Introduction

The Altera® Stratix® II GX 10 Gigabit Ethernet (10GbE) loopback reference design provides a sample design that demonstrates wire-speed operation of the 10GbE reference design described in AN516: 10-Gbps Ethernet Reference Design. The loopback reference design is an SOPC Builder system that includes two instances of the custom 10GbE component from the earlier reference design and a 10 Gigabit Attachment Unit Interface (XAUI) transceiver for each. This design provides a general platform on which you can control, test, and monitor 10GbE operations.

The loopback reference design has the following features:

■ Stand-alone and easy-to-use reference design example.
■ Requires minimal hardware.
■ Supports 10GbE operations in XAUI mode.
■ Supports programmable settings for number of packets, packet length, and payload-data type.
■ Demonstrates transmission and reception of Ethernet packets at the maximum theoretical data rates without errors, through both external loopback and internal loopback paths.
■ Supports external loopback through external X2 modules with a multimode optical cable assembly.
■ Includes support for gathering throughput statistics.
■ Provides custom Nios II IDE command-line interface (CLI) commands to control the design, monitor transmitted and received packets, and generate and display statistics information.

General Description

The loopback reference design is a fully operating subsystem that integrates two 10GbE reference design functions, the MediaAccess Control (MAC) protocol implementation and the XAUI implementation, for Ethernet applications. It uses the PCI Express Development Kit, Stratix II GX Edition, upgraded with the Stratix II GX EP2SGX130GF1508C3N FPGA, in a hardware platform that includes two HSMC-to-X2 boards and two optical X2 modules. The loopback reference design also provides software, including drivers, programming information, and a custom command-line interface (CLI) to control the hardware.

The loopback reference design connects each 10GbE reference design to an optical X2 module through a serial transceiver in XAUI mode. The Ethernet packets are looped back externally between the two 10GbE components through the two X2 modules, which are connected with a multimode optical cable assembly.

Loopback Reference Design Overview

Figure 1 shows a high level block diagram of the loopback reference design.
Figure 1. Loopback Reference Design Block Diagram
The loopback reference design generates a stream of Ethernet packets from one 10GbE component to the other, and back again. In the second 10GbE component, incoming packets are routed back out through an internal loopback path. A packet generator block creates the packets to be transmitted by the first 10GbE component, and a packet monitor block verifies incoming packet checksums and monitors packet receipt.

**Loopback Reference Design Functional Description**

Figure 2 shows the SOPC Builder components that comprise this system. For a block diagram and short overview of the Ethernet packet datapath, refer to “Loopback Reference Design Overview” on page 1.

**Figure 2.** SOPC Builder System for the 10GbE Loopback Reference Design

The following sections describe the roles of the main system components in the 10GbE loopback reference design, and describe the loopback datapath, the memory map, the system-specific registers, and the top-level interface signals of the SOPC Builder system.
**Nios II Embedded Processor**

The Nios II embedded processor is the control plane for setting up and configuring the system components. It triggers the start of Ethernet packet generation and monitors the status of packet reception.

**On-Chip Memory**

This memory stores the executable program. After the program is compiled, it is stored in this memory and executed by the Nios II embedded processor. The on-chip memory is a 256-KByte RAM.

**Phase-Locked Loop (PLL)**

The phase-locked loop (PLL) component takes its input clock from a 100-MHz crystal on the development board and generates the PLL output 83.33-MHz clock (sys_clk), the system-wide clock source for the SOPC Builder system.

**JTAG UART**

The JTAG UART sends serial character streams between the Nios II embedded processor and the rest of the SOPC Builder system. This component provides the mechanism for the FPGA to communicate with the nios2-terminal. The core provides a simple Avalon Memory-Mapped (Avalon-MM) interface that hides the complexities of the JTAG interface from embedded software programmers. The Nios II embedded processor communicates with the 10GbE component by reading and writing control and data registers.

**Interval Timer**

The interval timer component is a 32-bit timer that the Nios II processor system uses to calculate the performance and throughput of the 10GbE operations.

**Altera 10-Gbps Ethernet Reference Design Component**

The 10GbE reference design component provides an integrated Ethernet MAC, physical coding sublayer (PCS), and physical medium attachment (PMA) solution for Ethernet applications. It transmits Ethernet packets from an Avalon Streaming (Avalon-ST) interface to four lanes of 3.125-Gbaud XAUI serial transceiver interface, and receives packets sent to it on the XAUI interface.

For additional information about the 10GbE reference design, refer to *AN516: 10-Gbps Ethernet Reference Design*.

**Ethernet Packet Generator**

The Ethernet Packet Generator module is a custom SOPC Builder component. This component has an Avalon-MM slave interface for control signals and Avalon-ST source interface for Ethernet packet information. This component drives the 10GbE reference design component Transmit FIFO interface, by transmitting a stream of Ethernet packet contents to the 10GbE Transmit FIFO.

Figure 3 shows a high level block diagram of the Ethernet Packet Generator module.
This module contains the following components:
- Avalon-MM register file
- Ethernet packet generation block
- Altera CRC Generator MegaCore function
- Shift Register (RAM-based) megafunction

The Avalon-MM register file provides a register interface for the system to configure all of the parameters and settings necessary for Ethernet packet generation. After the settings are configured, the Ethernet Packet Generator starts generating and transmitting a stream of Ethernet packets to the 10GbE reference design module.

