XAPP648 (v1.1) November 30, 2004

Serial Backplane Interface to a Shared Memory

Author: Steve Trynosky

Summary

This application note utilizes the Virtex-II Pro™ RocketIO™ transceivers and the Xilinx Aurora protocol engine to provide a multi-ported interface to a shared memory system in a backplane environment. Multiprocessor systems are often encountered in backplane systems, and distributed processing applications require access to a shared memory across a backplane bus. Utilization of a hardware test-and-set lock mechanism, along with a software protocol to test for a semaphore grant prior to accessing the shared memory, guarantees atomic access to the shared memory.

The design described in this application note can be used in any multiprocessor backplane application that requires atomic access to a shared memory system. Memory system performance is secondary to the atomic access requirement. Target applications include:

- Message passing between multiple embedded processor line cards
- Sequential processing of data by embedded processor nodes
- Shared memory storage for data path bridges in protocol converter applications
- Disk controller cache memory allocation tables
- RAID controller logical-to-physical device mapping tables
- Distributed processing for front-end and back-end storage controller interfaces

Introduction

Historically, multiprocessor systems have used parallel buses in backplane applications. The bus structure can be an industry standard, such as PCI or PCI-X, or a proprietary interface. A typical parallel bus system is shown in Figure 1. Bus based systems require interface signals for addressing, data, control, and clock to transfer data between a host bus adapter (HBA) and the shared memory adapter (SMA). As performance requirements increase, physical limitations of the bus preclude going faster, so the bus must become wider in order to double performance. At some point, usually 64 bits, the bus structure becomes very costly in terms of number of I/O pins required, printed circuit board real estate for special drivers and receivers to interface to the back-plane structure, and limits for maximum number of adapters on the bus.

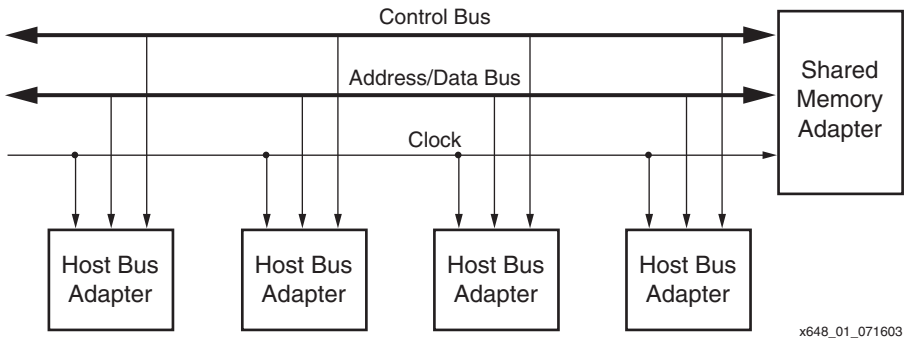


Figure 1: **Shared Memory System Using Parallel Bus Interface**

In fault-tolerant applications, a secondary path from each HBA to a redundant shared memory adapter is required. This path doubles the parallel bus I/O count requirements, physical size of the backplane connector system, and the HBA.

Virtex-II Pro Serial Backplane Solution

Utilizing the RocketIO transceivers in the Virtex-II Pro FPGA, a serial backplane reference system is constructed as shown in Figure 2. The parallel bus adapters of Figure 1 are replaced with serial host bus adapters (SHBA) that attach to a serial backplane. The shared memory is implemented on a serial shared memory adapter (SSMA). Each SHBA has a point-to-point to serial link to the SSMA constructed using the Xilinx Aurora protocol engine. The RocketIO transceivers utilize differential serial transmit (TX) and receive (RX) connections between each bus adapter and the shared memory, drastically reducing the I/O count on the backplane.

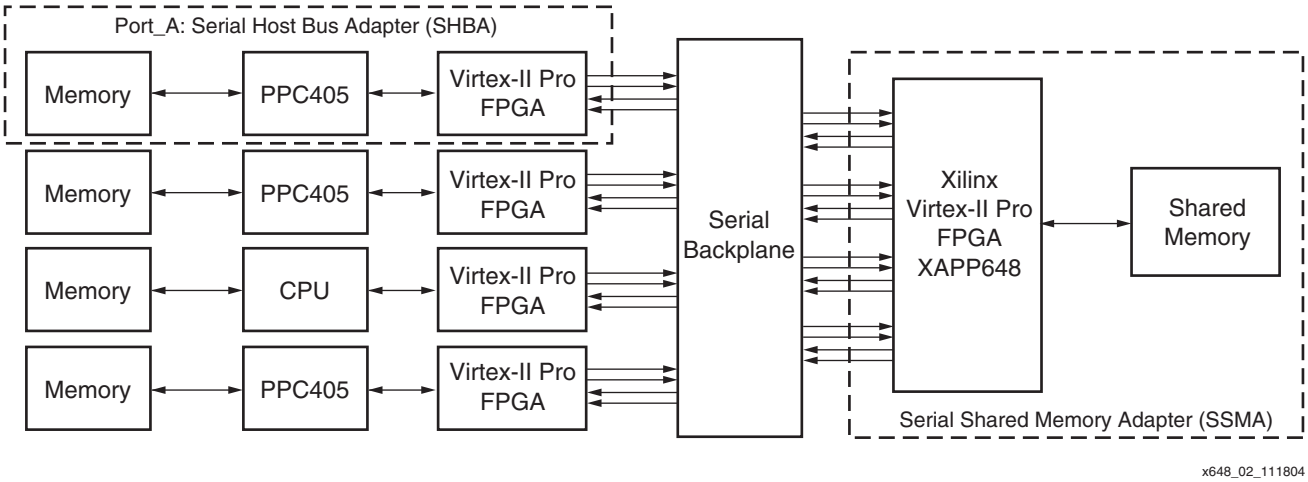


Figure 2: **Shared Memory System Block Diagram**

The architecture supports both homogeneous and heterogeneous SHBA platforms as shown in Figure 2. The focus of this design is the shared memory side of the interface.

Shared Memory and Semaphore Memory

Figure 3 shows the relationship between the semaphore memory and the shared memory. The physical capacity of semaphore memory is significantly smaller than the shared memory. It provides only a means to lock regions of the shared memory. External to the Virtex-II Pro FPGA is a 512K x 16-bit Cypress Semiconductor Micron SyncBurst SRAM memory used as shared memory for the SHBA devices. Virtex-II Pro FPGA block SelectRAM stores hardware locks, or semaphores, for the shared memory. The amount of shared memory reserved by the hardware lock is defined by application software.

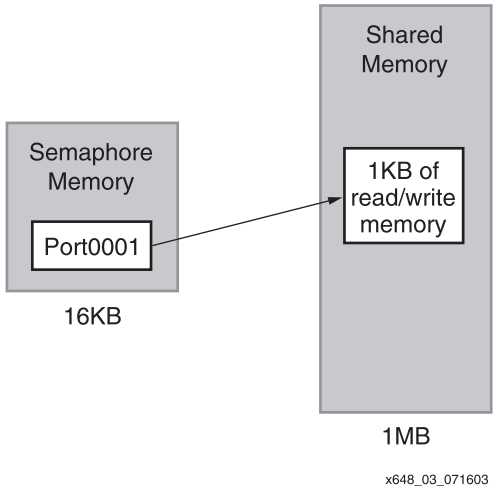


Figure 3: **Semaphore Memory and Shared Memory**

To guarantee atomic access to the shared memory, all SHBA devices must follow these four simple rules for accessing the shared memory:

1. Arbitrate for a semaphore or lock. If the semaphore is locked by another SHBA, retry until granted access. Do not access the shared memory block until the lock has been issued.
2. After the semaphore or lock is granted, read and write accesses are granted solely to this adapter. Other SHBA devices can access other shared memory locations, provided they have a valid lock for those memory locations.
3. All SHBA devices must use a common definition for the lock regarding shared memory block size. The block size granted with each semaphore is a linear mapping, a non-linear mapping, or a combination of linear and non-linear mapping of memory elements. The block size is defined uniquely by the software application. For linear mapping, the size of the shared memory block is simply the shared memory range divided by the semaphore memory range. In this reference design, a linear mapping means that each semaphore location maps to 512K/16K or 32 halfwords of memory. A halfword contains 16 bits.
4. When finished accessing the reserved shared memory locations, the SHBA must UNLOCK the semaphore memory location so that other SHBA devices can access that memory block.

Altering the depth of the semaphore memory permits the application software to allocate or reserve different size blocks of memory in the shared memory. Table 1 shows the capabilities of this reference design, assuming a linear mapping. Application software can choose to map the semaphore memory and shared memory in other configurations as well.

Table 1: **Semaphore Memory Depth versus Shared Memory Locked Block Size**

Semaphore Memory Organization	RAMB16 Elements Required	Shared Memory Locked Block Size (bytes)
16 KB	**8**	**64**
8 KB	4	128
4 KB	2	256
2 KB	1	512

Notes:

1. The reference design implements 16KB memory.

Each 8-bit semaphore memory location in the test and set lock area reserves a block of 16-bit memory locations in the shared memory. There are 512K lockable memory elements. The semaphore data bit definitions are described in Table 2.

Table 2: **Semaphore Memory Data Bit Definitions**

Semaphore Data Bit	Name	Description
7:4	PORT_ID	This field identifies which one of "N" SHBA devices has locked a particular semaphore memory location. Each backplane card slot contains a four-bit hardwired identifier used to uniquely identify each physical position. The PORT_ID information is included in the LOCK REQUEST message packet to permit the controller to write this information into a memory location if the adapter receives the LOCK.
3:1	Reserved	The hardware forces these bits to 0s on read-modify-write cycles. During an UNLOCK bus transaction, all eight bits can be written by a bus master.
0	LOCK	When LOCK is set, the SHBA has locked this region of shared memory. No other adapter can access this region while the lock is asserted. When LOCK is cleared, the SHBA cannot read or write to the shared memory. The SHBA that previously has locked this location of memory is identified by PORT_ID. **Note:** The hardware performs a read-modify-write to this memory location to write both the LOCK and the PORT_ID bits to the test and set memory locations. The hardware returns a LOCK status bit only when an SHBA has locked the memory location. Subsequent lock requests will return a status of 0. When the SHBA has completed accessing the shared memory locations, it must write to the locked memory location (data = 0x0) to remove the lock. An UNLOCK request must be issued to the SSMA.

