

Introduction

The Altera® Triple Speed Ethernet (TSE) data path reference design provides a sample SOPC Builder system using the Altera TSE MegaCore® function with two serial transceivers. This reference design demonstrates the operation of the Altera TSE MegaCore function up to the maximum wire-speed performance in hardware. The design enables you to evaluate the TSE MegaCore function for integration into Altera FPGA designs.

The reference design has the following features:

- Uses very few pieces of hardware for a complete test: a Stratix® II GX PCI Express development kit, two small form pluggable (SFP) modules, an Ethernet cable assembly, a PC, and a USB-Blaster™ or ByteBlaster™ cable assembly
- Implements two instances of the Altera TSE MegaCore function and supports 10/100/1000 Mbps Ethernet operations in SGMII mode with auto-negotiation
- Supports programmable settings such as the number of packets, packet length, payload data type, and source and destination MAC addresses
- Demonstrates sending and receiving Ethernet packets up to the maximum theoretical data rates without errors
- Supports external loopback via SFP modules with CAT5 Ethernet cable assembly and optionally through an Ethernet switch
- Provides a command line interface (CLI) to control the design and monitor TX and RX packet statistics results
- Provides serial interfaces to configure Copper SFP module Ethernet PHY

General Description

The reference design demonstrates a fully operational subsystem that integrates two Altera TSE MegaCore functions (MAC + PCS + PMA) for Ethernet applications. The design uses the Stratix II GX PCI Express Development Kit as a hardware platform, which includes two SFP cages. The reference design also comes with software that includes drivers, programming information, and a CLI to conduct the demonstrations or tests.

This reference design interfaces the TSE MegaCore function with a Copper SFP module via a 1.25 Gbps serial transceiver that enables all 10, 100, and 1000 Mbps Ethernet operations using SGMII mode. The reference design sends a stream of Ethernet packets to the TSE MegaCore

function. The TSE MegaCore function in turn sends out those packets to the SFP modules and out to the cable where the Ethernet packets are looped back externally via SFP modules with an Ethernet cable assembly or through an Ethernet switch. The reference design can demonstrate the operation of the TSE MegaCore function in various modes with live traffic up to the maximum throughput rate and show the error rate in the receiver, if any.

Reference Design Overview

The reference design provides a general platform that enables you to control, test, and monitor different speeds of Ethernet operations. The reference design is developed using SOPC Builder. [Figure 1](#) shows a high-level block diagram of the reference design.

Figure 1. Reference Design System Block Diagram

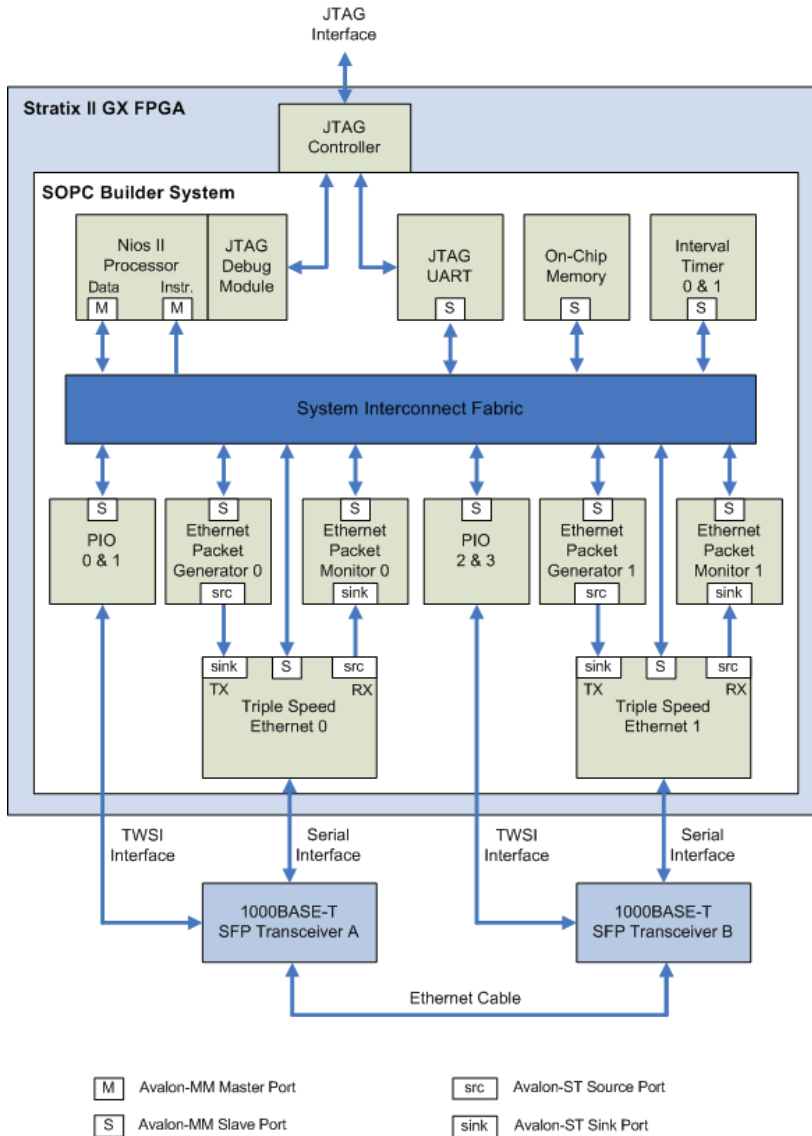


Figure 2 shows the SOPC Builder setup that is used to generate this system.

Figure 2. Reference Design SOPC Builder Set-Up

Clock Settings							
Name	Source			MHz			
clk	External			100.0			
sys_clk	pll.c0			83.333333			

Use	Connections	Module Name	Description	Clock	Base	End	I...
<input checked="" type="checkbox"/>		cpu	Nios II Processor	sys_clk			
		instruction_master	Avalon Master				
		data_master	Avalon Master				
		jtag_debug_module	Avalon Slave				
<input checked="" type="checkbox"/>		onchip_mem	On-Chip Memory (RAM or ROM)	sys_clk	0x00080800	0x00080fff	
		s1	Avalon Slave				
<input checked="" type="checkbox"/>		jtag_uart	JTAG UART	sys_clk	0x00081960	0x00081967	
		avalon_jtag_slave	Avalon Slave				
<input checked="" type="checkbox"/>		pll	PLL	clk	0x00081880	0x0008189f	
		s1	Avalon Slave				
<input checked="" type="checkbox"/>		pio	PIO (Parallel IO)	sys_clk	0x00081920	0x0008192f	
		s1	Avalon Slave				
<input checked="" type="checkbox"/>		pio_1	PIO (Parallel IO)	sys_clk	0x00081930	0x0008193f	
		s1	Avalon Slave				
<input checked="" type="checkbox"/>		pio_2	PIO (Parallel IO)	sys_clk	0x00081940	0x0008194f	
		s1	Avalon Slave				
<input checked="" type="checkbox"/>		pio_3	PIO (Parallel IO)	sys_clk	0x00081950	0x0008195f	
		s1	Avalon Slave				
<input checked="" type="checkbox"/>		eth_gen_inst	Ethernet Generator	sys_clk	0x00081800	0x0008183f	
		avalon_slave_0	Avalon Slave				
	avalon_streaming_sou...	Avalon Streaming Source					
<input checked="" type="checkbox"/>	altera_ethernet	Triple-Speed Ethernet	sys_clk				
	transmit	Avalon Streaming Sink	sys_clk				
	receive	Avalon Streaming Source	sys_clk				
	control_port	Avalon Slave	sys_clk	0x00081000	0x000813ff		
<input checked="" type="checkbox"/>	eth_mon_inst	Ethernet Monitor	sys_clk	0x000818a0	0x000818bf		
	avalon_slave_0	Avalon Slave					
	avalon_streaming_sink	Avalon Streaming Sink					
<input checked="" type="checkbox"/>	eth_gen_inst_1	Ethernet Generator	sys_clk	0x00081840	0x0008187f		
	avalon_slave_0	Avalon Slave					
	avalon_streaming_sou...	Avalon Streaming Source					
<input checked="" type="checkbox"/>	altera_ethernet_1	Triple-Speed Ethernet	sys_clk				
	transmit	Avalon Streaming Sink	sys_clk				
	receive	Avalon Streaming Source	sys_clk				
	control_port	Avalon Slave	sys_clk	0x00081400	0x000817ff		
<input checked="" type="checkbox"/>	eth_mon_inst_1	Ethernet Monitor	sys_clk	0x000818c0	0x000818df		
	avalon_slave_0	Avalon Slave					
	avalon_streaming_sink	Avalon Streaming Sink					
<input checked="" type="checkbox"/>	timer	Interval Timer	sys_clk	0x000818e0	0x000818ff		
	s1	Avalon Slave					
<input checked="" type="checkbox"/>	timer_1	Interval Timer	sys_clk	0x00081900	0x0008191f		
	s1	Avalon Slave					

The following sections provide brief descriptions of each of the components used in the reference design.

