits in an arbitrated loop system. The core arbitrates for control of the loop by supplying the address of the destination loop port, then signals back to the user logic when control is obtained. The availability of this Fast Port feature simplifies software development requirements by taking care of these functions for the storage system designer. Table 3 defines and describes the Fast Port Interface ports.

Table 3: **Fast Port Interface**

Name	Direction	Description
fastalporten	Input	When '1,' instructs the core to use the Fast AL Port signals to control AL circuits.
fastalportal_pa[7:0]	Input	Specifies the target AL_PA of the current transmission frame for the fast port.
fastalportcircuit[7:0]	Input	Provides circuit control data for the fast port.
fastalportforceclose	Input	When '1,' instructs the FC-AL to immediately closes any open circuit after the current frame is transmitted.
fastalportretrycount[7:0]	Input	Specifies how many times the fast port should attempt to transmit a frame after initially failing to do so.
fastalportretryaction[1:0]	Input	This input port is valid only when the output fastalporttxerror signal is asserted. This port is used to instruct the fast port on what should be done in the event of a transmission failure (see input port fastalportretrycount).
fastalportcircuitopen	Output	When '1,' indicates an AL circuit, including this port, is currently open. This signal is valid only when the fast port is enabled.
fastalporttxerror	Output	When '1,' indicates the fast port is unable to transmit the frame after attempting to re-transmit the frame for the number of times indicated on the fastalportretrycount input port. When asserted, the user application must instruct the fast port how to handle the current transmission frame as described in the fastalportretryaction input port. This signal is valid only when the fast port is enabled.

Management Interface

The Management Interface is used to configure the core and to perform status monitoring. A simple address-mapped interface reveals the internal state of the core in a processor-friendly format. Table 4 defines and describes the Management Interface ports.

Table 4: **Management Interface Ports**

Name	Direction	Description
address[9:0]	Input	Management address bus
selectn	Input	Management select (active Low)
writen	Input	Management write enable (active Low)
datain[31:0]	Input	Management data input bus
dataout[31:0]	Output	Management data output bus

Alternative Register Access Interface

In addition to the Management Interface, certain configuration registers also have a direct write-port that can be accessed through the Alternative Register Access Interface, enabling more direct manipulation of the configuration registers within a system. Table 5 defines and describes the Alternative Register Access Interface ports.

Table 5: **Alternative Register Access Interface Ports**

Name	Direction	Description
alcircuit_data[13:0]	Input	ALCircuitControl register input
alcircuit_wr_en	Input	Write enable for the ALCircuitControl register
alcircuit_wr_ack	Output	Acknowledge for the ALCircuitControl register
fp_timeout_data[63:0]	Input	Fast Port Timeout register input, providing access to the Fast Port Open Timeout and Fast Port Activity Timeout registers
fp_timeout_wr_en	Input	Write enable for the Fast Port Timeout register
fp_timeout_wr_ack	Output	Acknowledge for the Fast Port Timeout register
lipgentype_data[1:0]	Input	LipGenType register input
lipgentype_wr_en	Input	Write enable for the LipGenType register
lipgentype_wr_ack	Output	Acknowledge for the LipGenType register
freebuffercnt_data[15:0]	Input	Free buffer count register input
freebuffercnt_wr_en	Input	Write enable for the free buffer count register
freebuffercnt_wr_ack	Output	Acknowledge for the free buffer count register

Debug Interface

The Debug Interface provides low-level access to primitive loop operations and can be safely ignored in normal operation of the core. Table 6 defines and describes the Debug Interface ports.