The system configures the following programmable settings through the register interface:
- Total number of packets to be transmitted
- Incremental or random data type
- Fixed or random packet length
- Source and destination MAC address
- Random seed for the pseudo-random binary sequence (PRBS) generation block

In addition, the Avalon-MM register file provides the status of the transmit operation and reports the number of packets that were successfully transmitted. Refer to Table 2 on page 10 for details of the Ethernet Packet Generator registers.

After the system configures all the register settings, it sets the start bit of the Operation register to 1. Setting this start bit resets the Ethernet packet generation block state machine and all the Ethernet Packet Generator status registers. The Ethernet packet generation block generates an Ethernet packet header and a data payload for each packet.

To abort transmission, you can set the stop bit of the Operation register by pressing Enter. When the stop bit is asserted, the state machine completes generating the current packet and then stops.
The Ethernet packet generation block sends each packet to the CRC Generator MegaCore function and to the RAM-based shift register megafonction, ALTSHIFT_TAPS. The CRC Generator MegaCore function calculates the checksum for the packet, and the RAM-based shift register stores the packet until the checksum is available. After the Ethernet Packet Generator merges the valid CRC checksum with the packet stream, it sends the complete packet to the Avalon-ST interface.

Embedding the CRC checksum in the generated packet allows the receive interface to verify the packet easily. If a packet is dropped, the system can continue verifying the incoming packets based on their individual embedded CRC checksum values. This method enables the system to keep an accurate count of the total number of packets received. The packets that the Ethernet Packet Generator generates are standard Ethernet packet contents—Ethernet packets without the 1-byte Start field, 6-byte Preamble, 1-byte Start Frame Delimiter (SFD), and 4-byte MAC-calculated Frame Check Sequence (FCS) that the 10GbE component adds later. Figure 4 describes the format of the frames generated by the Ethernet Packet Generator.

**Figure 4. Ethernet Packet Generator Output Frame Format**

<table>
<thead>
<tr>
<th>Frame Length</th>
<th>Packet Generated by Ethernet Packet Generator</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 byte</td>
<td>START</td>
</tr>
<tr>
<td>6 bytes</td>
<td>PREAMBLE</td>
</tr>
<tr>
<td>1 byte</td>
<td>START FRAME DELIMITER</td>
</tr>
<tr>
<td>6 bytes</td>
<td>DESTINATION ADDRESS</td>
</tr>
<tr>
<td>6 bytes</td>
<td>SOURCE ADDRESS</td>
</tr>
<tr>
<td>2 bytes</td>
<td>LENGTH/TYPE</td>
</tr>
<tr>
<td>2 bytes</td>
<td>SEQUENCE NUMBER</td>
</tr>
<tr>
<td>0 - 1494 or 9576 bytes</td>
<td>PAYLOAD DATA</td>
</tr>
<tr>
<td>4 bytes</td>
<td>MAC FRAME CHECK SEQUENCE</td>
</tr>
<tr>
<td>4 bytes</td>
<td></td>
</tr>
</tbody>
</table>

You program the destination and source MAC addresses through the register interface. You can set the destination MAC address to be a unicast or a broadcast address. In the loopback reference design, a unicast destination MAC address must be the 10GbE component’s receiving MAC address. The source MAC address must be programmed to be the 10GbE component’s transmitting MAC address. This practice ensures that the packets are transmitted to the 10GbE component correctly and that the component can verify the contents of the packets it receives.

The packet length can be set to a programmable value or set to be random packet size. A fixed packet length can range from 24 to 9600 bytes, and a random packet length can range from 24 to 1518 bytes. A packet with length set to less than 64 bytes is automatically padded with padding bytes (0x00) to make it at least 64 bytes long.

The data payload can be incremental or pseudo-random. If you select the incremental data setting, the payload’s initial data value is 0x00, and each subsequent data value is incremented by 1. However, if the pseudo-random data type is selected, the random value generated by the PRBS block becomes the data content of the payload. To program the random seed used by the PRBS block, set the rand_seed0, rand_seed1, and rand_seed2 registers in the register file.
The 2-byte sequence number and 4-byte CRC checksum are part of the packet payload in the loopback reference design. The minimum payload length for packets in the loopback reference design is therefore six bytes. The sequence number is stored in the first two bytes of every packet payload and is used to keep track of the sequence of packets received for debugging purposes. The CRC checksum calculated by the CRC Generator MegaCore function is stored in the final four bytes of the packet payload. The checksum increases confidence in the data integrity of the packets received by the Ethernet Packet Monitor.

For more information about the CRC Generator MegaCore function, refer to the CRC Compiler User Guide.

For more information about the RAM-based shift register megafunction, ALTSHIFT_TAPS, refer to the Shift Register (RAM-Based) (ALTSHIFT_TAPS) Megafunction User Guide.

**Ethernet Packet Monitor**

The Ethernet Packet Monitor module is a custom SOPC Builder component. This component has an Avalon-MM slave interface for control signals and an Avalon-ST sink interface for Ethernet packet information. This component is driven by the 10GbE reference design component Receive FIFO interface. It receives the stream of Ethernet packets and verifies the contents of the payload.

Figure 5 shows a block diagram of this module.