Nios II Processor

The Nios II processor is used as a control plane component for setting up and configuring the reference design system components. The processor also begins the Ethernet packet generation and monitors the status of the packet reception.

On-Chip Memory

This reference design uses an on-chip memory block size of 256 Kbytes. The memory is used as an SOPC system RAM for software code storage.

JTAG UART

The JTAG UART core transfers serial character streams between a Nios II processor and an SOPC Builder system. The core provides a simple register-mapped Avalon® interface that hides the complexities of the JTAG interface. The Nios II processor communicates with the core by reading and writing control and data registers.

Phase-Locked Loop

The phase-locked loop (PLL) core takes an input clock from a 100 MHz crystal on the development kit and generates an 83.33 MHz PLL output clock as a system-wide clock source for the SOPC Builder system.

Parallel Input/Output

The parallel input/output (PIO) core provides general-purpose I/O ports that connect to the 1000BASE-T Copper SFP module's Two-Wire Serial Interface (TWSI). The PIO core provides easy I/O access to the 1000BASE-T Copper SFP module's PHY registers using a "bit banging" approach that conforms to the TWSI protocol.

Triple Speed Ethernet MegaCore Function

The TSE MegaCore function provides an integrated Ethernet MAC, PCS, and PMA solution for Ethernet applications. The megafunction transmits the Ethernet packets from the Avalon Streaming (Avalon-ST) interface to a 1.25 Gbps serial transceiver interface that is built in the Stratix II GX device and receives packets from the opposite direction.

Ethernet Packet Generator

The Ethernet Packet Generator block is an SOPC custom component created using the component editor. It has an Avalon Memory-Mapped (Avalon-MM) slave interface on one side for control purposes and an Avalon-ST source interface on the other side for sending Ethernet packets to the TSE MegaCore function. This block drives the TSE MegaCore function Transmit FIFO interface by generating a stream of Ethernet packets into it.

Ethernet Packet Monitor

The Ethernet Packet Monitor block is an SOPC custom component created using the component editor. It has an Avalon-MM slave interface on one side for control purposes and an Avalon-ST sink interface on the other side for the data path. This block is fed a stream of Ethernet packets by the TSE MegaCore function Receive FIFO interface. The Ethernet Packet Monitor also verifies the accuracy of the received payload.

Interval Timer

The interval timer core is a 32-bit timer used by the Nios II processor system to calculate the performance and throughput rate of various Ethernet operations.

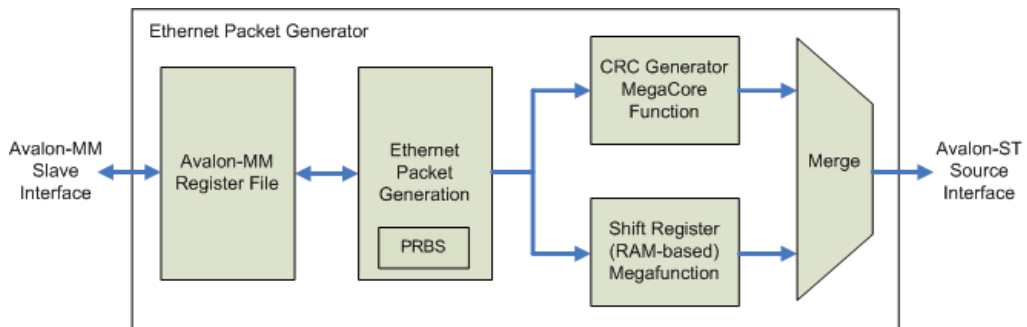
Functional Description

This section describes the operations of the following modules in the TSE reference design, in addition to memory and register maps and signals:

- [“Ethernet Packet Generation”](#)
- [“Ethernet Packet Reception and Verification”](#) on page 9
- [“Interfacing with the Physical Medium”](#) on page 11
- [“Ethernet Packet Loopback Operations”](#) on page 11
- [“Memory Map”](#) on page 13
- [“Register Maps”](#) on page 14
- [“Interface Signal Descriptions”](#) on page 19

Ethernet Packet Generation

The Ethernet packets are generated by the Ethernet Packet Generator block within the reference design. [Figure 3](#) shows a high-level block diagram of the Ethernet Packet Generator block.

Figure 3. Ethernet Packet Generator Block Diagram

This module consists of the following components:

- Avalon-MM Register File
- Ethernet Packet Generation Block
- Altera CRC Compiler Generator MegaCore Function
- Shift Register (RAM-based) Megafunction

The Avalon-MM Register File component provides a register interface for the system to configure all the parameters and settings necessary to start the Ethernet packet generation. Through this register interface, the system can configure programmable settings such as the following:

- The total number of packets to be transmitted
- Incremental or random data type
- Fixed or random packet length
- The source and destination MAC address
- A random seed for the PRBS 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 14](#) for details on the Ethernet Packet Generator register descriptions.

After the system has finished configuring all the registers' settings and the start bit of the operation register is set to 1, the Ethernet Packet Generator starts generating a stream of Ethernet packets to the TSE MegaCore function. Setting this start bit starts the state machine of the Ethernet Packet Generation Block by generating an Ethernet packet header together with a data payload for the number of packets programmed. The start bit also acts as a soft reset to the Ethernet Packet Generator block and resets the state machine along with all the generator's status registers before the generation.

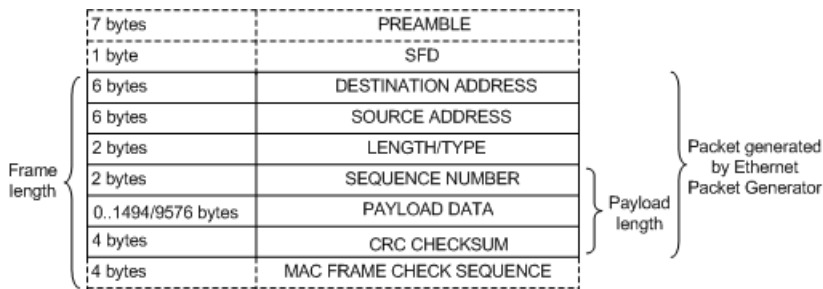
If you want to stop the operation before it is finished, you can issue a write to set the stop bit of the operation register, causing the current running operation to terminate. When the stop bit is asserted, the state machine finishes generating the current packet and then stops the operation.

Every packet generated by the Ethernet Packet Generation Block is then sent to the CRC Generator MegaCore function for checksum calculation and to the RAM-based Shift Register megafunction. The RAM-based Shift Register is used to temporarily store the packet while it waits to be merged with the calculated checksum output by the CRC Generator MegaCore function. After the merging of a valid CRC checksum with the packet stream is done, the complete frame is sent to the Avalon-ST interface.

The main advantage of embedding a CRC checksum into the generated packet is the ease of packet verification on the receive interface, as there is a possibility of an Ethernet packet getting dropped. If a packet is dropped, the system is able to continue verifying the incoming packets based on the embedded CRC checksum without being interrupted. The system is also able to provide the total number of packets received.

The packet that the Ethernet Packet Generator generates is a standard Ethernet packet with the exception of a 7-byte Preamble, 1-byte Start Frame Delimiter (SFD), and 4-byte MAC-calculated Frame Check Sequence (FCS). The generated frame format of the Ethernet Packet Generator is shown in Figure 4.

Figure 4. Ethernet Packet Generation Frame Format



The destination and source MAC addresses are programmable through a register interface. You can set the destination MAC address to be a Unicast or Broadcast address. If Unicast packets are transmitted, the destination MAC address must have the same value as the receiving TSE MAC address of the reference design. The source MAC address must be

programmed with the same value as the transmitting TSE MAC address. This practice ensures the receiving end of the TSE MegaCore function of the reference design always receives the correct packets and can verify the contents.