Interlocked Handshake

Flow control is accomplished by utilizing an interlocked handshake between the SHBA and the SSMA, as shown in Figure 4. Each control request from the SHBA is followed by a status response from the SSMA. After the status packet is received by the SHBA, the next packet can be transmitted.

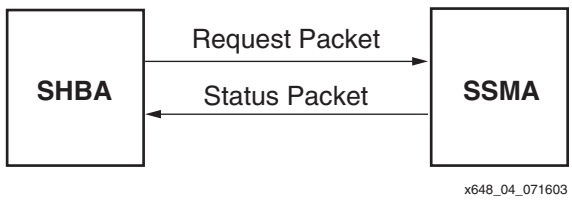


Figure 4: **Interlocked Handshake**

The Xilinx Aurora protocol is used to communicate chip-to-chip across a backplane. A FIFO is placed on receive and transmit data paths of the interface. The receive FIFO holds the data packet while the CRC is being checked. In cases where the memory is being accessed by another serial HBA, the request is queued up for the arbiter to resolve priorities.

Packets received with CRC errors do not generate memory accesses. A status packet is returned to the SHBA, indicating packet CRC error. The SHBA must retransmit the control request packet.

The SHBA sends four unique packet types to the SSMA:

1. Request to lock semaphore memory segments
2. Request to unlock semaphore memory segments
3. Read shared memory (block transfer) request
4. Write shared memory (block transfer) request

The SSMA responds with five unique status packets to the SHBA:

1. Status of lock semaphore request (lock issued or lock busy)
2. Status of unlock semaphore request (operation complete)
3. Status of read shared memory (return data block)
4. Status of write shared memory (transfer complete)
5. Received packet with CRC error (retransmit packet)

Refer to Figure 14 through Figure 22 for examples of packets exchanged between an SHBA and the SSMA.

Control and Status Message Packet Definition

Figure 5 shows the basic packet structure. Each packet contains an ADDRESS field, CONTROL or STATUS field, a DATA or PAD field, and a CRC field. The SHBA transmits packets with CONTROL fields, and the SSMA adapter transmits STATUS packets. Each field is a 32-bit word. The Aurora protocol engine contains a 16-bit interface, so the words are disassembled into two 16-bit fields for transmission across the channel.

32	32	32	32	32	32
ADDRESS	CONTROL or STATUS	DATA or PAD	DATA or PAD	DATA or PAD	CRC

Figure 5: **Packet Definition**

The minimum packet size is 24 bytes, or 12 halfwords, including the CRC field. The minimum packet size for the RocketIO transceiver with CRC enabled is 24 bytes. The maximum packet size is limited only by the size of the FIFO placed in the Aurora data path. Individual packet fields are further defined in Table 3.

Table 3: **Packet Fields and Packet Field Bit Definitions**

Field	Name	Bits	Description
ADDRESS	ADDRESS	31:0	Memory address to access. For shared memory accesses, only the least significant 19 bits are used. For semaphore memory accesses, only the least significant 14 bits are used. Reserved or unused bits should be 0s.
CONTROL	Reserved	31:17	Reserved or unused bits should be 0s.
	XFR_COUNT	16:8	Shared memory word transfer count.
	PORT_ID	7:4	Identifies which SHBA is requesting access. For semaphore accesses, this information is written into the memory if a lock is granted.
	BUSREQUEST	3	Set to indicate the SHBA is requesting access to shared memory. Access type is further qualified by RnW, bit 2.
	RnW	2	Set to read shared memory and cleared to write to shared memory. Must be qualified with BUSREQUEST, bit 3.
	UNLOCK	1	Set to indicate request to write to semaphore memory.
	LOCK	0	Set to indicate request to lock semaphore memory location prior to accessing shared memory.
DATA	DATA	31:0	Memory data for read/write accesses. For shared memory accesses, all 32 bits are used (the most-significant 16 bits are transferred first). For semaphore memory accesses, only the least-significant eight bits are used. Reserved or unused bits should be 0s.
PAD	PAD	31:0	Padding word used to meet the minimum, six-word packet with CRC enabled for the Virtex-II Pro RocketIO transceiver.
STATUS	Reserved	31:17	Reserved or unused bits should be 0s.
	XFR_COUNT	16:8	Shared memory halfword transfer count.
	Reserved	7:6	Reserved or unused bits should be 0s.
	CRC_ERROR	5	Set to indicate CRC Error detected on the receiver data path. The SHBA must retransmit request packet.
	COMPLETE	4	Set to indicate the SSMA has completed the memory request. Evaluate packet contents for status of operation.
	BUSREQUEST	3	Set to indicate the SHBA is requesting access to shared memory. Access type is further qualified by RnW, bit 2.
	RnW	2	Set to read shared memory and cleared to write to shared memory. Must be qualified with BUSREQUEST, bit 3.
	UNLOCK	1	Set to indicate request to write to semaphore memory.
	LOCK	0	Set to indicate request to lock semaphore memory location prior to accessing shared memory.
CRC	CRC	31:0	Cyclic Redundancy Check (CRC) field used by the RocketIO transceiver. The transceiver calculates the packet CRC and inserts the generated CRC in a transmitted packet. On the receiver side of the interface, the transceiver calculates the CRC on the incoming packet and compares it against the CRC inserted on the transmit side of the interface. If the two values do not match, there is a CRC error in the received packet. At the Aurora interface, the user logic must include a 32-bit CRC field that permits the transceiver to modify the data flow to include the generated CRC. On the receiver end, the CRC value is discarded.

Design Elements

Figure 6 shows a block diagram of the reference design. The design elements are described in the following subsections.

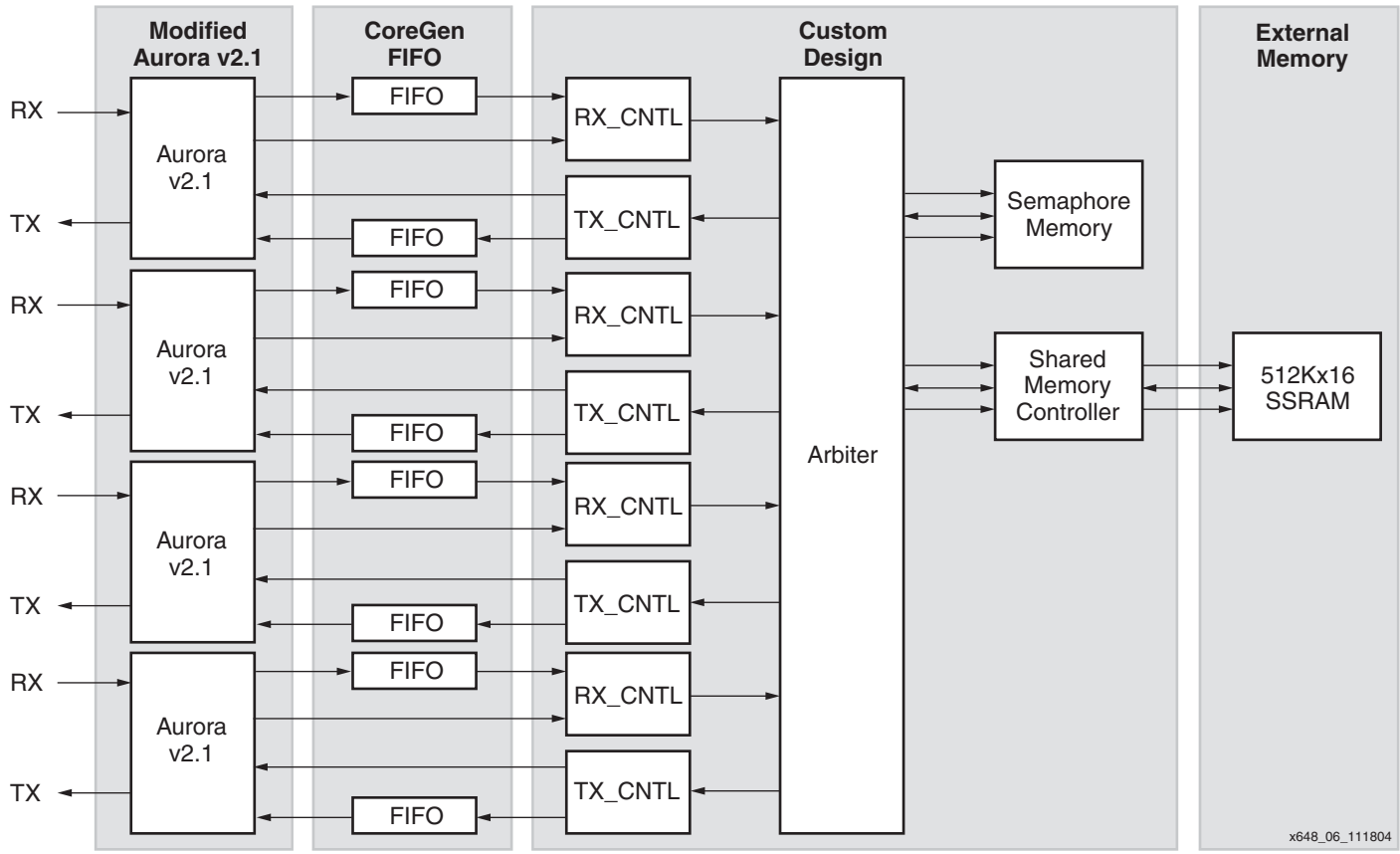


Figure 6: **Reference Design Block Diagram**

Aurora Protocol Engine

The Aurora protocol engine is an open-source core available from the Xilinx web site. The protocol engine is a serial backplane standard that is tailored for maximum efficiency and ease of use in high-performance, point-to-point data transmission systems. The protocol engine includes the Virtex-II Pro RocketIO Multi-Gigabit Transceiver (MGT). The protocol engine provides a LocalLink user interface that is used to encapsulate data packets. For additional information on the Aurora Protocol specification or the Aurora reference design, visit the Xilinx http://www.xilinx.com/products/design_resources/conn_central/grouping/aurora.htm web site. This reference design utilizes four Aurora version 2.1 16-bit elements, one for each SSMA to SHBA interface.