**Figure 5.** Ethernet Packet Monitor Block Diagram

This module consists of the following components:

- Altera CRC Checker MegaCore function
- Avalon-MM register file

The Ethernet packets that arrive at the receiving 10GbE component in the loopback reference design are streamed directly to the CRC Checker MegaCore function in the Ethernet Packet Monitor module. The CRC Checker MegaCore function calculates the correct CRC checksum for the received packet and verifies the calculated value against the checksum value embedded in the final four bytes of the packet payload. It then outputs a status signal, crcbad, that identifies whether the packet received is good or corrupted, and updates the statistics registers accordingly.
The Avalon-MM register file provides a register interface for the system to configure all of the settings necessary to start and monitor Ethernet packet reception. Through this register interface, the system can configure the total number of packets to be received. The Ethernet Packet Monitor module provides the status of the receive operation and a set of statistics counters that track the number of good packets received, the number of bad packets received, the number of bytes received, and the number of clock cycles. This information is used to calculate the performance and throughput rate of the loopback reference design. Refer to Table 5 on page 12 for descriptions of the fields in the Ethernet Packet Monitor register.

After the system configures the register settings, it sets the `start` bit of the Ethernet Packet Monitor Receive Control and Status register. Setting this `start` bit resets the packet reception statistics counters and enables the Ethernet Packet Monitor to receive incoming packets. When the `stop` bit of the Receive Control and Status register is asserted, the Ethernet Packet Monitor stops receiving packets and stops updating the statistics counters. Refer to Table 6 on page 13 for descriptions of the fields in the Ethernet Packet Monitor Receive Control and Status register.

For more information about the CRC Checker MegaCore function, refer to the CRC Compiler User Guide.

**Ethernet Packet Loopback Datapath**

The Stratix II GX PCI Express development board has two HSMC connectors. To assemble the hardware for the loopback reference design, you populate the two HSMC connectors with two HSMC-to-X2 boards, connect an optical X2 module to each of the two boards, and attach the optical cable assembly to the two X2 modules.

The loopback reference design implements an external loopback connection through the two optical X2 modules using the optical cable assembly. Figure 6 shows a high level diagram of the external loopback operation.

**Figure 6. External Loopback Operation Through Optical Cable Assembly**
When the design operates in external loopback mode using the optical cable assembly, one optical cable connects the X2 optical modules. This cable forms a link between the two 10GbE reference design components, eth_10ginst and eth_10ginst_1, in XAUI mode.

In addition, the loopback reference design sends data through the internal loopback path in eth_10ginst_1, the 10GbE component connected to Port B. As shown in Figure 1 on page 2, the data packets flow through the following path:

1. From the eth_10ginst instance of the 10GbE reference design component through the Port A XAUI transceiver connection
2. Through the optical cable assembly to the Port B XAUI transceiver connection
3. To the eth_10ginst1 instance of the 10GbE reference design component through the Port B XAUI transceiver connection
4. Through an internal loopback path in eth_10ginst_1
5. From eth_10ginst_1 through the Port B XAUI transceiver connection
6. Through the optical cable assembly to the Port A XAUI transceiver connection
7. To eth_10ginst through the Port A XAUI transceiver connection

After system configuration completes, the Ethernet Packet Generator begins generating packets to the transmitting 10GbE function through the Transmit FIFO interface. The 10GbE MAC prefixes a 1-byte Start field, a 6-byte Preamble, and a 1-byte SFD and appends a 4-byte FCS to each packet and transmits the packet to the transmitting XAUI transceiver. This XAUI transceiver sends the packets through the optical cable to a XAUI transceiver in the other 10GbE component. The second 10GbE component sends the packet through its internal loopback datapath and back through the optical cable to the receiving function of the originating 10GbE component. After the receiving 10GbE component receives a packet, it verifies the packet with its FCS. The remainder of the packet is streamed out through the Receive FIFO interface to the Ethernet Packet Monitor, which verifies the checksum.

After the system receives the total number of expected packets, it reports the packet statistics and the throughput rate based on the statistics registers. This information confirms the real throughput rate that the 10GbE reference design can achieve.

**Loopback Reference Design Control**

This section describes the details of the loopback reference design control. It includes the following topics:

- Memory Map
- Register Map
- Interface Signal Descriptions

**Memory Map**

Table 1 describes the memory map of the SOPC Builder system for the loopback reference design.
Table 1. 10GbE Loopback Reference Design Memory Map

<table>
<thead>
<tr>
<th>Address</th>
<th>SOPC Builder Component Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00040000 – 0x0007FFFF</td>
<td>onchip_mem</td>
<td>On-Chip Memory - RAM (256KB)</td>
</tr>
<tr>
<td>0x00080000 – 0x00080FFF</td>
<td>cpu</td>
<td>CPU JTAG Debug Module</td>
</tr>
<tr>
<td>0x00081000 – 0x000813FF</td>
<td>eth_10ginst</td>
<td>10-Gbps Ethernet 0</td>
</tr>
<tr>
<td>0x00081400 – 0x000817FF</td>
<td>eth_10ginst_1</td>
<td>10-Gbps Ethernet 1</td>
</tr>
<tr>
<td>0x00081800 – 0x0008183F</td>
<td>gen</td>
<td>Ethernet Packet Generator</td>
</tr>
<tr>
<td>0x00081840 – 0x0008187F</td>
<td>mon</td>
<td>Ethernet Packet Monitor</td>
</tr>
<tr>
<td>0x00081880 – 0x0008189F</td>
<td>pll</td>
<td>Phase Lock Loop</td>
</tr>
<tr>
<td>0x000818A0 – 0x000818BF</td>
<td>timer</td>
<td>Interval Timer 0</td>
</tr>
<tr>
<td>0x000818C0 – 0x000818DF</td>
<td>timer_1</td>
<td>Interval Timer 1</td>
</tr>
<tr>
<td>0x000818E0 – 0x000818E7</td>
<td>jtag_uart</td>
<td>JTAG UART</td>
</tr>
</tbody>
</table>