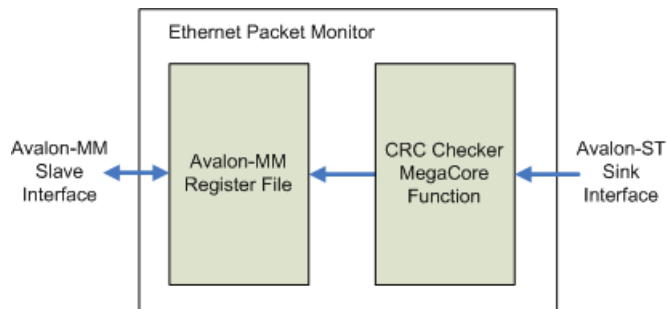
The packet length of each packet generated can be fixed at a programmable value or a random packet size. A programmable fixed packet length can range from 24 to 9600 bytes, while a random packet length can range from 24 to 1518 bytes. Any frame length that is less than 64 bytes is automatically padded by the TSE MegaCore function with padding bytes (0x00) to make it at least 64 bytes long.

The data payload can be either incremental or pseudo-random. If the incremental data type is selected, the payload's data value starts from 0x00 and continues with 0x01, 0x02, and so forth. However, if the pseudo-random data type is selected, the random value generated from the PRBS block is used as the contents of the payload. The random seed used by the PRBS block is programmable via the `rand_seed0` and `rand_seed1` registers in the register file. The PRBS polynomial used is PRBS23 ($2^{23}-1$).

Note that the 2-byte sequence number and 4-byte CRC checksum are part of the packet payload for this reference design. 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 last four bytes of every packet payload. The checksum ensures the data integrity of the packets that start from Ethernet Packet Generator and end in the Ethernet Packet Monitor. The minimum payload length for this reference design is therefore six bytes.

Ethernet Packet Reception and Verification

The Ethernet packets that are sent from the TSE MegaCore function RX FIFO interface are received by the Ethernet Packet Monitor block within the reference design. Their payloads are then verified. [Figure 5](#) shows a block diagram of the Ethernet Packet Monitor.

Figure 5. Ethernet Packet Monitor Block Diagram

This module consists of the following components:

- Avalon-MM Register File
- Altera CRC Compiler Checker MegaCore Function

The Avalon-MM Register File component provides a register interface for the system to configure the settings necessary to start and monitor the Ethernet packet reception. Through this register interface, the system can configure the total number of packets to be received. It also provides the status of the receive operation and a set of statistics counters on the number of good packets received, number of bad packets received, number of bytes and cycles received for performance, and the throughput rate. Refer to [Table 5 on page 16](#) for descriptions of the fields in the Ethernet Packet Monitor register.

After the register interface is configured, the Ethernet Packet Monitor must be enabled before the packets can be received. Setting the Start bit of the `receive_ctrl_status` register resets the packet reception statistics counters and enables the monitor to receive incoming packets.

When the Stop bit of the `receive_ctrl_status` register is asserted, the monitor stops receiving packets and stops updating the statistics counters.

The Ethernet packets that have been looped back to another instance of the TSE MegaCore function in the reference design are streamed directly to the CRC Checker MegaCore function within the Ethernet Packet Monitor module. The module calculates the CRC checksum based on the received packet and verifies the value against the checksum value embedded in the last four bytes of the packet payload. It then outputs a status that identifies whether the packet received is good or corrupted and updates the statistics registers accordingly.

Interfacing with the Physical Medium

The reference design uses two PIO pins to interface with the Two-Wire Serial Interface (TWSI) of Finisar's 1000BASE-T Copper SFP module to configure the registers of its internal PHY device. The TWSI operates with a Serial Data Line (SDA) as a bi-directional line, and with a Serial Clock Line (SCL) as its output line. The software toggles these two PIO pins that conform to the TWSI protocol to communicate with the SFP module PHY registers. The PHY chip of the Finisar's 1000BASE-T Copper SFP module can be accessed via TWSI at slave address 0xAC.



Refer to the *Marvell PHY 88E1111 Datasheet* for more information.

Configuring the PHY registers of the SFP modules enables the interfaces to perform auto-negotiation with each other and get resolved to a common mode of Ethernet operation. Through the CLI, the following modes of Ethernet operations are supported:

- 10BASE-T Full-Duplex
- 10BASE-T Half-Duplex
- 100BASE-T Full-Duplex
- 100BASE-T Half-Duplex
- 1000BASE-T Full-Duplex

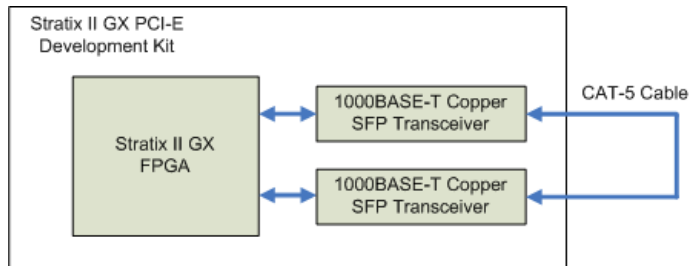
Ethernet Packet Loopback Operations

The reference design can be operated in either of the following options:

- External loopback via 1000BASE-T Copper SFP modules with an Ethernet cable assembly
- External loopback via 1000BASE-T Copper SFP modules with a network switch

External Loopback via 1000BASE-T Copper SFP Modules with Ethernet Cable Assembly

Figure 6 shows a high-level diagram of an external loopback operation using an Ethernet cable assembly.

Figure 6. External Loopback Operation via Ethernet Cable Assembly

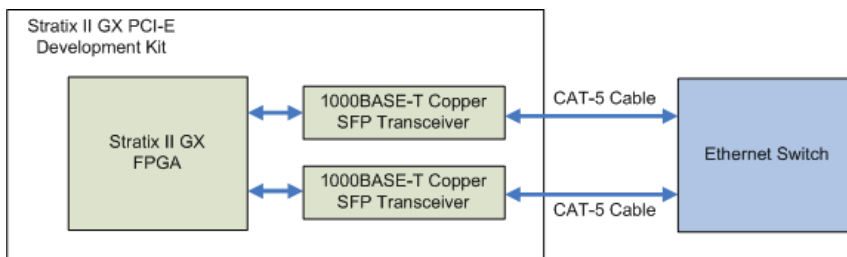
When the reference design operates in external loopback mode using an Ethernet cable assembly, one CAT-5 cable is needed to connect to both 1000BASE-T Copper SFP transceivers. This cable forms a link between the two TSE MegaCore functions in SGMII mode.

When the system is configured, the Ethernet Packet Generator starts generating packets to the transmitting TSE MegaCore function through the TX FIFO interface. Each packet has a four-byte Frame Check Sequence appended by the TSE MAC and then is transmitted out to a 1000BASE-T Copper SFP transceiver. The packets are sent through the Ethernet cable assembly to another 1000BASE-T Copper SFP transceiver reaching the receiving TSE MegaCore function. The packets are then verified with their FCS field. The checksums are then discarded. The remaining packets are streamed out through the RX FIFO interface to the Ethernet Packet Monitor for further packet verification.

When the system has finished receiving all the packets, it reports the statistics of the packets and the throughput rate based on the statistics registers. This is to showcase the real throughput rate that can be achieved by the TSE MegaCore function.

External Loopback via 1000BASE-T Copper SFP Modules with Network Switch

Figure 7 shows a high-level diagram of an external loopback operation using an Ethernet switch and Ethernet cable assembly.

Figure 7. External Loopback Operation via Network Switch

When the reference design operates in this external loopback mode, two CAT-5 cables are needed to connect the 1000BASE-T Copper SFP transceivers to the two ports of the Ethernet Switch. This design forms a link between the two TSE MegaCore functions via an external Ethernet switch.

The operation is exactly the same as the external loopback mode with an Ethernet cable assembly only. The packets are transmitted to the Ethernet switch and the switch routes the packets to the correct port of the receiving TSE MegaCore function that is also connected to it. The switch has self-learning routing capability on the source MAC address of each port connected to it and it routes the packets to the correct port based on the destination MAC address of the packets. Ethernet switches also send extra management packets that are received by the TSE reference design but considered to be extra packets.

Memory Map

Table 1 defines the complete memory mapping of the SOPC system for the reference design.