The Aurora protocol engine is not provided with this reference design. To generate these required files, the Xilinx CORE Generator software must be used. After obtaining a license for the CoreGen software, follow the steps shown in Figure 7 through Figure 11, which explain in detail the options to select when generating the Aurora core.

Generating the Aurora Module

First, visit http://www.xilinx.com/products/design_resources/conn_central/grouping/aurora.htm to register for the Aurora Solutions Suite and install the Aurora license. After doing so, start the CORE Generator application in stand-alone mode (see Figure 7), and create a new project for XAPP648. Choose the project directory, design entry language (Verilog or VHDL), `Virtex2P` as the target architecture, and `B[n:m]` as the netlist bus format, then click OK.

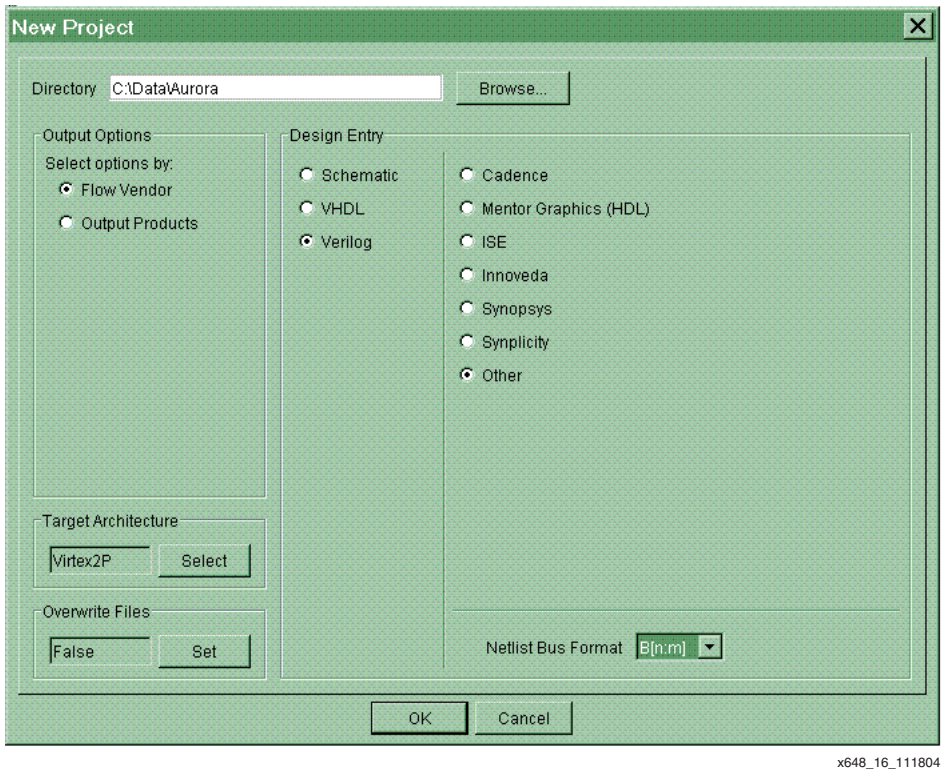
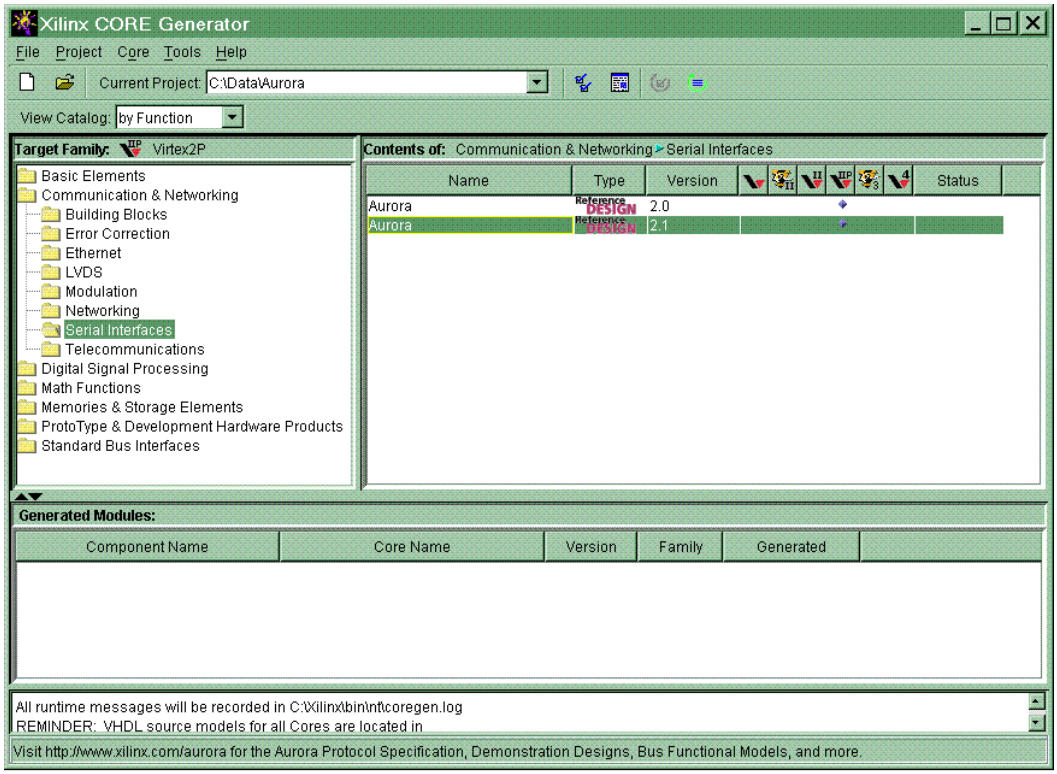


Figure 7: **Step 1 in Generating the Aurora Module**

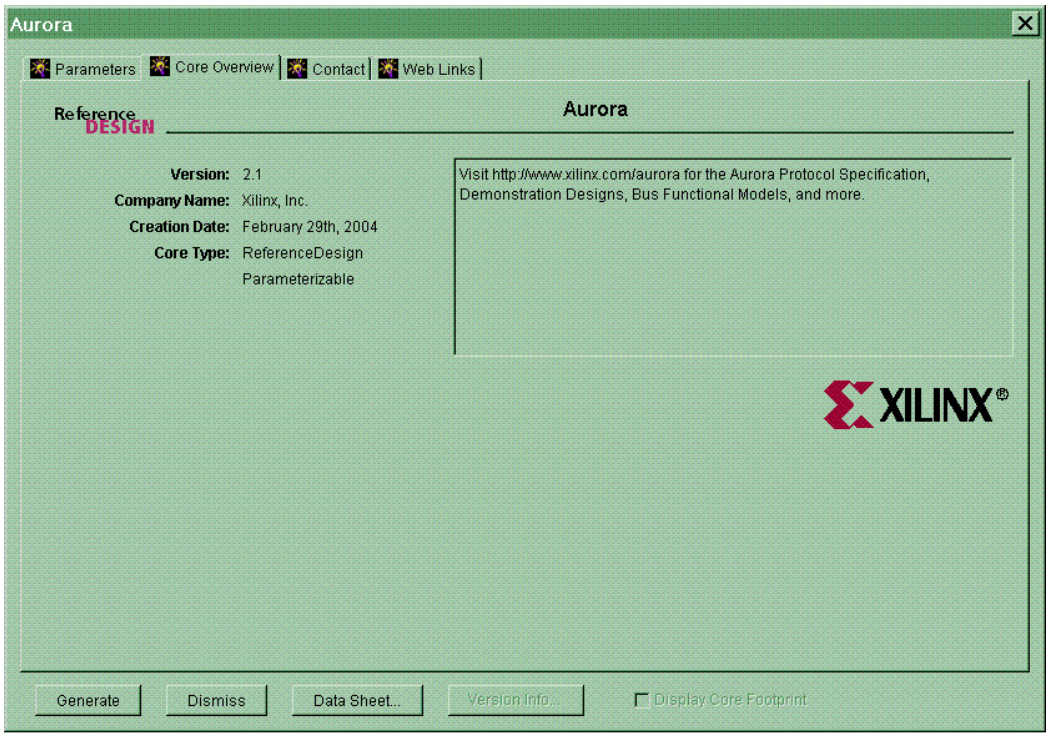
Select the Aurora Reference Design V2.1 in the Serial Interfaces Section of the Communications & Networking Catalog, as shown in Figure 8.