Table 2. Ethernet Packet Generator Register Map (Part 1 of 2)

<table>
<thead>
<tr>
<th>Address Offset</th>
<th>Register Name</th>
<th>Description</th>
<th>Access</th>
<th>HW Reset</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>number_packet</td>
<td>32-bit Number of Packets register. Total number of packets to be generated by the Ethernet Packet Generator and sent to the 10GbE component.</td>
<td>RW</td>
<td>0x0</td>
</tr>
<tr>
<td>0x04</td>
<td>config_setting</td>
<td>16-bit Configuration Setting register. Refer to Table 3 for register field descriptions.</td>
<td>RW</td>
<td>0x0</td>
</tr>
<tr>
<td>0x08</td>
<td>operation</td>
<td>3-bit Operation register. Refer to Table 4 for register field descriptions.</td>
<td>RW/RO</td>
<td>0x0</td>
</tr>
<tr>
<td>0x0C</td>
<td>source_addr0</td>
<td>32-bit Lower Source MAC Address register. Bits [31:0] of the source MAC address field in the Ethernet packet. This source MAC address should be programmed to the MAC address of the transmitting 10GbE component.</td>
<td>RW</td>
<td>0x0</td>
</tr>
<tr>
<td>0x10</td>
<td>source_addr1</td>
<td>16-bit Upper Source MAC Address register. Bits [15:0] of this register hold bits [47:32] of the source MAC address field in the Ethernet packet. This source MAC address should be programmed to the MAC address of the transmitting 10GbE component. Bits [31:16] of this register are reserved.</td>
<td>RW</td>
<td>0x0</td>
</tr>
</tbody>
</table>
Table 2. Ethernet Packet Generator Register Map (Part 2 of 2)

<table>
<thead>
<tr>
<th>Address Offset</th>
<th>Register Name</th>
<th>Description</th>
<th>Access</th>
<th>HW Reset</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x14</td>
<td>destination_addr0</td>
<td>32-bit Lower Destination MAC Address register.</td>
<td>RW</td>
<td>0x0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bits [31:0] of the destination MAC address field in the Ethernet packet.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>This destination MAC address should be programmed to the receiving 10GbE</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>component's MAC address for unicast packets.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x18</td>
<td>destination_addr1</td>
<td>16-bit Upper Destination MAC Address register.</td>
<td>RW</td>
<td>0x0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bits [15:0] of this register hold bits [47:32] of the destination MAC</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>address field in the Ethernet packet. This destination MAC address should</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>be programmed to the receiving 10GbE component's MAC address for unicast</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>packets.                      Bits [31:16] of this register are reserved.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x1C</td>
<td>rand_seed0</td>
<td>32-bit Lower Random Seed register.</td>
<td>RW</td>
<td>0x0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The value in this register is preloaded in bits[31:0] of the PRBS generator</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>when config_setting[15] is set to 1.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x20</td>
<td>rand_seed1</td>
<td>32-bit Middle Random Seed register.</td>
<td>RW</td>
<td>0x0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The value in this register is preloaded in bits [63:32] of the PRBS generator</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>when config_setting[15] is set to 1.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x24</td>
<td>rand_seed2</td>
<td>32-bit Upper Random Seed register.</td>
<td>RW</td>
<td>0x0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The value in this register is preloaded in bits [91:64] of the PRBS generator</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>when config_setting[15] is set to 1.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x28</td>
<td>packet_tx_count</td>
<td>32-bit Packet Transmit Count register.</td>
<td>RO</td>
<td>0x0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Current count of data packets successfully transmitted by the Ethernet</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Packet Generator. This register resets when operation[0] is asserted.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 3. Configuration Setting Register Field Descriptions

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field Name</th>
<th>Description</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>length_sel</td>
<td>Packet-length type select. The value in this bit determines the packet-length type according to the following list:</td>
<td>RW</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: Fixed packet length.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: Random packet length.</td>
<td></td>
</tr>
<tr>
<td>[14:1]</td>
<td>pkt_length</td>
<td>Fixed packet length. Programmable packet length is specified in bytes, and ranges from 24 to 9600. This field is valid only when the Packet-length type select bit has value 0.</td>
<td>RW</td>
</tr>
<tr>
<td>15</td>
<td>pattern_sel</td>
<td>Data type select. The value in this bit determines the data pattern type according to the following list:</td>
<td>RW</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: Incremental data pattern.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: Random data pattern.</td>
<td></td>
</tr>
<tr>
<td>[31:16]</td>
<td>Reserved</td>
<td>Reserved. Read returns 0.</td>
<td>RO</td>
</tr>
</tbody>
</table>
Table 4. Operation Register Field Descriptions

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field Name</th>
<th>Description</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>start</td>
<td>Start operation. Set to 1 to start the packet generation from the Ethernet Packet Generator.</td>
<td>RW</td>
</tr>
<tr>
<td>1</td>
<td>stop</td>
<td>Stop operation. Set to 1 to stop the Ethernet Packet Generator.</td>
<td>RW</td>
</tr>
<tr>
<td>2</td>
<td>tx_done</td>
<td>Transmit-done status. This bit is set to 1 when the Ethernet Packet Generator completes transmission of all the packets programmed in the Number of Packets register. This bit is cleared when the Operation register start bit is set to 1.</td>
<td>RO</td>
</tr>
<tr>
<td>[31:3]</td>
<td>Reserved</td>
<td>Reserved. Read returns 0.</td>
<td>RO</td>
</tr>
</tbody>
</table>

Table 5 describes the Ethernet Packet Monitor registers and Table 6 describes the fields of the Ethernet Packet Monitor Receive Control and Status register.