Table 1. TSE Reference Design Memory Map (Part 1 of 2)	
Address	Description
0x00040000 – 0x0007FFFF	On-Chip Memory - RAM (256 KB)
0x00080800 – 0x00080FFF	CPU JTAG Debug Module
0x00081000 – 0x000813FF	Triple-Speed Ethernet 0
0x00081400 – 0x000817FF	Triple-Speed Ethernet 1
0x00081800 – 0x0008183F	Ethernet Packet Generator 0
0x00081840 – 0x0008187F	Ethernet Packet Generator 1
0x00081880 – 0x0008189F	Phased-Lock Loop

Table 1. TSE Reference Design Memory Map (Part 2 of 2)

Address	Description
0x000818A0 – 0x000818BF	Ethernet Packet Monitor 0
0x000818C0 – 0x000818DF	Ethernet Packet Monitor 1
0x000818E0 – 0x000818FF	Interval Timer 0
0x00081900 – 0x0008191F	Interval Timer 1
0x00081920 – 0x0008192F	PIO 0
0x00081930 – 0x0008193F	PIO 1
0x00081940 – 0x0008194F	PIO 2
0x00081950 – 0x0008195F	PIO 3
0x00081960 – 0x00081967	JTAG UART

Register Maps

Table 2 describes the function of each field of the Ethernet Packet Generator registers.

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

Address Offset	Name	Description	Access	HW Reset
0x00	number_packet	32-bit Number of Packets Register. Total number of packets to be generated by the Ethernet Packet Generator to the TSE MegaCore function.	RW	0x0
0x04	config_setting	16-bit Configuration Setting Register. Refer to Table 3 for bit descriptions.	RW	0x0
0x08	rand_seed0	32-bit Lower Random Seed Register. Used to preload bits 0 to 31 of the PRBS generator when config_setting[15] is set to 1.	RW	0x0
0x0C	rand_seed1	16-bit Upper Random Seed Register. Used to preload bits 32 to 47 of the PRBS generator when config_setting[15] is set to 1. Bits 16 to 31 are reserved.	RW	0x0
0x10	source_addr0	32-bit Lower Source MAC Address Register. Used to define bits 0 to 31 of the source MAC address field of the Ethernet packet. This source MAC address should be programmed with the same MAC address as the transmitting TSE MegaCore function.	RW	0x0

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

Address Offset	Name	Description	Access	HW Reset
0x14	source_addr1	16-bit Upper Source MAC Address Register. Used to define bits 32 to 47 of the source MAC address field of the Ethernet packet. This source MAC address should be programmed with the same MAC address as the transmitting TSE MegaCore function. Bits 16 to 31 are reserved.	RW	0x0
0x18	destination_addr0	32-bit Lower Destination MAC Address Register. Used to define bits 0 to 31 of the destination MAC address field of the Ethernet packet. This destination MAC address should be programmed with the same MAC address as the receiving TSE MegaCore function for Unicast packets.	RW	0x0
0x1C	destination_addr1	16-bit Upper Destination MAC Address Register. Used to define bits 32 to 47 of the destination MAC address field of the Ethernet packet. This destination MAC address should be programmed with the same MAC address as the receiving TSE MegaCore function for Unicast packets. Bits 16 to 31 are reserved.	RW	0x0
0x20	operation	3-bit Operation Register. Refer to Table 4 for bit descriptions.	RW/RO	0x0
0x24	packet_tx_count	32-bit Packet Transmit Count Register. Used to count the number of data packets successfully transmitted by the Ethernet Packet Generator. This register gets cleared to 0 when <code>operation[0]</code> is asserted.	RO	0x0

[Table 3](#) shows the bit descriptions of the `config_setting` register.

Table 3. config_setting Register Bit Descriptions (Part 1 of 2)

Bit(s)	Bit Name	Access	Description
0	length_sel	RW	Packet Length Type Select. 0: Fixed packet length. 1: Random packet length.
14:1	pkt_length	RW	Fixed Packet Length. Programmable packet length ranges from 24–9600. This field is only valid when the Packet Length Type Select bit is set to 0.

Bit(s)	Bit Name	Access	Description
15	pattern_sel	RW	Data Type Select. 0: Incremental data pattern. 1: Random data pattern.
31:16	reserved	—	Reserved bits. Reads return 0.

Table 4 shows the bit descriptions of the operation register.

Bit(s)	Bit Name	Access	Description
0	start	RW	Start Operation. Write 1 to start the packet generation from the Ethernet Packet Generator to the TSE MegaCore function. This bit is automatically cleared after the packet generation begins.
1	stop	RW	Stop Operation. Write 1 to stop the Ethernet Packet Generator from generating and sending further packets to the TSE MegaCore function. The operation finishes sending the current packet when this bit gets asserted and then stops. This bit is cleared when the Start Operation bit is set to 1.
2	tx_done	RO	Transmit Done Status. This bit is set to 1 when the Ethernet Packet Generator finishes transmitting all the packets programmed in the number_packet register. This bit is cleared when the Start Operation bit is set to 1.
31:3	reserved	—	Reserved bits. Reads return 0.

Table 5 describes the function of each field of the Ethernet Packet Monitor registers.

Address Offset	Name	Description	Access	HW Reset
0x00	number_packet	32-bit Number of Packets Register. The total number of packets to be received by the Ethernet Packet Monitor from the TSE MegaCore function.	RW	0x0
0x04	packet_rx_ok	32-bit Packet Received OK Register. Used to count the number of packets received without error.	RO	0x0

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

Address Offset	Name	Description	Access	HW Reset
0x08	packet_rx_error	32-bit Packet Received with Error Register. Used to count the number of packets received with a CRC error.	RO	0x0
0x0C	byte_rx_count0	32-bit Lower Byte Received Count Register. Used to count the number of data bytes received. The register maps to bits 0 to 31 of the 64-bit byte counter. Read the <code>byte_rx_count0</code> register first, followed by the <code>byte_rx_count1</code> register in the next clock cycle while the counter is still incrementing. This procedure avoids reading an incorrect count during packet reception.	RO	0x0
0x10	byte_rx_count1	32-bit Upper Byte Received Count Register. Used to count the number of data bytes received. The register maps to bits 32 to 63 of the 64-bit byte counter. Read the <code>byte_rx_count0</code> register first, followed by the <code>byte_rx_count1</code> register in the next clock cycle while the counter is still incrementing. This procedure avoids reading an incorrect count during packet reception.	RO	0x0
0x14	cycle_rx_count0	32-bit Lower Cycle Receive Count Register. Used to count the number of cycles from the start of the first packet byte received to the end of the last packet byte received. The register maps to bits 0 to 31 of the 64-bit cycle counter.	RO	0x0
0x18	cycle_rx_count1	32-bit Upper Cycle Receive Count Register. Used to count the number of cycles from the start of the first packet byte received to the end of the last packet byte received. The register maps to bits 32 to 63 of the 64-bit cycle counter.	RO	0x0
0x1C	receive_ctrl_status	10-bit Receive Control and Status Register. Refer to Table 6 for bit descriptions.	RW/RO	0x0

Table 6 shows the bit descriptions of the `receive_ctrl_status` register.

Bit(s)	Bit Name	Access	Description
0	<code>start</code>	RW	Start Operation. Write 1 to start the packet reception from the TSE MegaCore function to the Ethernet Packet Monitor. This bit is automatically cleared after the packet reception begins.
1	<code>stop</code>	RW	Stop Operation. Write 1 to stop the Ethernet Packet Monitor from receiving further packets. It immediately stops updating the rest of the status registers when this bit gets asserted. This bit is cleared when Start Operation bit is set to 1.
2	<code>rx_done</code>	RO	Receive Done Status. This bit is set to 1 when the Ethernet Packet Monitor finishes receiving all the packets programmed in the <code>number_packet</code> register. This bit is cleared when the Start Operation bit is set to 1.
3	<code>crcbad</code>	RO	CRC Error Packet. This bit is set to 1 when the current packet being received is detected as a CRC error packet. This status bit is updated continuously for every packet received.
9:4	<code>rx_err</code>	RO	Receive Error Status. The <code>rx_err[5:0]</code> signals from the receive FIFO interface of the TSE MegaCore function are mapped to this field during every final frame octet. Refer to the <i>Triple Speed Ethernet MegaCore Function User Guide</i> for the definitions of every signal.
31:10	reserved	—	Reserved bits. Reads return 0.



Refer to the *Triple Speed Ethernet MegaCore Function User Guide* and the *Marvell PHY 88E1111 Datasheet* for the details of their register maps.

Interface Signal Descriptions

This section describes the interface signals of the reference design application logic.

Table 7 lists the clock and reset signals.