Table 6: **Debug Interface Port Descriptions**

Name	Direction	Description
debugarbloop	Input	Instructs the core to arbitrate for access to the loop. Arbitration won is indicated by the DebugArbWon output signal. When deasserted, the core releases the loop as detailed in the FC-AL-2 specification.
debugsendlip	Input	Instructs the core to immediately send a LIP ordered set. This ordered set overrides any currently transmitted data and can interrupt frame transmission. This signal should be asserted for a minimum of 1 clock cycle. In compliance with the FC-AL-2 specification, 12 LIP ordered sets are generated. If this signal remains asserted after the 12 LIP ordered sets are sent, another 12 are generated.
debugsendcls	Input	Instructs the core to immediately send a CLS ordered set. This Ordered set overrides any currently transmitted data and can interrupt frame transmission. A CLS ordered set is generated for every clock cycle this signal is held High.
debugarbwon	Output	Asserted when the Loop Port State Machine is in the ARBITRATION_WON state

LPSM Control Interface

Certain aspects of the Loop Port State Machine (LPSM) can be controlled by the user application. Table 7 defines and describes the LPSM Control Interface ports.

Table 7: **LPSM Control Interface Port Descriptions**

Name	Direction	Description
lpsm_req_initialize	Input	Instructs the LPSM to enter the INITIALIZE state
lpsm_req_participate	Input	Instructs the LPSM to enter the PARTICIPATE state
lpsm_req_nonparticipate	Input	Instructs the LPSM to enter the NONPARTICIPATE state
req_default_init[1]	Input	Instructs the LPSM to enter the Monitoring state, using the Preferred AL_PA register (register 0x210) as its assigned AL_PA.

[1] This signal does not exist in the FC-AL-2 standard, but does act on the LPSM during INITIALIZING state.

Configuration Status and Statistic Interfaces

The configuration and operational state of the core can be monitored by the configuration/status vector. A statistics vector also provides real-time information core traffic that can be used to gather statistics for system analysis and monitoring. Table 8 defines and describes the Configuration Status and Statistics Interface ports.

Table 8: **Configuration Status and Statistics Interface Port Descriptions**

Name	Direction	Description
configuration_status[168:0]	Output	Presents information about the internal status of the core as a vector.
statistics_vector[45:0]	Output	Certain events in the core are indicated on this port, which can be used to gather statistics on core operation.

Optical Control Interface

The FC-AL core can control a connected SFP optical module using the Optical Control Interface. Table 9 defines and describes the Optical Control Interface ports.

Table 9: **Optical Control Interface Port Descriptions**

Name	Direction	Description
mod_def_1_p	Output	Module Definition 1: Serial Clock.
mod_def_0_p	Input	Module Definition 0: Module is present.
rx_los_p	Input	Loss of Signal.
tx_fault_p	Input	Indicates the optical device has a transmit fault.
tx_disable_p	Output	Disable the transmit laser when ‘1’.
moddef_2_en	Output	Module Definition 2: Serial Data Out, Tri-state enable.
moddef_2_in	Input	Module Definition 2: Serial Data In.
moddef_2_out	Output	Module Definition 2: Serial Data Out.

Other Signals

Table 10 defines and describes ports not defined in previous interface tables.

Table 10: **Other Port Descriptions**

Name	Direction	Description
clk	Input	Main core clock 1 Gbps/2 Gbps cores: 53.125 MHz 2 Gbps/4 Gbps cores: 106.25 MHz
core_reset	Input	Main core reset
enable_1g	Output	When the core is operating at the lower of the two speeds, this output toggles every clock cycle to signify the internal state of the clock-enable circuit.

Table 10: **Other Port Descriptions *(Continued)***

Name	Direction	Description
txusrclk	Input	Double-rate transmit-side clock 1 Gbps/2 Gbps: 106.25 MHz 2 Gbps/4 Gbps: 212.5 MHz
rxrecclk	Input	Receive recovered clock from the RocketIO transceiver
rec_reset	Input	Reset for the clock domain driven by rxreclk

Example Design

An example design that instantiates the core in a chip-level wrapper is provided with the FC-AL core (Figure 5) in both VHDL and Verilog. The example design includes a LocalLink-compliant FIFO and statistics-gathering module. The design also includes a demonstration test bench that exercises the core and illustrates some of the interfacing techniques.