Table 5. Ethernet Packet Monitor Register Map (Part 1 of 2)

<table>
<thead>
<tr>
<th>Address Offset</th>
<th>Name</th>
<th>Description</th>
<th>Access</th>
<th>HW Reset</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>number_packet</td>
<td>32-bit Number of Packets register. Total number of packets to be received by the Ethernet Packet Monitor.</td>
<td>RW</td>
<td>0x0</td>
</tr>
<tr>
<td>0x04</td>
<td>cycle_deassert_ready</td>
<td>32-bit Cycle Deassert Ready register. Numbers of cycles from the start-of-packet of the first packet received to deassertion of the Avalon-ST RX Ready signal. This delay is critical for flow control to the Avalon-ST source.</td>
<td>RW</td>
<td>0x0</td>
</tr>
<tr>
<td>0x08</td>
<td>cycle_assert_ready</td>
<td>32-bit Cycle Assert Ready register. Numbers of cycles from deassertion of the Avalon-ST RX Ready signal to reassertion of the Avalon-ST RX Ready signal. This delay is critical for flow control to the Avalon-ST source.</td>
<td>RW</td>
<td>0x0</td>
</tr>
<tr>
<td>0x0C</td>
<td>packet_rx_ok</td>
<td>32-bit Packet Received OK register. Current count of packets received without error.</td>
<td>RO</td>
<td>0x0</td>
</tr>
<tr>
<td>0x10</td>
<td>packet_rx_error</td>
<td>32-bit Packet Received with Error register. Current count of packets received with error.</td>
<td>RO</td>
<td>0x0</td>
</tr>
<tr>
<td>0x14</td>
<td>byte_rx_count0</td>
<td>32-bit Lower Byte Received Count register. Bits [31:0] of the current count of data bytes received.</td>
<td>RO</td>
<td>0x0</td>
</tr>
<tr>
<td>0x18</td>
<td>byte_rx_count1</td>
<td>32-bit Upper Byte Received Count register. Bits [64:32] of the current count of data bytes received.</td>
<td>RO</td>
<td>0x0</td>
</tr>
<tr>
<td>0x1C</td>
<td>cycle_rx_count0</td>
<td>32-bit Lower Cycle Receive Count register. Bits [31:0] of the current count of cycles from start of first packet byte received to end of last packet byte received.</td>
<td>RO</td>
<td>0x0</td>
</tr>
</tbody>
</table>
Table 5. Ethernet Packet Monitor Register Map (Part 2 of 2)

<table>
<thead>
<tr>
<th>Address Offset</th>
<th>Name</th>
<th>Description</th>
<th>Access</th>
<th>HW Reset</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x20</td>
<td>cycle_rx_count1</td>
<td>32-bit Upper Cycle Receive Count register. Bits [64:32] of the current count of cycles from start of first packet byte received to end of last packet byte received.</td>
<td>RO</td>
<td>0x0</td>
</tr>
<tr>
<td>0x24</td>
<td>receive_ctrl_status</td>
<td>10-bit Receive Control and Status register. Refer to Table 6 for register field descriptions.</td>
<td>RW/RO</td>
<td>0x0</td>
</tr>
</tbody>
</table>

Table 6. Receive Control and Status Register Field Descriptions

<table>
<thead>
<tr>
<th>Bit(s)</th>
<th>Bit Name</th>
<th>Description</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>start</td>
<td>Start operation. Write 1 to start packet reception by the Ethernet Packet Monitor. This bit is automatically cleared after packet reception begins.</td>
<td>RW</td>
</tr>
<tr>
<td>1</td>
<td>stop</td>
<td>Stop operation. Write 1 to stop the Ethernet Packet Monitor from receiving further packets. The Ethernet Packet Monitor stops updating the status registers immediately after this bit is asserted. This bit is cleared when the Receive start bit is set to 1.</td>
<td>RW</td>
</tr>
<tr>
<td>2</td>
<td>rx_done</td>
<td>Receive-done status. This bit is set to 1 when the Ethernet Packet Monitor has received all of the packets programmed in the Number of Packets register. This bit is cleared when the Receive start bit is set to 1.</td>
<td>RO</td>
</tr>
<tr>
<td>[31:3]</td>
<td>Reserved</td>
<td>Reserved bits. Read returns 0.</td>
<td>RO</td>
</tr>
</tbody>
</table>

Interface Signal Descriptions

This section describes the top-level interface signals in the loopback reference design. Table 7 lists the clock and reset signals, Table 8 lists the top-level 10GbE component 0 (eth_ginst) signals, and Table 9 lists the top-level 10GbE component 1 (eth_ginst_1) signals.