Table 7. Clock and Reset Signals		
Signal Name	Direction	Description
clk	In	Reference design clock. The clock is sourced from the 100 MHz Oscillator (X1).
ref_clk_to_the_altera_ethernet	In	TSE transceiver reference clock. The clock is sourced from the 156.25 MHz Oscillator (X3).
reset_n	In	Single reset for all logic of the reference design. It is connected to the RESET (S4) push button.

Table 8 lists the triple-speed Ethernet 0 signals.

Table 8. Triple-Speed Ethernet 0 Signals		
Signal Name	Direction	Description
rxp_to_the_altera_ethernet	In	Serial Differential Receive Interface. It is connected to the sfpa_rx_p0 signal of SFP A (J6).
txp_from_the_altera_ethernet	Out	Serial Differential Transmit Interface. It is connected to the sfpa_tx_p0 signal of SFP A (J6).
led_an_from_the_altera_ethernet	Out	Auto-negotiation status. It is connected to USER_LED0 (D16).
led_link_from_the_altera_ethernet	Out	Link synchronization status. It is connected to USER_LED2 (D14).
led_col_from_the_altera_ethernet	Out	Collision detection of frame transmission. It is connected to USER_LED4 (D12).
led_crs_from_the_altera_ethernet	Out	Carrier sense on transmit and receive path. It is connected to USER_LED6 (D10).

Table 9 lists the triple-speed Ethernet 1 signals.

Signal Name	Direction	Description
rxp_to_the_altera_ethernet_1	In	Serial Differential Receive Interface. It is connected to the <code>sfpb_rx_p0</code> signal of SFP B (J7).
txp_from_the_altera_ethernet_1	Out	Serial Differential Transmit Interface. It is connected to the <code>sfpb_tx_p0</code> signal of SFP B (J7).
led_an_from_the_altera_ethernet_1	Out	Auto-negotiation status. It is connected to USER_LED1 (D15).
led_link_from_the_altera_ethernet_1	Out	Link synchronization status. It is connected to USER_LED3 (D13).
led_col_from_the_altera_ethernet_1	Out	Collision detection of frame transmission. It is connected to USER_LED5 (D11).
led_crs_from_the_altera_ethernet_1	Out	Carrier sense on transmit and receive path. It is connected to USER_LED7 (D9).

Table 10 lists the SFP interface signals.

Signal Name	Direction	Description
out_port_from_the_pio	Out	SFP A Serial Clock Line. It is connected to the <code>sfpa_mod1_scl</code> signal of SFP A (J6).
bidir_port_to_and_from_the_pio_1	In/Out	SFP A Serial Data Line. It is connected to the <code>sfpa_mod2_sda</code> signal of SFP A (J6).
out_port_from_the_pio_2	Out	SFP B Serial Clock Line. It is connected to the <code>sfpb_mod1_scl</code> signal of SFP B (J7).
bidir_port_to_and_from_the_pio_3	In/Out	SFP B Serial Data Line. It is connected to the <code>sfpb_mod2_sda</code> signal of SFP B (J7).
sfpa_txdisable	Out	SFP A Transmitter Disable. It is connected to the <code>sfpa_txdisable</code> signal of SFP A (J6).
sfpb_txdisable	Out	SFP B Transmitter Disable. It is connected to the <code>sfpb_txdisable</code> signal of SFP B (J7).

Using the TSE Reference Design

The TSE reference design contains both hardware and software components.

Prerequisites

You must have the following hardware to run this design:

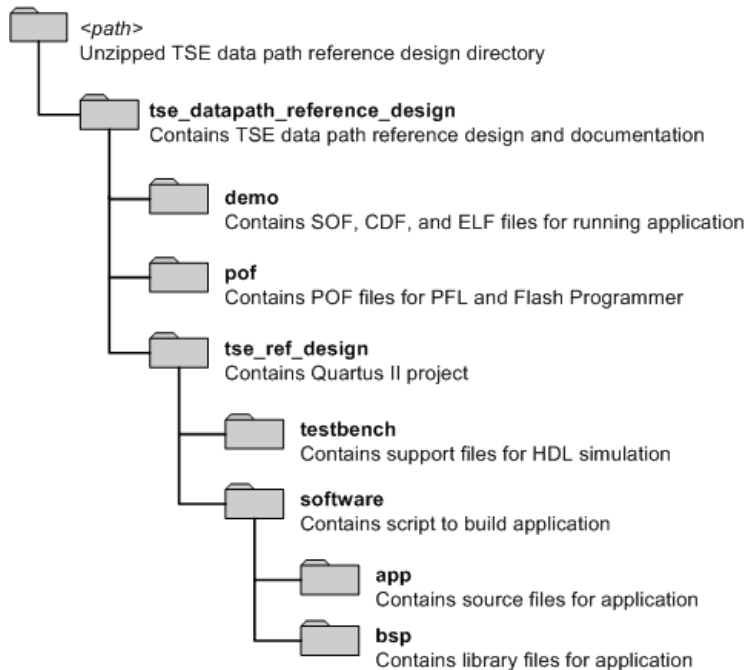
- Stratix II GX PCI Express development board
- Altera programming cable (ByteBlaster or USB-Blaster)
- 2x Finisar Active Copper SFP (FCMJ-8521-3) Modules
- Straight-through CAT5 Ethernet cable assembly

Additionally, you must have the following software installed on your workstation:

- Quartus® II software version 7.2
- Nios II EDS version 7.2

Directory Structure

Figure 8 shows the directory structure after you unzip the reference design (`tse_datapath_reference_design.zip`) to a *<path>* directory.

Figure 8. Directory Structure

Hardware Setup

This section describes how to set up the necessary hardware, including the development board and the Ethernet configuration.

Board Setup

Perform the following steps to set up your development board:

1. Insert the Finisar 1000BASE-T Copper SFP modules into the SFP_A (J6) and SFP_B (J7) slots on the development board. Ensure that the modules “click” into place.
2. Connect both ports of the Finisar PHY modules using either a “point-to-point” connection, or through a hub or switch. Refer to [“Ethernet Configuration”](#) to determine which connection scheme you want to use.
3. Connect the Altera programming cable to the JTAG connection port (J5).

4. Connect the Stratix II GX development board's power adapter to the board (J3).
5. Toggle the power switch (SW1) on the development board to supply the board with power.

Ethernet Configuration

You must choose a network configuration to use for your test setup, either "point-to-point" or through a switch or hub. In a "point-to-point" configuration, you connect the Ethernet ports directly to one another through a CAT5 Ethernet cable assembly. This connection configuration is recommended for your first use because it is simple, robust, and has the least likelihood of errors.

However, if you decide to connect both ports through a switch or hub, set the following parameters to ensure that your test is successful:

- **Ethernet Speed**—Ensure that your switch or hub supports the network speed at which you want to conduct your test
- **Network Topology**—If you are connecting to an active network (LAN), ensure that the MAC addresses you assign the TSE MACs do not conflict with other MAC addresses found on the network (refer to ["TSE MAC Menu" on page 31](#))
- **Network Traffic**—If you are connecting to an active network (LAN), be aware that this reference design has the potential to flood the network with a huge amount of traffic, disrupting the normal network function

You must decide which network connection methodology you want to use and ensure that both Ethernet ports of the Finisar PHY modules are connected appropriately.

Configuring the Board with the Hardware Design

The next step is 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 instance. On a Windows workstation, on the **Start** menu, point to **Programs**, point to **Altera**, point to **Nios II EDS <version>**, and click **Nios II <version> Command Shell**.
2. In the Nios II Command Shell, change directories to the `<path>/tse_datapath_reference_design/demo/` folder where the files `tse_ref_design_top.cdf` and `tse_ref_design_top.sof` are located.

- On the command line, enter the following:

```
> quartus_pgm tse_ref_design_top.cdf ←
```



If you have more than one programming cable installed in the PC, use the `--list` option to display a list of available hardware. Specify the correct programming cable by adding the `--cable=< cable number >` option. For example:

```
quartus_pgm --cable=1 tse_ref_design_top.cdf
```

Include the `-h` option for more information about the Quartus II Programmer. For example: `quartus_pgm -h`

- If the image is successfully programmed, the information shown in [Figure 9](#) appears in the Command Shell.