x648_17_111804

Figure 8: **Step 2 in Generating the Aurora Module**

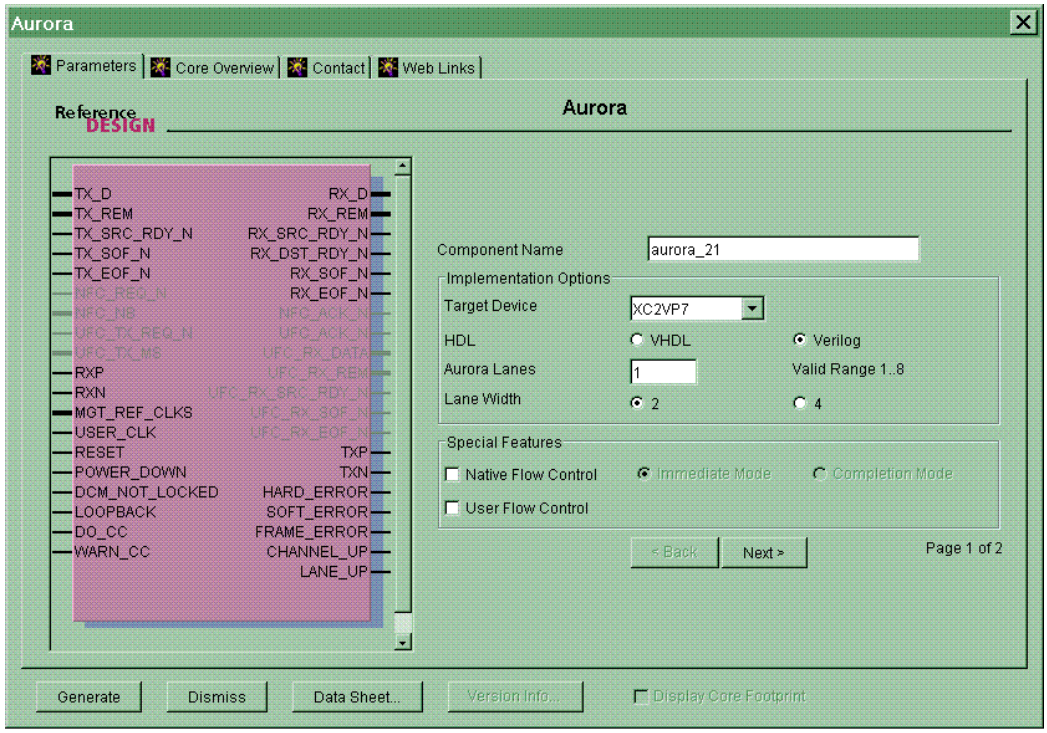
As shown in Figure 9, select the Core Overview tab for additional information on the Aurora Reference Design. Select the Parameters tab to display the steps for generating the required protocol engine design files.



x648_18_111804

Figure 9: **Step 3 in Generating the Aurora Module**

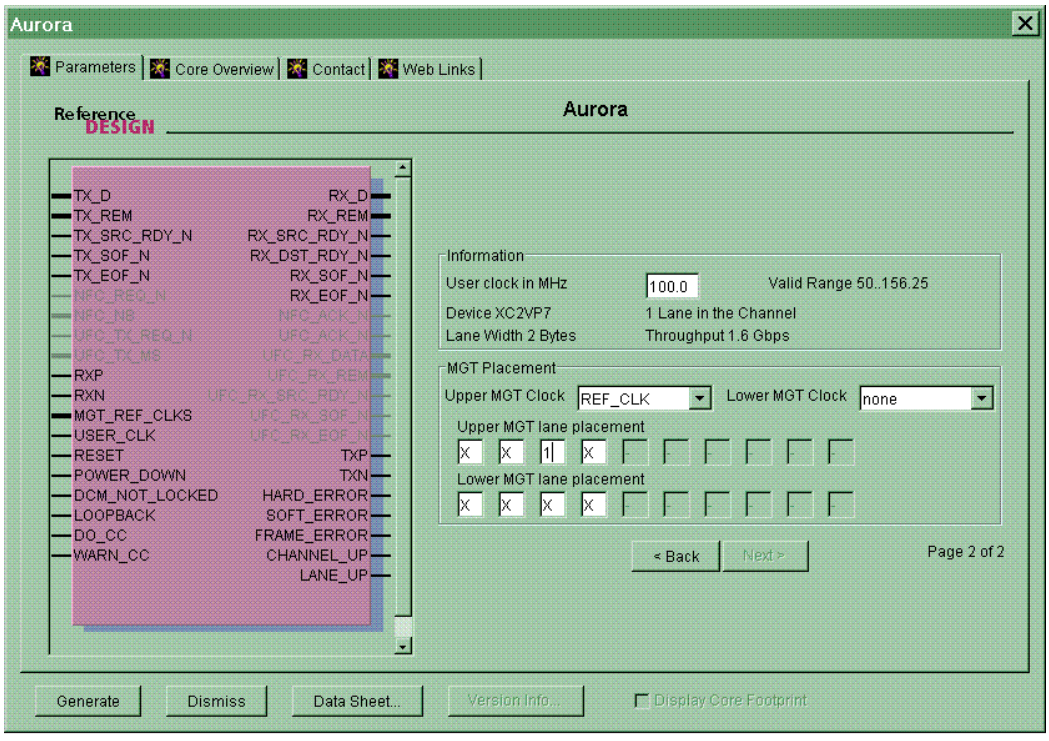
Name the component aurora_21, select target device XC2VP7, HDL language (Verilog or VHDL), 1 lane, and a lane width of 2, as shown in Figure 10. Deselect Native Flow Control and User Flow Control, as this design does not use either method. Click Next.



x648_19_111804

Figure 10: **Step 4 in Generating the Aurora Module**

As shown in Figure 11, enter 100.0 MHz for user clock speed, select REF_CLK for Upper MGT Clock, and select the third lane for Upper MGT Lane Placement (by entering 1 in the third box). Click Generate.



x648_20_111804

Figure 11: **Step 5 in Generating the Aurora Module**

Enabling Serial Channel CRC

By default, the Aurora protocol engine disables the CRC feature of the RocketIO transceiver. Enabling the embedded CRC function within the transceiver is accomplished by editing the downloaded Aurora Verilog or VHDL design files. For functional or timing simulation, the GT_CUSTOM attributes in the Aurora design file must be altered or added as shown in Table 4. The reference design includes attribute modifications for the implementation tool constraint file.

Note: The customer must register for the Aurora protocol engine on the Xilinx web site, download the Aurora license, and run CoreGen to generate the Verilog or VHDL design files to complete this reference design. Instructions for editing the design files to enable CRC are described in the README file contained in `XAPP648.zip`.

Table 4: **Enabling RocketIO CRC Generation and Checking**

Attribute	Value
RX_CRC_USE	TRUE
TX_CRC_USE	TRUE
CRC_FORMAT	USER_MODE
CRC_START_OF_PKT	K28_2
CRC_END_OF_PKT	K29_7

The following ports need to be brought up to the top design level from the Aurora module and then connected in the top-level design module:

- Inputs:

  TXFORCECRCERROR
- Outputs:

  RXCHECKINGCRC

  RXCRCERROR

When CRC is enabled, the RocketIO transceiver transmit data path must receive the entire data packet, from SOF to EOF, without any interruption. To guarantee a complete data packet transmission to the Aurora interface, two outputs DISABLE_TX and DISABLE_TX_BIG, are generated in the CC_COUNT module of the design. Both outputs are asserted ahead of the clock correction sequence with enough guardband to permit one last transmitted packet, including CRC, before clock correction occurs. These two signals are used in the TXFIFO_CNTL module.

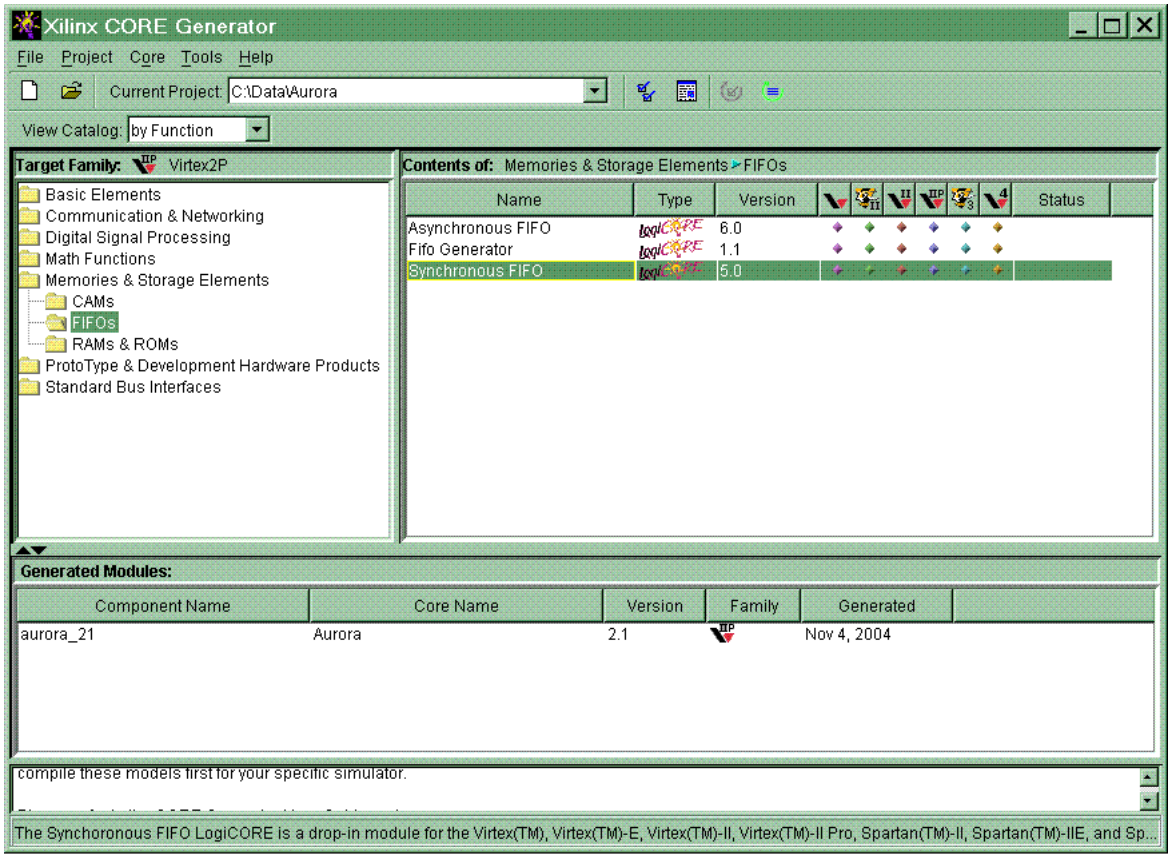
The DISABLE_TX output is monitored prior to transmitting small control and status packets, while the DISABLE_TX_BIG output is monitored prior to transmitting large control or status packets. Small control and status packets are defined as LOCK request, LOCK status, UNLOCK request, UNLOCK status, Shared Memory Read request, and Shared Memory Write status. Large control and status packets are defined as Shared Memory Write request and Shared Memory Read status.