Table 7. Clock and Reset Signals

<table>
<thead>
<tr>
<th>Signal Name</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>clk</td>
<td>In</td>
<td>Loopback reference design clock. This clock is sourced from the development kit 100-MHz oscillator (X1).</td>
</tr>
<tr>
<td>ref_clk</td>
<td>In</td>
<td>10GbE reference design component transceiver reference clock. This clock is sourced from the development kit 156.25-MHz oscillator (X3).</td>
</tr>
<tr>
<td>reset_n</td>
<td>In</td>
<td>Single reset for all the logic in the reference design. This signal is connected to the development kit RESET (S4) push-button switch.</td>
</tr>
</tbody>
</table>

Table 8. 10GbE Component 0 Signals

<table>
<thead>
<tr>
<th>Signal Name</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>a_xaui_rx[3:0]</td>
<td>In</td>
<td>Port A - XAUI Differential Receive Interface. This bus is connected to the HSMC board.</td>
</tr>
<tr>
<td>a_xaui_tx[3:0]</td>
<td>Out</td>
<td>Port A - XAUI Differential Transmit Interface. This bus is connected to the HSMC board.</td>
</tr>
</tbody>
</table>
Using the Loopback Reference Design

The following sections describe how to set up and use the loopback reference design:

- Hardware and Software Requirements
- Loopback Reference Design Installation
- Preparing to Run the Application
- Running the Benchmark Application Software

## Hardware and Software Requirements

The reference design application requires the following hardware:

- One computer running the Windows XP operating system
- PCI Express Development Kit, Stratix II GX edition, upgraded with the Stratix II GX EP2SGX130GF1508C3N FPGA
- Altera USB-Blaster™ or ByteBlaster™ download cable
- Two HSMC-to-X2 connectors.
- Two optical X2 modules
- One multimode (mm) optical cable assembly

Figure 7 shows the PCI Express Development Kit, Stratix II GX edition, the HSMC-to-X2 connectors, and the two optical X2 modules.

### Table 9. 10GbE Component 1 Signals

<table>
<thead>
<tr>
<th>Signal Name</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>b_xawi_rx[3:0]</td>
<td>In</td>
<td>Port B - XAUI Differential Receive Interface. This bus is connected to the HSMC board.</td>
</tr>
<tr>
<td>b_xawi_tx[3:0]</td>
<td>Out</td>
<td>Port B - XAUI Differential Transmit Interface. This bus is connected to the HSMC board.</td>
</tr>
</tbody>
</table>
Figure 7 also shows the following connectors:
- Altera USB-Blaster download cable
- Power supply connector

The reference design application also requires the Altera Quartus® II 8.0 SP1 software, including the following features:
- USB-Blaster or ByteBlaster driver
- SOPC Builder
- Nios II Embedded Design Suite (EDS)
- 10GbE reference design

**Loopback Reference Design Installation**

This section describes how to install the loopback reference design.

**Downloading the Package**
All of the files necessary for the loopback reference design are included in the 10GbE_Hardware_Demo.zip file. This file is available for you to download from the Stratix II GX 10GbE Loopback Reference Design web page.
Extracting the Files

Unzip the 10GbE_Hardware_Demo.zip file in the working directory you designated for this project. After you unzip the file, your working directory contains the 10GbE loopback reference design subdirectory, 10GbE_Hardware_Demo. Figure 8 shows the 10GbE_Hardware_Demo directory structure.

Figure 8. Loopback Reference Design Directory Structure After Installation

Preparing to Run the Application

Before you can run the application, you must install the required software, connect the hardware, and program the Altera FPGA. The following sections teach you how to perform these steps.

Connecting the Hardware

Before connecting the hardware, you must install the required software listed in “Hardware and Software Requirements” on page 14.

To connect the hardware, perform the following steps:

1. Connect the two HSMC-to-X2 adapter cards to the two HSMC connectors J1 and J2 on the Stratix II GX PCI Express development board.
2. Connect the two X2 modules to the two HSMC-to-X2 adapter cards.
3. Connect the two X2 modules with the optical cable, as shown in Figure 9.
4. Connect the Altera programming cable to the JTAG connection port (J5).
5. Connect the power supply cord to the power supply input (J3).

For information about the development board connectors, refer to the Stratix II GX PCI Express Development Board Reference Manual.

This reference design uses point-to-point configuration. You connect the Ethernet ports to one another directly through the optical cable. This configuration is simple and robust. Figure 9 illustrates the optical cable connections.
Configuring the Board With the Hardware Design

To program the Stratix II GX FPGA with the hardware image for the design, perform the following steps:

1. Open a Nios II command shell.

   To start the Nios II command shell on Windows platforms, on the Start menu, click All Programs. On the All Programs menu, on the Altera submenu, on the Nios II EDS <version> submenu, click Nios II <version> Command Shell.

2. Change to the top-level directory of the unzipped design files, at <installation path>/10GbE_Hardware_Demo/demo.

   The 10gDemo.elf and eth_10g_hw_ref_design.sof files are in this directory.

   ![Figure 9. Optical Cable Connections](image)

   If the SRAM Object File (.sof) or Executable and Linkable Format (.elf) file is missing or corrupted, you must regenerate the design. For details of the regeneration process, refer to “Regenerating and Recompiling the 10GbE Loopback Reference Design” on page 24.

3. Type the following command:

   ```bash
   nios2-configure-sof --device 2 eth_10g_hw_ref_design.sof
   ```

   Figure 10 shows the Nios II command shell after the hardware image is successfully programmed.
Programming the FPGA with the Benchmark Application

The pre-compiled benchmark application is provided with the 10GbE loopback reference design files.