Figure 9. Nios II Command Shell

```

C:\ SOPC Builder 7.2
/cygdrive/d
[ISOPC Builder]$ cd tse_standalone_demo_reference_design/
/cygdrive/d/tse_standalone_demo_reference_design
[ISOPC Builder]$ cd demo/
/cygdrive/d/tse_standalone_demo_reference_design/demo
[ISOPC Builder]$ quartus_pgm tse_ref_design_top.cdf
Info: *****
Info: Running Quartus II Programmer
Info: Version 7.2 Build 151 09/26/2007 SJ Full Version
Info: Copyright (C) 1991-2007 Altera Corporation. All rights reserved.
Info: Your use of Altera Corporation's design tools, logic functions
Info: and other software and tools, and its AMPP partner logic
Info: functions, and any output files from any of the foregoing
Info: (including device programming or simulation files), and any
Info: associated documentation or information are expressly subject
Info: to the terms and conditions of the Altera Program License
Info: Subscription Agreement, Altera MegaCore Function License
Info: Agreement, or other applicable license agreement, including,
Info: without limitation, that your use is for the sole purpose of
Info: programming logic devices manufactured by Altera and sold by
Info: Altera or its authorized distributors. Please refer to the
Info: applicable agreement for further details.
Info: Processing started: Thu Nov 01 16:41:08 2007
Info: Command: quartus_pgm tse_ref_design_top.cdf
Info: Using programming cable "USB-Blaster [USB-0]"
Info: Started Programmer operation at Thu Nov 01 16:41:13 2007
Info: Configuring device index 2
Info: Device 2 contains JTAG ID code 0x020E30DD
Info: Configuration succeeded -- 1 device(s) configured
Info: Successfully performed operation(s)
Info: Ended Programmer operation at Thu Nov 01 16:41:20 2007
Info: Quartus II Programmer was successful. 0 errors, 0 warnings
Info: Allocated 163 megabytes of memory during processing
Info: Processing ended: Thu Nov 01 16:41:20 2007
Info: Elapsed time: 00:00:12
/cygdrive/d/tse_standalone_demo_reference_design/demo
[ISOPC Builder]$

```

5. If both Finisar 1000BASE-T Copper SFP modules are already plugged into the SFP cage of the development board, USER_LED D13 and USER_LED D14 light up, indicating the link is synchronized.
6. USER_LED D15 and USER_LED D16 light up slightly later, when auto-negotiation is completed and the link is established.

Optionally, you can program a POF image into the flash memory, which then loads the programmed content into the FPGA when the board powers up. For more information, refer to [“Programming with POF” on page 39](#).

You have now successfully configured the board with the hardware image.

Software Setup

After the hardware setup is complete, you can set up the software, as described in the following sections.

Download and Run the Benchmark Application

The next step is to download the pre-compiled benchmark application to the system memory and start the Nios II processor’s operation. Perform the following steps:

1. Continue with the Nios II Command Shell. The file **main.elf** is located in the same directory, `<path>/tse_datapath_reference_design/demo/`.
2. On the command line, enter the following:

```
> nios2-download -g main.elf; nios2-terminal ↵
```

This command first downloads the Nios II processor’s image file to system memory and then restarts the processor to begin code execution. Additionally, a terminal application begins running, which lets you interact with the software application.



If you have more than one programming cable installed in the PC, use the `--list` option to display a list of the available hardware. Specify the correct programming cable by adding the `--cable=<cable number>` option to the command. For example:

```
nios2-download --cable=1 -g main.elf; nios2-terminal --cable=1
```

Include the `-h` option for more information about the Nios II downloader and terminal. For example:

```
nios2-download -h; nios2-terminal -h
```

Figure 10 shows the information that appears in the command shell.

Figure 10. Benchmark Application

```

SOPC Builder 7.2
/cygdrive/d/tse_standalone_demo_reference_design/demo
[SOPC Builder]$ nios2-download -g main.elf; nios2-terminal
Using cable "USB-Blaster [USB-0]", device 2, instance 0x00
Pausing target processor: OK
Initializing CPU cache <if present>
OK
Downloaded 87KB in 1.0s (87.0KB/s)
Verified OK
Starting processor at address 0x000401C8
nios2-terminal: connected to hardware target using JTAG UART on cable
nios2-terminal: "USB-Blaster [USB-0]", device 2, instance 0
nios2-terminal: <Use the IDE stop button or Ctrl-C to terminate>

*****
*                               Test Menu                               *
*****
1) Link Speed - PHY link speed (Mbits/sec)
   Value: 1000
   a) 10 b) 100 c) 1000

2) Link Configuration- Duplex rate for link
   <Only full for 1000 Mbit link>
   Value: full
   a) half b) full

3) Packet Length - Size in bytes (64 < x < 9600 or random)
   Value: 1518
   a) 64 b) 1518 c) 9600 d) random e) <user specified>

4) Number Packets - Packets to send (0 < x < 4294967296)
   Value: 1000000 <1 < value < 2^32>
   a) 1000000 b) 10000000 c) 100000000 d) <user specified>

5) Packet Data - Controls type of data sent for test
   Value: increment
   a) increment b) random

6) Test Control - Starts the test
   a) start (stops on <Enter> during test)

7) Other Menus - Switch to Other Menus
   a) Test Menu b) TSE MAC Menu c) Report Menu d) Console Menu

Command>

```

When the information in Figure 10 appears, you have successfully started the benchmark application.

Quick Software Tutorial — Interacting with the System

The benchmark application provides a menu-driven interface to configure, run, and analyze tests. The menu is shown in [Figure 11](#).

Figure 11. Application Test Menu

```

SOPC Builder 7.2
*****
*                               Test Menu                               *
*****
1) Link Speed - PHY link speed <Mbits/sec>
   Value: 100
   a) 10 b) 100 c) 1000

2) Link Configuration- Duplex rate for link
   <Only full for 1000 Mbit link>
   Value: half
   a) half b) full

3) Packet Length - Size in bytes <64 < x < 9600 or random>
   Value: 1000
   a) 64 b) 1518 c) 9600 d) random e) <user specified>

4) Number Packets - Packets to send <0 < x < 4294967296>
   Value: 1000000 <1 < value < 2^32>
   a) 1000000 b) 10000000 c) 100000000 d) <user specified>

5) Packet Data - Controls type of data sent for test
   Value: increment
   a) increment b) random

6) Test Control - Starts the test
   a) start <stops on <Enter> during test>

7) Other Menus - Switch to Other Menus
   a) Test Menu b) TSE MAC Menu c) Report Menu d) Console Menu

Command>6a_

```

This section provides a brief tutorial on how to use the menu system by running a simple test.

In this tutorial, you configure the system to use a 100 Mbit link (half duplex). The test is configured to send 100,000 packets with a length of 1000 bytes each. To configure and run this test, perform the following steps:

1. Configure the link speed to be 100 Mbit/sec. On the command line, enter the following:

```
> 1b ←
```

The command is processed and the **Link Speed Value** field is set to 100. For all subsequent commands, verify that the reported Value fields change accordingly.

2. Configure the link duplex to be half. On the command line, enter the following:

```
> 2a ↵
```

3. Configure the packet size to be 1000 bytes. On the command line, enter the following:

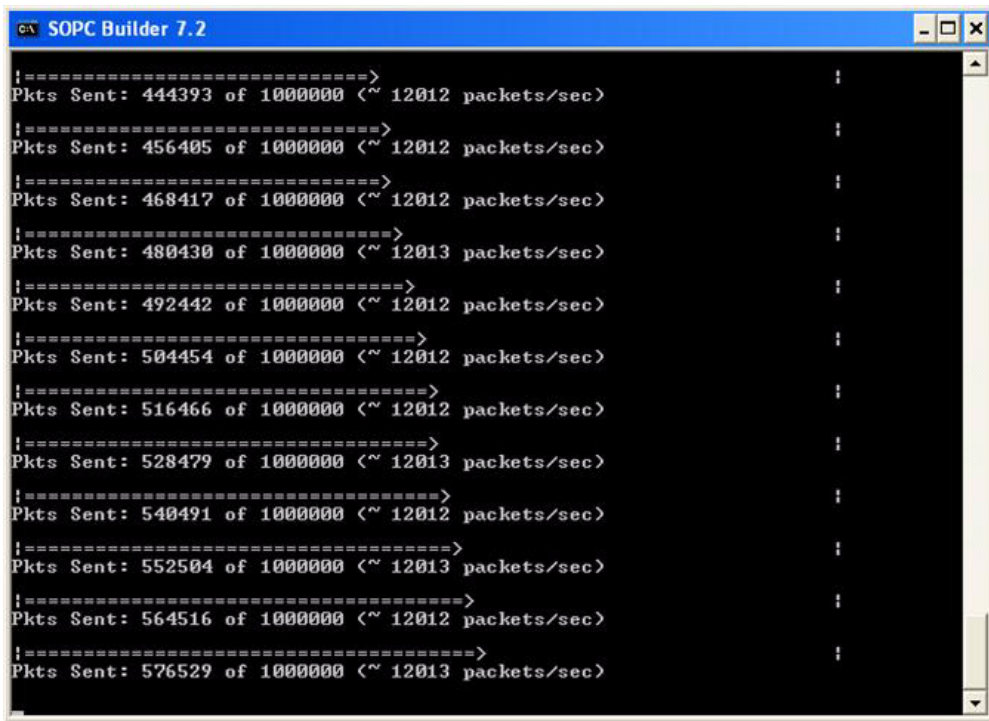
```
> 3e 1000 ↵
```

4. The system is now configured. Run the test by entering the following on the command line:

```
> 6a ↵
```

As the test executes, a series of status messages appear in the console window (Figure 12).