CoreGen FIFO

A synchronous 1Kx16 bit FIFO is placed between transmit and receive data paths of the Aurora protocol engine and the arbiter module. The FIFO permits the received data to be completely received before initiating a request to the shared memory or semaphore memory, which facilitates error checking of the received frame prior to initiating a memory transfer. The CRC word is not stored in the FIFO. The receive data path FIFO (RXFIFO) is controlled by the RXFIFO_CNTL design module, and the transmitter data path FIFO (TXFIFO) is controlled by the TXFIFO_CNTL design module.

The data transfer size for shared memory access is limited by the size of the FIFO. The FIFO stores the address, control/status, and data information for the transmit/receive data path. When an SHBA adapter has been granted the LOCK for a shared memory segment, multiple data packets can be exchanged between the SHBA and SSMA devices.

The RXFIFO is generated by using the Xilinx CoreGen software. As shown in Figure 12, select Synchronous FIFO from the FIFOs group under Memories and Storage Elements.



x648_21_111804

Figure 12: **Step 1 in Generating the RXFIFO Module**

As shown in Figure 13, name the component, select Block Memory as the memory type, a data width of 16 bits, and a FIFO depth of 1024, then click Generate.

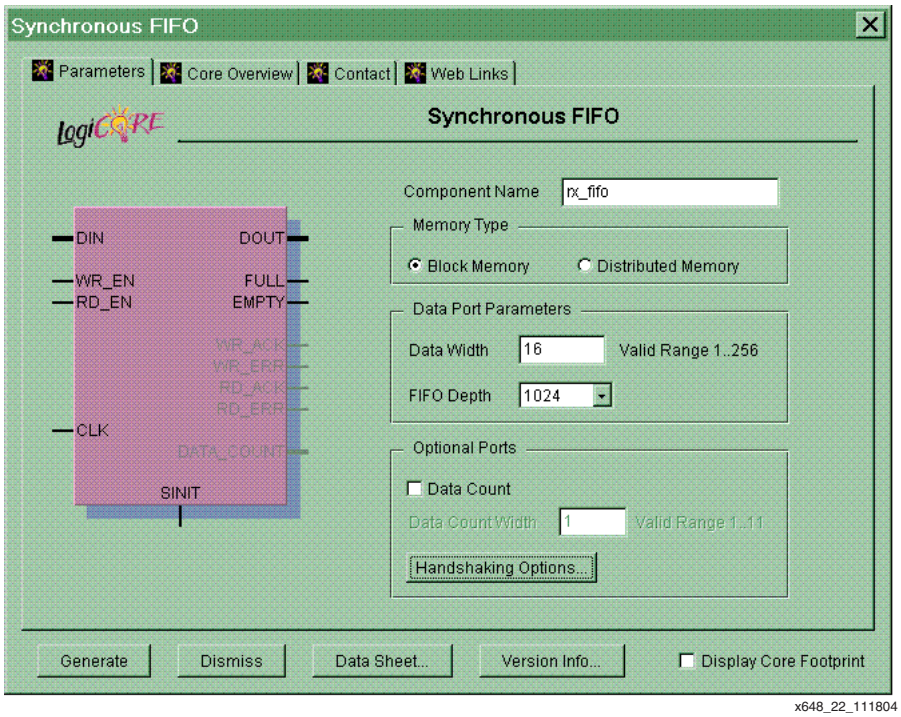


Figure 13: **Step 2 in Generating the RXFIFO Module**

RXFIFO_CNTL

The RXFIFO_CNTL module provides the interface between the Aurora receive data path and the arbiter. A state machine provides the LocalLink interface to the Aurora protocol engine. This module disassembles the received packet and provides address, control, and data to the arbiter module. The control information in the data request packet is stored for the duration of the memory cycle.

In the event of a CRC error, the hardware inhibits memory accesses. The RXFIFO is cleared, and the TXFIFO_CNTL module returns a status packet with the CRC_ERROR bit set to 1.

TXFIFO_CNTL

The TXFIFO_CNTL module controls the interface between the memory arbiter and the TXFIFO transmit data path of the SSMA. A state machine provides the LocalLink interface to the Aurora protocol engine. This module creates the status packets in response to request packets received in the RXFIFO_CNTL module. Data returned from either the semaphore controller or the shared memory controller is routed to the TXFIFO and then to the transmitter data path of the Aurora protocol engine.

The state machine monitors the state of the DISABLE_TX and DISABLE_TX_BIG outputs. If either output is asserted, the status packet is transmitted after the RocketIO clock correction sequence. If one of the output signals is asserted after the start of frame is asserted, then there is sufficient guardband to enable a complete packet transmission before the mandatory clock correction sequence is performed in the RocketIO transceiver.

If a received control packet contains a CRC error, the TXFIFO_CNTL module waits for the RXFIFO to empty and then transmits a status packet back to the SHBA. Data received is invalid, so the hardware creates a packet with the following information to indicate a CRC error:

Address field = BAAD_AAAAh

Status field = 0000_0030h

Data fields = BAAD_BAADh

XAPP648_CC_MODULE

The RocketIO transceiver has a reference clock frequency tolerance of +/- 100 parts per million (ppm). In backplane applications where the SHBA and SSMA devices each contain an oscillator for the REFCLK, clock correction is required every 5000 clock cycles in order to compensate for small frequency variations between the two REFCLKs.

The XAPP648_CC_MODULE provided with this reference design interfaces directly with the Aurora protocol engine to request a clock correction sequence every 5000 clock cycles. A 13-bit counter counts up to 5000 (decimal) and then resets. A clock correction sequence request signal, NEED_CC, is the output from this module. This signal is connected to the DO_CC input of the Aurora protocol engine. Two additional outputs of this module are used to warn the TXFIFO_CNTL module that a clock correction sequence is about to begin and that transmission should be disabled during that time. The DISABLE_TX signal is used for small status packets, and the DISABLE_TX_BIG signal is used for large memory read status packets to the SHBA.

Arbiter

The arbiter module services requests to access both the semaphore and the shared memory from four Aurora interface channels. The arbiter implements a fairness algorithm to permit each channel equal access to the memory facilities. For example, if all four SHBA devices request access to the shared memory at the same time, the arbiter grants the memory to Port_A, Port_B, Port_C, and finally Port_D. After Port_A has finished, if Port_A again requests the memory during any time that Port_B, Port_C, or Port_D has an outstanding request, Port_A must wait until the other devices have been granted access to the shared memory facility. The arbiter for the semaphore memory resource operates in the same manner as the shared memory arbiter.

The arbiter grants the memory resource for the duration of a data transfer, beginning with the initial received request for memory access, during the memory access cycle, and terminating upon completion of the data transmittal to the Aurora protocol engine. When the TXFIFO_CNTL module asserts the TX_DONE signal, the arbiter grants the bus to the next master.

The arbiter module contains two separate arbiters: one for the shared memory and one for the semaphore memory. Shared memory accesses generally take longer than semaphore accesses, so a degree of parallelism is provided. Note that in the parallel bus implementation of a shared memory system shown in Figure 1, parallelism cannot be designed into the system. A single bus interface limits parallel data transmission. In this serial reference system, packets can be transmitted on all four Aurora channels, and both the semaphore and shared memory can operate simultaneously.

Semaphore

The semaphore module services requests for locking or unlocking a location in the eight-bit wide block RAM (BRAM). The user can modify the instantiations of BRAM memory elements to support locking larger or smaller blocks of shared memory. The reference design provides 16KB of semaphore memory.

For the LOCK control request function, the hardware returns RD_DATA to the SHBA. From the host processor's perspective, if the least-significant bit (LSB) is 0, another host bus adapter (HBA) has locked this memory location. If the LSB is 1, the lock has been granted to this HBA. The hardware performs read-modify-write cycles in order to lock the memory and write the PORT_ID into the BRAM location identified by LOCK_ADDRESS.

For the UNLOCK control request function, the hardware writes the value of WR_DATA into the LOCK_ADDRESS location. The UNLOCK function is used for semaphore memory initialization and to remove a hardware lock when the SHBA has completed read/write access to the shared memory block. In the event of an SHBA failure, another SHBA also can remove the LOCK from a memory location to free up the shared memory resource.

All locking and unlocking requests are under the control of the distributed processing elements contained on the serial host bus adapters. The hardware does not check that a specific SHBA has a valid lock before initiating reads or writes to the shared memory. This method only works if the hardware is backed up by a software protocol to guarantee atomic access to the memory.

Shared Memory

The shared memory module provides an interface to the external 512Kx16-bit SSRAM memory. All memory accesses are 16 bits wide. Byte writes are not supported. The design uses the address advance feature of the SSRAM memory, therefore read operations must begin on a quad-word-aligned address boundary. The two least-significant address bits are always 00b for reads. LVCMOS25 I/O buffers and data bus pull-up resistors are provided in this module.

The maximum transfer count per data packet is limited by the size of the RXFIFO or TXFIFO. A nine-bit transfer count field is provided in the COMMAND or STATUS field of the packet. The address, control, and data must all fit in the 512 deep FIFO.

A DCM generates a deskewed board clock for the SSRAM memory. The printed circuit board SSRAM_CLKFB is input to this DCM to provide the board clock feedback. The shared memory runs at the same speed, 100 MHz, as the 16-bit Aurora protocol engine interface.

Clocking

The external memory clock is limited to 100 MHz. The maximum baud rate of the RocketIO transceiver is speed-grade dependent. The RocketIO transceiver can be clocked with either BREFCLK or REFCLK. This reference design uses REFCLK for the RocketIO transceiver clock. This reference design is targeted for a Virtex-II Pro FPGA with a –5 speed grade, which in turn limits the baud rate on the serial channel to 2 Gb/s.

An external 100 MHz LVDS differential clock source provides the centralized clocking of the design.

Serial Shared Memory Adapter Ports

The inputs and outputs of the FPGA are defined in Table 5, Table 6, and Table 7. The design contains three unique interfaces:

- RocketIO transceiver serial I/O (Table 5)
- External clocks and reset (Table 6)
- Shared memory SyncBurst SSRAM (Table 7)

The SSRAM device used in this design is an MT58L512V18F-7.5 or equivalent. For more information on this memory device, refer to the Cypress Semiconductor web site at http://www.cypress.com/products/micron/micron.cfm.