If your software application file, the .elf file, is missing or corrupted, you must regenerate the design. For information about how to regenerate and recompile the 10GbE loopback reference design hardware and software application, refer to “Regenerating and Recompiling the 10GbE Loopback Reference Design” on page 24. Altera recommends that you regenerate and recompile both the hardware and software application, if you require regeneration of the software application files. For information about how to simulate the software application, refer to “Testbench Simulation” on page 26.

To download the pre-compiled benchmark application to system memory and start the Nios II processor, type the following command in the Nios II command shell:

```
nios2-download -g 10gDemo.elf && nios2-terminal
```

This command downloads the Nios II processor’s image file to system memory, and restarts the processor to begin code execution. The command also opens a terminal application, nios2-terminal, which lets you interact with the software application.

If the benchmark application starts successfully, the test menu appears, as shown in Figure 11.
Running the Benchmark Application Software

The Nios II embedded processor runs the custom software application that configures and runs benchmark tests for this system. The application provides a custom menu driven user interface, which you can use to parameterize and run tests on the hardware. As shown in “Example Using the Menu Driven Interface” on page 22, you select options in the test menu by typing the menu item number, followed by the setting value, and pressing Enter. For user-specified options that require a numerical value, you may specify the value in hexadecimal format (for example, 0x1000) or in decimal format (for example, 4000).

The benchmark application has two menus, one for configuring and running a test, and one for viewing reports about tests already run.
Test Menu

The test menu allows you to configure and run a benchmark test in the system. Initially, it displays the default values for the test parameters. You can use it to control the following test parameters:

- **Packet Length**—Controls the length of the packets sent over the link. Specify random size, or a specific fixed size. You can specify a fixed packet-length of 64 bytes to 9600 bytes. If you specify random size, the generated packets can range in length from 24 bytes to 1518 bytes.

- **Number of Packets**—Controls the number of packets transmitted in the test. Specify the value as a 32-bit unsigned integer. The value must be between 1 and \((2^{32} - 1)\), inclusive.

- **Packet Data**—Controls the data pattern type in the payload portion of the packet. Specify increment or random. If you specify the incremental data pattern, the data payload, a 32 bit unsigned integer, increments for each subsequent packet. After this integer reaches \((2^{32} - 1)\), it rolls over to 0 on the next increment operation. If you specify the random data type, a new pseudo-random, 32-bit unsigned integer is used for each data payload.

- **Start Test**—Triggers the test. After the test starts, it runs until the number of packets transmitted is equal to the number specified in the **Number of Packets** setting, or until the user interrupts the test by pressing Enter. After the test stops, the Report Menu appears, displaying the results. During the operation of a test, throughput approximations appear periodically, printed in packets per second.

Report Menu

The report menu displays the status of the tests run on the system, and allows you to specify the report you want to see, or to switch to another menu. The reports menu allows you to store several reports in first-in, first-out order. The default number of reports that can be stored is 10, but you can change this number by modifying the value of `FSIZE` in `test_menus/config.h`. After several reports have been generated, the oldest report data is discarded to make room for new reports. Each report is displayed in the format illustrated in Figure 12.
A report provides the following information about the test run:

- **Number**—An unique integer for each test. Starts at 1 and increments with every test started.
- **Status**—Test status is one of Done, Interrupted, or Error.
- **Run Time**—Total run time of the test.
- **10G Sender MAC**—10GbE MAC sender MAC address.
- **10G Receiver MAC**—10GbE MAC receiver MAC address.
- **Link Speed**—The link speed (always 10000) in Mbits/second and the duplex mode (always full).
- **Packets to Send**—Number of packets the test was configured to send, the packet length, the data type selected for the packets, and the seed value used if the data type is random.
- **Packets Sent**—Total number of packets sent by the Ethernet Packet Generator during the test, and the final throughput rate in packets per second. The actual number of packets may be less than the number specified for the test, if an error occurred in the system.
- **Packets Received**—Total number of packets received by the Packet Monitor during the test, number of valid packets, and number of error packets.
The total number of packets should be equal to the sum of the number of valid packets and the number of error packets.

- **Bytes Received**—Total number of bytes received by the receiver, and the final throughput rate in bits per second. The byte count includes the complete packet lengths (header plus data).

- **Report Control**—Controls the selection of the report. The current test number, first available test number, and final available test number are all displayed. Specify a report to view using the **next** and **previous** controls or by typing a report number.

The total amount of time to complete the test is calculated from the 64-bit **Cycle Receive Count** register in the Ethernet Packet Monitor block. This cycle-count register is reset to 0 at the start of the test and begins incrementing at the system clock frequency rate after the first start-of-packet is received. The register stops counting immediately after all packets are received by the monitor, that is, after the end-of-packet from the final packet is received. You can also stop the register from incrementing by interrupting the test. The total time for the test is computed by dividing the 64-bit value in the **Cycle Receive Count** register by the reference clock (**ref_clk**) frequency (156.25 MHz), using floating point operations.

Although this method of calculating time is very accurate, its accuracy depends on the monitor receiving all the packets during the test. If the monitor does not receive all the packets during the test, or if a major error occurs in the Ethernet Packet Monitor block, inaccurate time values are reported.

**Example Using the Menu Driven Interface**

The following simple test example shows you how to use the test menu.