Figure 12. Test Status Messages



```
cx SOPC Builder 7.2
|=====>
Pkts Sent: 444393 of 1000000 (<~ 12012 packets/sec)
|=====>
Pkts Sent: 456405 of 1000000 (<~ 12012 packets/sec)
|=====>
Pkts Sent: 468417 of 1000000 (<~ 12012 packets/sec)
|=====>
Pkts Sent: 480430 of 1000000 (<~ 12013 packets/sec)
|=====>
Pkts Sent: 492442 of 1000000 (<~ 12012 packets/sec)
|=====>
Pkts Sent: 504454 of 1000000 (<~ 12012 packets/sec)
|=====>
Pkts Sent: 516466 of 1000000 (<~ 12012 packets/sec)
|=====>
Pkts Sent: 528479 of 1000000 (<~ 12013 packets/sec)
|=====>
Pkts Sent: 540491 of 1000000 (<~ 12012 packets/sec)
|=====>
Pkts Sent: 552504 of 1000000 (<~ 12013 packets/sec)
|=====>
Pkts Sent: 564516 of 1000000 (<~ 12012 packets/sec)
|=====>
Pkts Sent: 576529 of 1000000 (<~ 12013 packets/sec)
```

These messages show the progress of the test. The arrow above each line represents how many packets have been sent, with the right margin indicating completion of the test. A throughput rate calculation is also printed, providing a rough approximation of the packets/second rate achieved. Throughout the test, both USER_LED D9 and USER_LED D10 should light up, indicating the activity of the link. When the test is complete, the command shell displays the Report menu (Figure 13). If you want to run a new test after this test is complete, change to the Test menu by entering 2a at the command prompt. From the Test menu you can either rerun the same test configuration by entering 6a at the command prompt or reconfigure new settings before starting the new test.

Figure 13. Report Menu

```

SOPC Builder 7.2
*****
*                               Report Menu                               *
*****
-----Test-----
Number: 1
Status: Done (no errors)
Run Time: 30.061 seconds
-----Link Configuration-----
TSE Sender MAC: 0xes1122334450
TSE Receiver MAC: 0xee2233445560
Link Speed: 100 (Duplex: half)
-----Sender-----
Packets to Send: 1000000 packets
(length: 1000, data type: increment)
Packets Sent: 1000000 packets (33266.085 packets/sec)
-----Receiver-----
Packets Received: 1000000 (valid: 1000000, error: 0)
Bytes Received: 1000000000 bytes (rate: 266128676.368 bits/second)

1) Report Control - Prints test results.
   (Either "prev" and "next" to scroll through tests
   or enter in a test number.)
   Reports: 1
   a) prev b) next c) <user specified>

2) Other Menus - Switch to Other Menus
   a) Test Menu b) TSE MAC Menu c) Report Menu d) Console Menu

Command>

```

TSE Software Description

The reference design uses a Nios II processor to run a software application that configures and runs benchmark tests for the system. This section describes the functionality of this application.

Menu-Driven Interface

The application provides a menu-driven user interface, with which you can parameterize and run tests on the hardware. Options are selected in the menu interface by selecting the menu item, followed by an option (if present), and pressing **Enter**. Some menu options also allow you to specify a value for the selected option. For user-specified options that require a numerical value, the value may be specified as a hexadecimal value (for example, 0x1000) or a decimal value (for example, 4000).

There is a default value for all the test parameters. Generally, you can start a default test by entering 6a at the command prompt without changing any settings.

The following sections describe the available menus.

Test Menu

The Test menu is used to configure and run a benchmark test in the system (refer to [Figure 11](#)). The following test parameters can be controlled:

- **Link Speed**—Specifies the Ethernet link's speed. It can be set to 10, 100, or 1000 Mbits/sec.
- **Link Configuration**—Specifies the duplex mode for the link. The mode can be set to either full or half duplex. (The 1000 Mbits/sec link only supports full duplex.)
- **Packet Length**—Specifies the length of the packets sent over the link. The length can be set to either a fixed or random size. When specifying fixed packet lengths, the acceptable range is from 24 bytes to 9600 bytes. When the random option is selected, generated packets range in length from 24 bytes to 1518 bytes.
- **Number of Packets**—Specifies the number of packets to send for the test. The number is a 32-bit, unsigned integer. The valid range is from 1 to 2^{32} packets.
- **Packet Data**—Specifies the data type sent in the payload portion of the packet. The type can be either increment or random. When the increment type is specified, the data payload consists of a 32-bit, unsigned integer, which is incremented in every packet. When this integer reaches 2^{32} , it resets to 0. Selecting the random data type results in a pseudo-random, 32-bit unsigned integer ($2^{32}-1$) being used at the data payload. The pseudo-random integer is generated using the **Seed value** menu option.

- **Seed Value**—The **Seed value** menu option only appears when the **Packet Data** menu option is set to random. It specifies the generation of the random integer. The seed value is a 46-bit, hexadecimal number (any value exceeding this length is truncated).
- **Test Control**—Controls starting the test. After starting, the test runs until either the number of packets sent is equal to the value specified in the **Number of Packets** parameter, or you manually interrupt the test by pressing **Enter**. After the test has halted, the view changes to the **Report Menu**, enabling you to view the results. During the operation of a test, a throughput approximation, printed out as packets/sec, is displayed.

TSE MAC Menu

The TSE MAC menu is used to configure the MAC address of both the sender and receiver TSE MACs (Figure 14).

- **TSE Sender MAC Address**—Specifies the MAC address for the sender. The MAC address is a 12-digit, hexadecimal number. The default TSE Sender MAC address used in the reference design is 0xEE1122334450. This MAC address can be configured to any Unicast or Multicast address for test purposes as long as it does not conflict with any other address on the network, if it is connected to one through the network switch.
- **TSE Receiver MAC Address**—Specifies the MAC address for the receiver. The MAC address is a 12-digit, hexadecimal number. The default TSE Receiver MAC address used in the reference design is 0xEE2233445560. This MAC address can be configured to any Unicast or Multicast address for test purposes as long as it does not conflict with any other address on the network, if it is connected to one through the network switch.

Figure 14. Application TSE MAC Menu

```

SOPC Builder 7.2
*****
*                               TSE MAC Menu                               *
*****
1) TSE Sender MAC Address:
   Value: 0xee1122334450 <hexadecimal>
   a) 0xEE1122334450 b) <user specified>
2) TSE Receiver MAC Address:
   Value: 0xee2233445560 <hexadecimal>
   a) 0xEE2233445560 b) <user specified>
3) Other Menu - Switch to Other Menu
   a) Test Menu b) TSE MAC Menu c) Report Menu d) Console Menu
Command>

```

Report Menu

The Report menu displays the status of the tests run on the system. Conceptually, the Report menu system provides you with ten slots in which to store reports. Reports are stored in first-in, first-out (FIFO) format. When the tenth report slot has been used, the oldest report data is discarded to make room for new reports.

Figure 15 displays the Report menu.