Note: In the three tables below, "_N" in the signal name indicates the signal is active Low.

Table 5: **Serial I/O Signal Descriptions**

Signal Name	I/O	Description
A_TXP, A_TXN	O	Differential transmitter outputs to the SHBA Port_A receiver inputs.
A_RXP, A_RXN	I	Differential receiver inputs from the SHBA Port_A transmitter outputs.
B_TXP, B_TXN	O	Differential transmitter outputs to the SHBA Port_B receiver inputs.
B_RXP, B_RXN	I	Differential receiver inputs from the SHBA Port_B transmitter outputs.
C_TXP, C_TXN	O	Differential transmitter outputs to the SHBA Port_C receiver inputs.
C_RXP, C_RXN	I	Differential receiver inputs from the SHBA Port_C transmitter outputs.
D_TXP, D_TXN	O	Differential transmitter outputs to the SHBA Port_D receiver inputs.
D_RXP, D_RXN	I	Differential receiver inputs from the SHBA Port_D transmitter's outputs.

Table 6: **Clock and Reset Signal Descriptions**

Signal Name	I/O	Description
SYSCLK_P	I	Positive edge-triggered LVDS input clock for RocketIO transceiver (REFCLK)
SYSCLK_N	I	Negative edge-triggered LVDS input clock for RocketIO transceiver (REFCLK)
PB_RESET_N	I	Power monitor and push-button reset input
SSRAM_CLKFB	I	Shared memory board clock input. The clock is deskewed by a DCM.
SSRAM_CLK	O	Shared memory SSRAM clock output

Table 7: **SSRAM Shared Memory Signal Descriptions**

Signal Name	I/O	Description
SSRAM_SA[18:0]	O	Synchronous address bus
SDQ[15:0]	I/O	Bidirectional shared memory data bus
SSRAM_N_CE	O	Synchronous chip enable
SSRAM_N_ADV	O	Synchronous address advance
SSRAM_N_ADSC	O	Synchronous address status controller select
SSRAM_N_ADSP	O	Synchronous address status processor select
SSRAM_N_OE	O	Output enable
SSRAM_N_WE[2:0]	O	SSRAM write enables. SSRAM_N_WE[2] is asserted for all write cycles. SSRAM_N_WE[1] writes SDQ[15:8]. SSRAM_N_WE[0] writes SDQ[7:0]

Design Implementation

Synthesis

The reference design is synthesized using the Xilinx XST synthesis tool, version 6.3.02i.

Simulation

The following simulation data is provided:

- Functional and timing simulation directories
- Verilog and VHDL functional simulations for the design
- Verilog timing simulation

See the README file in `XAPP648.zip` for additional information on how to simulate the design.

SWIFT models are used for RocketIO transceiver simulation. A SWIFT compliant simulator must be used with this reference design.

Design Resources

Table 8 shows the FPGA resources required to implement this reference design in a Virtex-II Pro XC2VP7-5 FF676 device. Resource utilization is based upon place-and-route results from Xilinx ISE 6.3i SP2.

Design performance is limited by the external SSRAM memory clock speed of 100 MHz. The maximum baud rate for the transceiver similarly is 2 Gb/s, regardless of device speed grade.

Table 8: **Design Resources for XC2VP7-FF676 Implementation**

RocketIO Transceivers	DCM	RAMB16	Slices	PPC405	Baud Rate	REFCLK Frequency	I/O
4 of 8	2 of 4	16 of 44	2195 of 4928	0 of 1	2 Gb/s	100 MHz	46 of 396
50%	50%	36%	44%	0%	maximum	maximum	11%

Notes:

1. Depending upon the implementation tool synthesis and place-and-route properties (specifically optimization for either area or speed), the slice results can vary significantly.

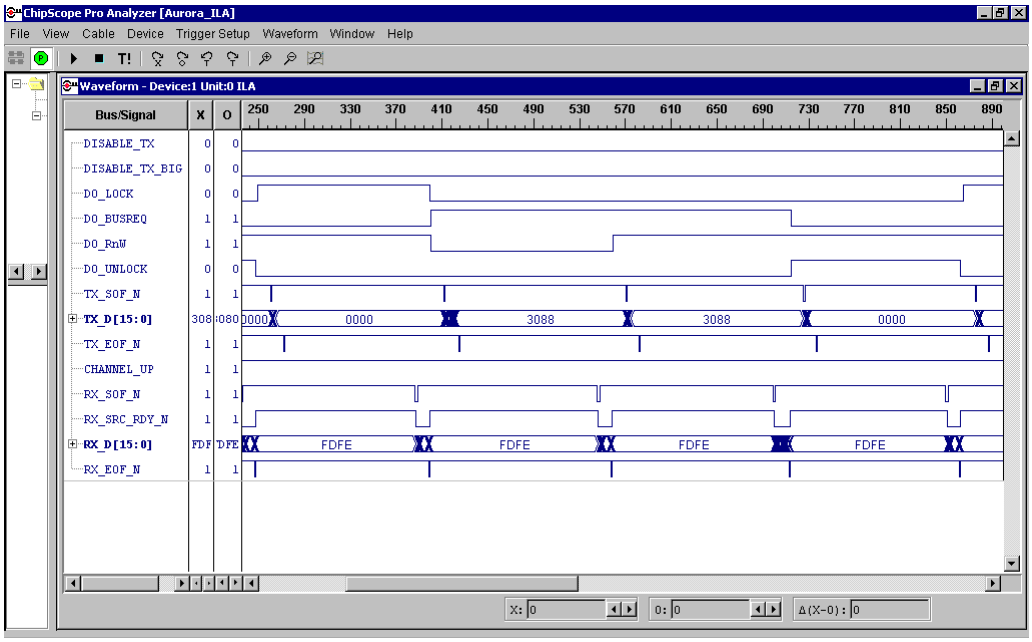
Reference Design Sequence of Operations Example

Figure 14 through Figure 22 contain Xilinx ChipScope™ Pro Integrated Logic Analyzer waveforms demonstrating a typical shared memory transaction between one of the SHBA devices and the SSMA device. The ChipScope Pro logic analyzer waveform signals are captured inside the FPGA, at the SHBA side of the interface.

Control packets, generated by the SHBA, are defined by the transmit data (TX_D[15:0]), the transmit start-of-frame (TX_SOF_N), and the transmit end-of-frame (TX_EOF_N) signals. Status packets, generated by the SSMA, are defined by the receive data (RX_D[15:0]), the receive start-of-frame (RX_SOF_N), the receive source ready (RX_SRC_RDY_N), and the receive end-of-frame (RX_EOF_N) signals. The DO_LOCK, DO_BUSREQ, DO_RnW, and DO_UNLOCK signals indicate the type of shared memory transaction in process. The signals are sampled using the 100 MHz clock.

Figure 14 shows a complete protocol cycle consisting of lock request, lock status, write request, write status, read request, read status, unlock request, and unlock status packets. All

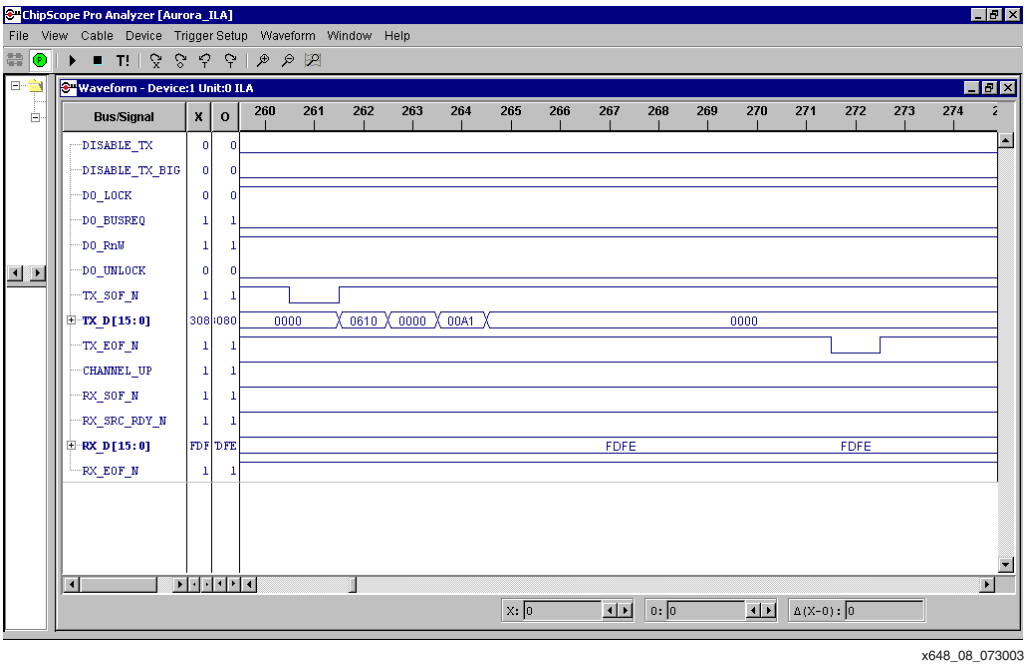
request packets are generated by the SSHA, and all status packets are generated by the SSMA. Each individual packet type is expanded in Figure 15 through Figure 22.