In this example, we would like to configure the system to send 10,000,000 packets with a data length of 1200 bytes each. To configure and run this test, perform the following steps:

1. To configure the packet size to 1200 bytes, at the test menu Command prompt, type **3e 1200**

2. To configure the number of packets to 10,000,000, at the test menu Command prompt, type **4b**

3. To run the test, at the test menu Command prompt, type **6a**

**Figure 13** shows the Nios II command shell after you perform steps 1 and 2 and type **6a**, but before you press Enter. The new values for the packet size and number of packets are displayed.
**Figure 13.** Benchmark Application Test Menu With Selections

![Image of Benchmark Application Test Menu With Selections]
The test starts, and displays a series of status messages such as that shown in Figure 14.

Figure 14. Progress of Test Status

The messages displayed describe the progress of the test. The length of the arrow represents the percentage of total packets already sent; when the test completes, the arrowhead reaches the right margin. Each arrow is printed with a progressive throughput calculation, providing a rough approximation of the packets per second achieved in the test so far. After the test completes, the Report Menu is displayed. The Report Menu shown in Figure 12 on page 21 is the Report Menu for this example.

Regenerating and Recompiling the 10GbE Loopback Reference Design

If your .sof or .elf file is missing or becomes corrupted, you must regenerate your design. To regenerate and recompile the 10GbE loopback reference design on your computer, perform the following steps:

1. Start the Quartus II software v8.0 SP1.
2. Open the 10GbE loopback reference design Quartus II project, at <installation_path>/10GbE_Hardware_Demo/hardware/10g_hw_ref_design.qpf.
3. On the Tools menu, click SOPC Builder. SOPC Builder opens, displaying the 10GbE loopback reference design SOPC Builder system.
4. Click Generate.
5. Click **Exit** to close SOPC Builder.

6. To add the 10GbE component library to the project after regeneration, on the Assignments menu, click **Settings**.

7. In the **Settings** dialog box, under **Category**, click **Libraries**.

8. Under **Project libraries**, add the directory where you installed the 10GbE reference design library, at `<10GbE reference design installation path>/lib`

   ▶ Note that `<10GbE reference design installation path>` and `<installation_path>` are not the same directory. `<10GbE reference design installation path>` is the location where you installed the 10GbE design component source files, and `<installation_path>` is the location where you installed the 10GbE loopback reference design files.

9. To recompile the design, on the Processing menu, click **Start Compilation**.

10. Open a Nios II command shell.

    To start the Nios II command shell on Windows platforms, on the Start menu, click **All Programs**. On the All Programs menu, on the Altera submenu, on the Nios II EDS `<version>` submenu, click **Nios II `<version>` Command Shell**.

11. Create a new directory in which to store the `.sof` and `.elf` files for the regenerated SOPC Builder system, by typing the following command sequence in the Nios II command shell:

```
   cd <installation_path>/10GbE_Hardware_demo
   mkdir tmp
```

12. Copy the regenerated `.sof` file to the new directory, by typing the following command:

```
   cp hardware/eth_10g_hw_ref_design.sof tmp/.
```

13. Change directories to `<installation_path>/10GbE_Hardware_Demo/software/bsp/10gBSP`

14. Maintain the `create-this-bsp` file in this directory, and remove all other files and subdirectories.

15. Rebuild the library files for the software application, by typing the following command:

```
   ./create-this-bsp
```

16. Change directories to `<installation_path>/10GbE_Hardware_Demo/software/app/10gDemo`

17. Remove the unnecessary files in this directory by typing the following command:

```
   make clean_all
```

18. Rebuild the software application by typing the following command:

```
   make
```

19. Copy the `.elf` file to your new directory by typing the following command:

```
   cp 10gDemo.elf <installation_path>/10GbE_Hardware_Demo/tmp/.
```

20. Change directories to `<installation_path>/10GbE_Hardware_Demo/tmp`
21. To configure the board with the regenerated hardware design, follow the instructions in step 3 in “Configuring the Board With the Hardware Design” on page 17.

22. To download the regenerated software application, follow the instructions in “Programming the FPGA with the Benchmark Application” on page 18.

The application test bench appears in the nios2-terminal application, as shown in Figure 11 on page 19.

Testbench Simulation

You can simulate the 10GbE loopback reference design using the IP functional simulation model and testbench provided with the loopback reference design files. The testbench files are located in the testbench subdirectory, at <installation_path>/10GbE_Hardware_Demo/testbench. The testbench files include a script to run the ModelSim simulator.

To run a simulation using the ModelSim simulator, perform the following steps:

1. Start the ModelSim simulator.
2. Change directories to <installation_path>/10GbE_Hardware_Demo/testbench/.
3. To compile the IP functional simulation model and run this model with the testbench, type the following command:
   
   do run_eth_10g_ref_design_tb.tcl

   In the ModelSim main window, the ModelSim transcript pane displays messages from the testbench.

Referenced Documents

This application note references the following documents:

- AN516: 10-Gbps Ethernet Reference Design
- CRC Compiler User Guide
- Shift Register (RAM-Based) (ALTSHIFT_TAPS) Megafunction User Guide
- Stratix II GX PCI Express Development Board Reference Manual

Document Revision History

Table 10 shows the revision history for this application note.

<table>
<thead>
<tr>
<th>Date and Document Version</th>
<th>Changes Made</th>
<th>Summary of Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>- Update Figure 12 on page 21 using updated design files.</td>
<td></td>
</tr>
<tr>
<td>February 2009 v1.0</td>
<td>Initial release.</td>
<td>—</td>
</tr>
</tbody>
</table>