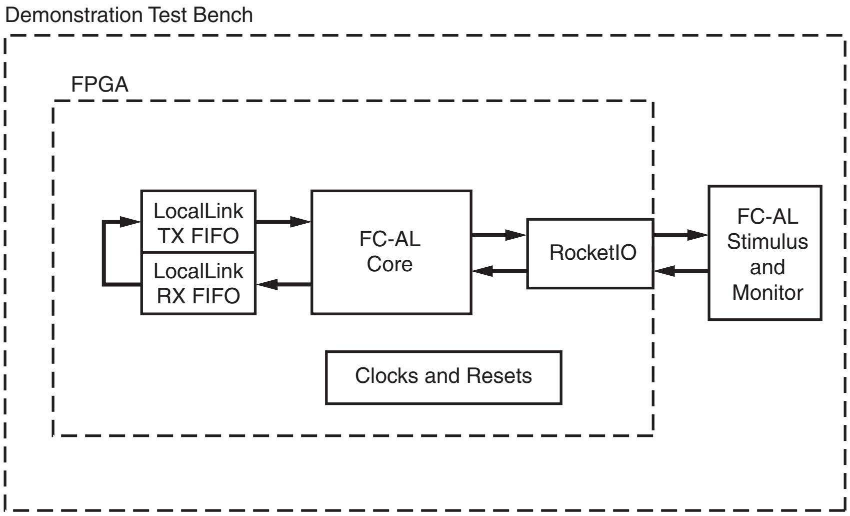

Figure 5: **FC-AL IP Core Example Design**

Support

Xilinx provides technical support for this LogiCORE IP product when used as described in the product documentation. Xilinx cannot guarantee timing, functionality, or support of product if implemented in devices that are not defined in the documentation, if customized beyond that allowed in the product documentation, or if changes are made to any section of the design labeled *DO NOT MODIFY.*

Ordering Information

The FC-AL core, part of the LogiCORE IP Fibre Channel solution, is provided under the SignOnce IP Site License and can be generated using the Xilinx CORE Generator™ v10.1 or higher. The CORE Generator software is shipped with Xilinx ISE Foundation Series Development software.

A simulation evaluation license for the core is shipped with the CORE Generator. To access the full functionality of the core, including FPGA bitstream generation, a full license must be obtained from Xilinx.

Please contact your local Xilinx sales representative for pricing and availability of additional Xilinx LogiCORE IP modules and software. Information about additional Xilinx LogiCORE IP modules is available on the Xilinx IP Center.

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
1/11/06	1.0	Initial Xilinx release.
2/15/07	2.0	Updated core to v2.1; Xilinx tools 9.1i. Added req_default_init pin to pinout and interface description; changed fastalportforce to fastalportforceclose.
8/8/07	2.1	Updated core to v2.2; Xilinx tools 9.2i.
3/24/08	2.5	Updated core to v2.3; Xilinx tools 10.1.

Disclaimer

Xilinx is providing this information (collectively, the "Information") to you "AS IS" with no warranty of any kind, express or implied. Xilinx makes no representation that the Information, or any particular implementation thereof, is free from any claims of infringement. You are responsible for obtaining any rights you may require for any implementation based on the Information. All specifications are subject to change without notice. XILINX EXPRESSLY DISCLAIMS ANY WARRANTY WHATSOEVER WITH RESPECT TO THE ADEQUACY OF THE INFORMATION OR ANY IMPLEMENTATION BASED THEREON, INCLUDING BUT NOT LIMITED TO ANY WARRANTIES OR REPRESENTATIONS THAT THIS IMPLEMENTATION IS FREE FROM CLAIMS OF INFRINGEMENT AND ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Except as stated herein, none of the Information may be copied, reproduced, distributed, republished, downloaded, displayed, posted, or transmitted in any form or by any means including, but not limited to, electronic, mechanical, photocopying, recording, or otherwise, without the prior written consent of Xilinx.