x648_07_073003

Figure 14: **Complete Shared Memory Access Interlocked Handshake Example**

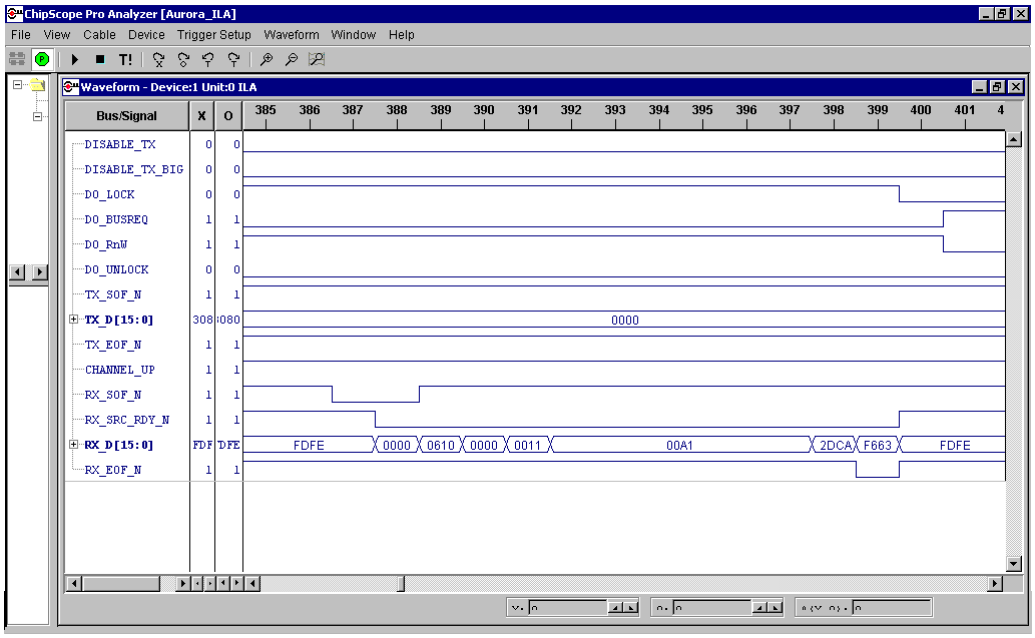
Figure 15 shows a lock request packet. The semaphore address field is 0x00610, and the PORT_ID field is 0xA.



x648_08_073003

Figure 15: **Semaphore Memory Lock Request Packet**

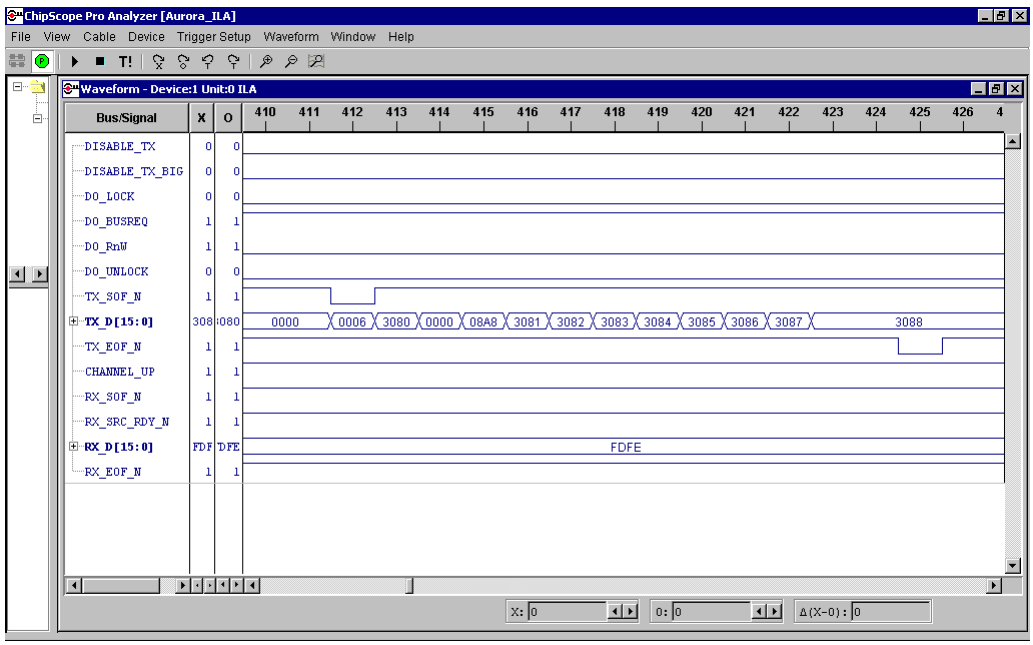
Figure 16 shows a lock status packet received from the SSMA. The lock address is 0x00610. The status field contains the value 0x0011, which indicates a lock request (bit 0 asserted) has completed (bit 4 asserted). The data field contains the value 0x00A1, which indicates this adapter (PORT_ID = 0xA) has been granted a lock to semaphore address 0x610.



x648_09_073003

Figure 16: **Semaphore Memory Lock Status Packet**

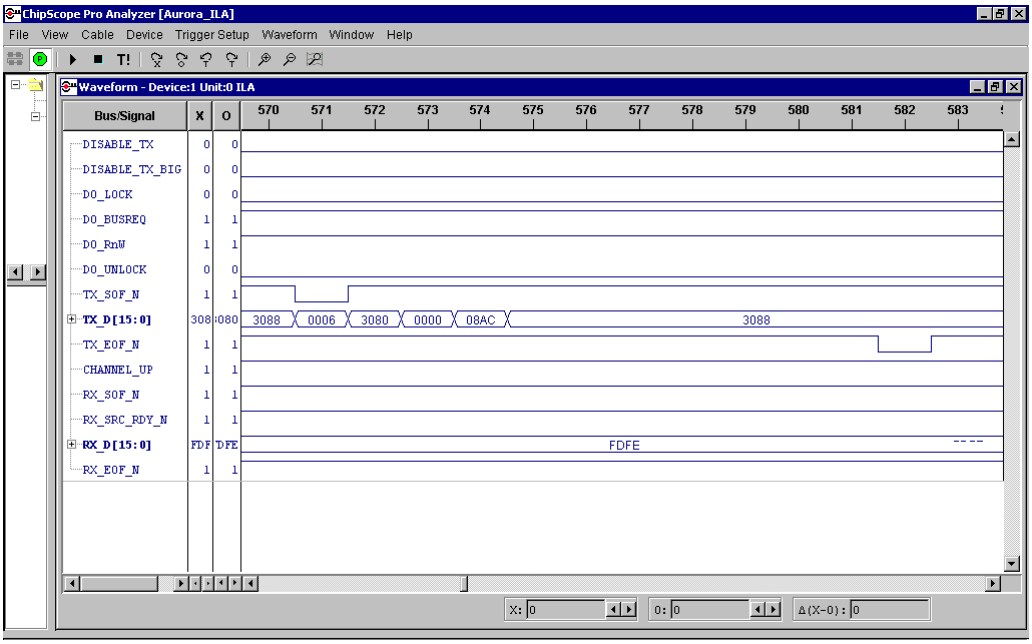
Figure 17 shows a shared memory write request packet. The shared memory starting address is 0x63080. The transfer count is 0x8, PORT_ID =0xA, bit 3 is asserted (BUSREQUEST), and bit 2 is deasserted (write). The first data value is 0x3081. Each subsequent write data value is incremented by one. The CRC placeholder for the packet is 0x30883088.



x648_10_073003

Figure 17: **Shared Memory Write Request Packet**

Figure 18 shows a shared memory read request packet. Again the starting address for the shared memory read location is 0x63080. The control word is requesting eight halfwords to be returned in the status packet from the SSMA. Control word bits 3 and 2 are both asserted, indicating a shared memory read request. The padding and CRC placeholders in the transmit data fields contain the value 0x3088.

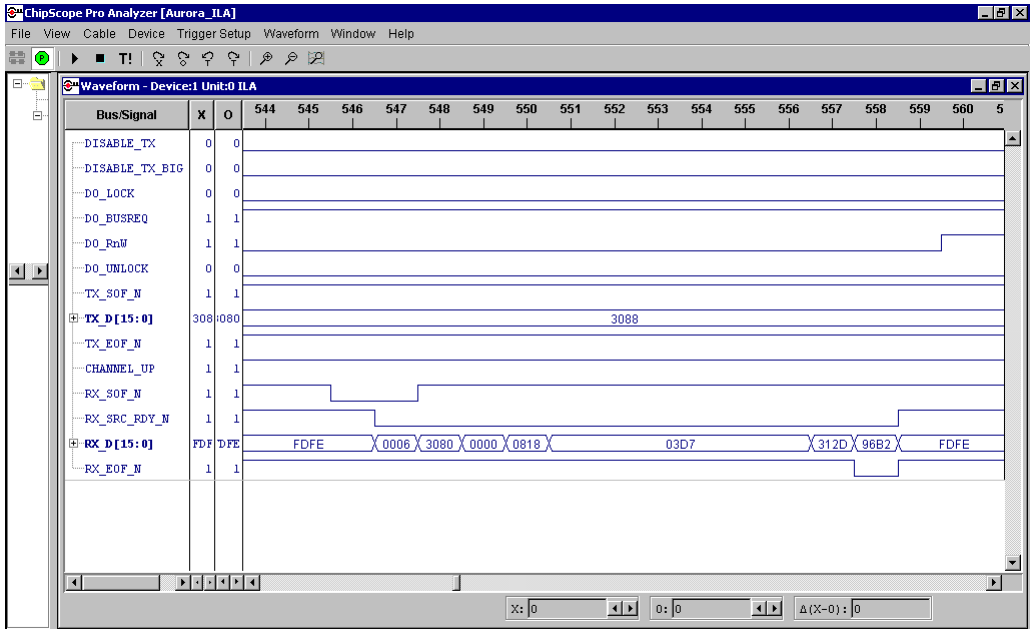


x648_12_073003

Figure 18: **Shared Memory Read Request Packet**

Figure 19 shows a memory write status packet. The shared memory address is 0x063080. The status field bit 4 is asserted, which indicates the write operation has completed. Note that the command field position at bit 5 is deasserted, which indicates the SSMA received the write

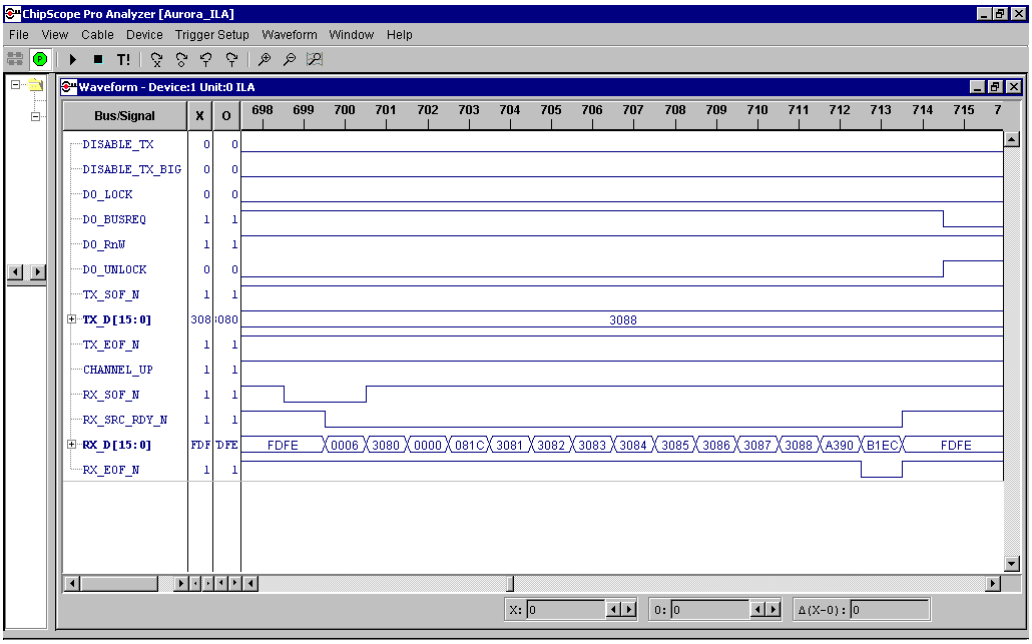
request packet without a CRC packet error. The padding fields contain the value 0x3D7. The CRC word value, calculated by the SSMA RocketIO transceiver, contains the value 0x312D96B2.



x648_11_073003

Figure 19: **Shared Memory Write Status Packet**

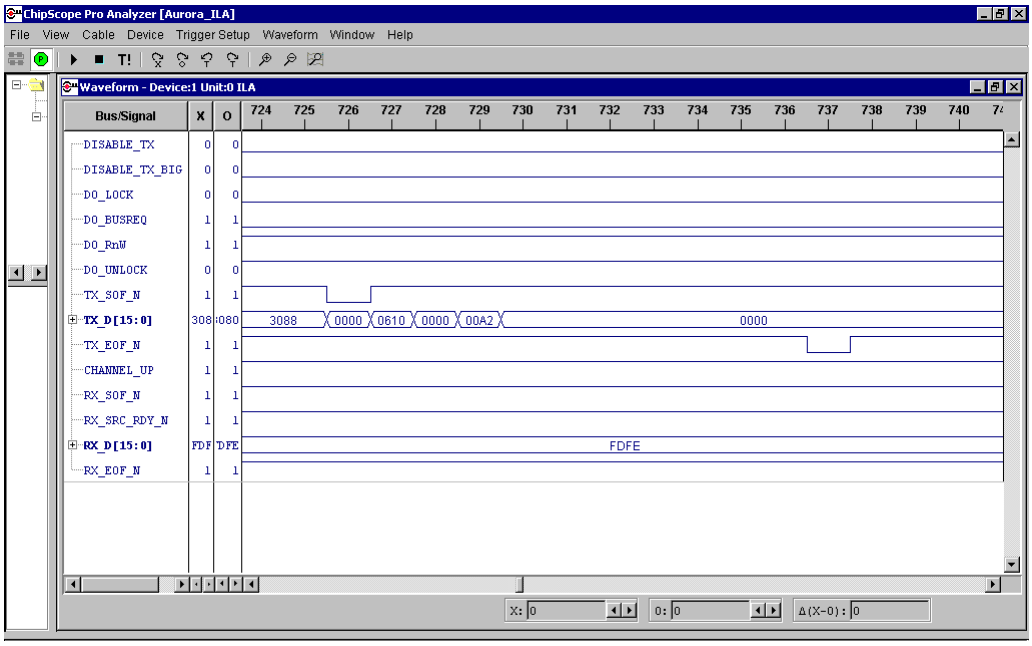
Figure 20 shows the shared memory read status packet. The read address is 0x063080. Eight halfwords are contained in the status packet. The first data halfword is 0x3081, and each subsequent data field is incremented by one. Note that the write data packet contents shown in Figure 17 and the read status data packet contents in Figure 20 agree! The SHBA successfully has written to and read back from the shared memory system.



x648_13_073003

Figure 20: **Shared Memory Read Status Packet**

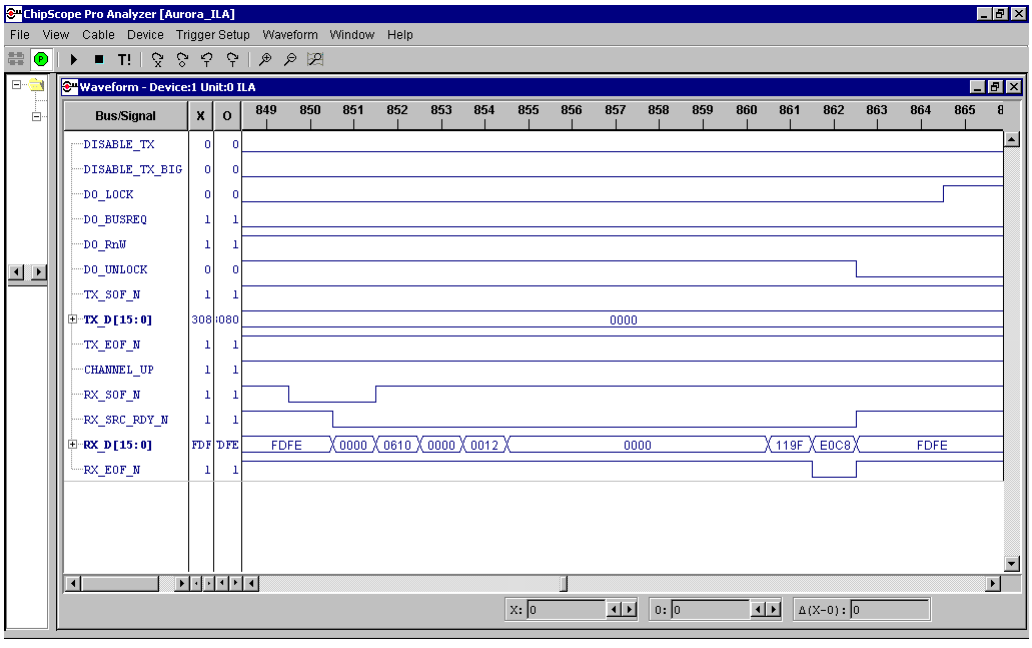
The final step in the protocol requires the SHBA to unlock the semaphore when the host has completed all read/write accesses to the shared memory. Figure 21 shows the UNLOCK request packet issued to accomplish this step. The semaphore address contains the value 0x00610. Bit 1 of the control field is asserted, indicating an UNLOCK request. The data field value is 0x00. Writing a value of 0x00 to the semaphore byte clears all bits in the semaphore memory location, thus removing the hardware inserted lock performed by the SSMA hardware.



x648_14_073003

Figure 21: **Semaphore Memory Unlock Request Packet**

Figure 22 shows the SSMA generated unlock status packet. Receipt of this packet completes the shared memory access protocol cycle, provided that CRC_ERROR (status bit 5) is deasserted. The status packet indicates that semaphore address 0x0610 has been unlocked successfully.



x648_15_073003

Figure 22: **Semaphore Memory Unlock Status Packet**

Conclusion

This reference system design demonstrates a multi-ported, low pin count, shared memory interface suitable for multiprocessor backplane applications. The design can be used to replace a bused, multiple adapter, backplane interface. Multiple, point-to-point Virtex-II Pro RocketIO MGTs provide higher bandwidth, lower pin count, and more transactions per pin interface than a similar bused system is capable.

Utilizing the Virtex-II Pro RocketIO transceiver and the Aurora protocol engine provides the following system advantages:

- The I/O interface between the SHBA and SSMA is reduced greatly. In the case of a 64-bit PCI bus implementation, 80+ signals are required to interface to a shared memory adapter. For this reference design, only 16 signals are required to interface four SHBA to a four-port shared memory system.
- Shared memory system bandwidth is increased.
  - Parallel transmission of data packets, across the point-to-point links, permits some of the overhead in access time to be reduced. In a shared bus configuration, each adapter has to wait for data transmission until the bus is granted.
  - Separating the semaphore memory and shared memory arbiters permits two transfers to occur in parallel rather than sequentially in a parallel bus system.
- The integrity of the data transmission, across the backplane interface, is enhanced through enabled data packet cyclic redundancy checking in the RocketIO transceiver. Parallel bus standards typically only provide byte or word parity checking.
- The Aurora protocol engine with LocalLink interface removes much of the complexity involved in implementing an MGT design.
- Channels can be added or removed from the reference design to suit end-user applications. Replicating protocol logic and modifying the arbiter adds additional channels. The architecture supports alternate or larger capacity shared memory by changing the memory controller interface module in the reference design.
- Redundant channels can be constructed easily with this architecture.
- The architecture supports both homogeneous and heterogeneous distributed processing elements.

Atomic access to a shared memory, across a serial backplane bus, is guaranteed through the use of the following simple interface rules enforced by application software:

- Shared memory access is not attempted prior to arbitrating for and receiving a semaphore memory lock.
- All serial host bus adapters agree on mapping of semaphore memory locks to shared memory block reservation sizes.
- After completing the read/write access to shared memory, the serial host bus adapter releases the lock by requesting an UNLOCK control packet request to the SSMA.
- If a failure occurs in a serial host bus adapter, other processing elements can remove the lock and continue processing.

Related Materials and References

Refer to the following Xilinx documents for additional information:

- UG024: *Virtex-II Pro RocketIO Transceiver User Guide*
- UG061: *Aurora Reference Design*
- WP162: *Multiprocessor Systems White Paper*

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
07/31/03	1.0	Initial Xilinx release.
11/30/04	1.1	Updated reference design and added sections for v2.1 of the Aurora protocol engine delivered through Xilinx CoreGen and ISE 6.3.02i.