Figure 15. Report Menu

```

SOPC Builder 7.2
*****
*                               Report Menu                               *
*****
-----Test-----
Number: 1
Status: Done (no errors)
Run Time: 12.304 seconds
-----Link Configuration-----
TSE Sender MAC: 0xee1122334455
TSE Receiver MAC: 0xee2233445566
Link Speed: 1000 (Duplex: full)
-----Sender-----
Packets to Send: 1000000 packets
(length: 1518, data type: random, seed value: 0x291454e59c61)
Packets Sent: 1000000 packets (81274.132 packets/sec)
-----Receiver-----
Packets Received: 1000000 (valid: 1000000, error: 0)
Bytes Received: 1518000000 bytes (rate: 986993059.140 bits/second)

1) Report Control - Prints test results.
   (Either "prev" and "next" to scroll through tests
    or enter in a test number.)
   Reports: 1
   a) prev b) next c) <user specified>

2) Other Menus - Switch to Other Menus
   a) Test Menu b) TSE MAC Menu c) Report Menu d) Console Menu

Command>

```

The Report menu provides the following information:

- **Number**—An integer that corresponds to the completed test. The first test is number 1 and the value increments with each new test started.
- **Status**—Indicates the status of the test: **Completed**, **Interrupted**, or **Error**. (Any additional status information is printed, if available.)
- **TSE Sender MAC**—The MAC address of the TSE MAC sender.
- **TSE Receiver MAC**—The MAC address of the TSE MAC receiver.
- **Link Speed**—Displays the Ethernet link speed (10, 100, or 1000 Mbits/sec), along with the duplex mode (**half** or **full**).
- **Packets to Send**—Displays the number of packets the test was configured to send. The packet length and the data type are also displayed. If the data type is **random**, the seed value is displayed.

- **Packets Sent**—Displays the total number of packets actually sent by the Ethernet Packet Generator during the test. The actual value may be less than requested, if there was an error in the system. A throughput measurement for the packets sent is also provided in a packets per second format.
- **Packets Received**—Displays the total number of packets received by the Ethernet Packet Monitor during the test. The number of valid packets and error packets are also displayed. (The total number of packets should be equal to the number of valid packets plus the number of error packets received.)
- **Bytes Received**—Displays the total number of bytes received by the Ethernet Packet Monitor. The byte count includes the complete packet lengths (header plus data) in its calculation. A byte throughput rate is also displayed in terms of bits/second.



The total amount of time taken to complete the test is based on a cycle count register found in the Ethernet Packet Monitor. This cycle count register is set to 0 at the start of the test and begins incrementing at the system clock frequency rate after the first packet has been received. The cycle count register automatically stops counting when all packets have been received by the monitor, or if the test is manually interrupted. The time is computed by dividing the cycle count register value by the system clock frequency, using floating point operations.

While this method for calculating time is very accurate, it is dependent on the monitor receiving the first and last packets during the test to start and stop the cycle count register. If these events do not happen, or if a major error occurs in the Ethernet Packet Monitor, the reported time values are inaccurate.

- **Report Control**—Controls the selection of the report. The current test number being viewed is displayed, along with the first and last report numbers accessible. You can view the reports via the **next** and **previous** controls, or by entering in a report number.

Console Menu

The Console menu provides a console-based interface to the system. All of the commands and functionality provided via the Test, TSE MAC, and Report menus are available via the console interface. You can also read and write arbitrary memory locations via the console.

Figure 16 shows the Console menu.

Figure 16. Console Menu

```

SOPC Builder 7.2
***** Console Menu *****
***** The console mode allows you to use the command line to enter commands. *****
1) Console
a) Enter console mode
2) Other Menus - Switch to Other Menus
a) Test Menu b) TSE MAC Menu c) Report Menu d) Console Menu
Command>

```

The Console menu provides the following menu item:

- **Console**—Controls the launching of the console application. To exit the console operation and return to the menu system, press the x key and then press **Enter**.

Console Operation

The console operates in a command-line driven mode, with commands terminated by pressing **Enter**. Most commands also accept additional arguments, which are delimited by white space.

The console recognizes the commands listed in Table 11. Most of the commands mirror the functionality found in the menu system, providing you with an alternate interface to conduct throughput tests. More information about a particular command can be obtained by entering the command name, followed by the '?' key (with no white space in between).

Table 11. Console Commands (Part 1 of 2)

Command	Purpose	Menu Equivalent	
		Menu	Location
speed	link speed	Test	Link Speed
duplex	link duplex	Test	Link Configuration
plen	packet length	Test	Packet Length
pnum	number of packets	Test	Number of Packets
pdata	packet data	Test	Packet Data
seed	seed value	Test	Seed Value
start	starts test	Test	Start Test
macaddr_send	Sender MAC address	TSE MAC	TSE Sender MAC Address

Table 11. Console Commands (Part 2 of 2)

Command	Purpose	Menu Equivalent	
		Menu	Location
macaddr_recv	Receiver MAC address	TSE MAC	TSE Receiver MAC Address
report	displays report	Report	—
reportn	queues report	Report	Report Control
oreg	read / write memory location (with offset)	—	—
reg	read / write memory location	—	—
test	list test parameters	Report	—
map	display memory map	—	—
?	command information	—	—
x	exits console	—	—

Appendix: Running the Software Application

To compile the software application, you must install the Nios II Processor Embedded Development Suite, version 7.2.

The software application has been written in the C programming language, and can be easily built through a standard Makefile script. The Nios II software development environment includes a set of utilities that let you build software applications quickly and easily.

This application includes a bash shell script that uses the Nios II software development utilities. When the script is run, a set of Makefiles for generating the application is created.



For more information about the Nios II CLI or software build flow, refer to the *Altera-Provided Development Tools* chapter of the *Nios II Software Developer's Handbook* or the software tutorial in the *Nios II IDE Help System*.

Building the Software Application

This section describes how to create the standard Makefiles for the software application: one for the system library and one for the main application. After these Makefiles are created, you can compile the software application and run it on the target hardware (described in “*Hardware Setup*” on page 22).

To create the Makefiles, perform the following steps:

1. Open the **Nios II SDK Command Shell**. On a Windows workstation, on the Start menu, point to **Programs**, point to **Altera**, point to **Nios II EDS <version>**, and click **Nios II <version> Command Shell**.
2. Change directories to the folder where you unzipped the **tse_datapath_reference_design.zip** project files and open the **software** directory.
3. In the **software** directory, enter the following command:

```
> sh create-software ↵
```

As this script creates a system library and the application, a series of messages are sent to the console.

4. To build the system library for the project, first change directories to the **bsp** directory. On the command line, enter the following:

```
> make ↵
```

This command runs the system library's Makefile, resulting in a compiled system library.

5. To build the application, change directories to the **app** directory. On the command line, enter the following:

```
> make ↵
```

This command creates an executable ELF file that runs the application.

You have completed compiling the application. You can now run the application on your target hardware.

Application Structure

The application has been structured in a semi-hierarchical manner, delineated along major function pieces.

Menu/Console

You interact with the application through the menu or console interfaces. At the beginning of the application's execution, the **main** entry point creates four menus that you can interact with:

- Test Menu (**test_menu.c**)
- Report Menu (**report_menu.c**)
- TSE MAC Menu (**tsemac_menu.c**)
- Console Menu (**console_menu.c**)

Each of these menus uses the simple menuing library provided through **elem.c**. These menus enable you to operate the test controls in an intuitive manner.

Test Controls

Configuring, starting, and stopping tests are done through the test control API (declared in **test_control.h**), by calling the **tseStartTest**, **tsePollTest**, and **tseStopTest** functions. Before starting a test, all parameters for the test are specified via the structure **struct gTestStruct**. When this structure has been filled out, it can be used to start a test via the **tseStartTest** function. The status of the test is polled for with **tsePollTest**, with status information passed back through the **struct gTestStruct** structure. To stop a test, the **tseStopTest** function is called.

The test control functions, in turn, make direct calls to the peripheral control functions, described in the following section.

Monitor/Generator/TSE MAC/PHY Peripherals

The main hardware components in this system are the Monitor, Generator, TSE MAC, and PHY components. Control of these peripherals primarily occurs in **tse_control.c**. The register macros and some hardware information are found in **hardware_def.h**.

Additionally, software control of the PHY components occurs through a simple TWSI API found in **alt_2_wire.c**. This API provides a "bit-bang" implementation of a TWSI master peripheral via a PIO peripheral.

Source File Layout

Table 12 lists the source files used in the application, along with their functionality.

Table 12. Test Controls	
Source File	Purpose
test_control.h	Test control routines header
test_control.c	Test control routines
helpers.h	Conversion library header
helpers.c	Conversion library routines
Menu / Console	
main.c	Entry point to software
test_menu.c	Test menuing element
console_menu.c	Console menuing element
report_menu.c	Report menuing element
tsemac_menu.c	TSE MAC menuing element
elem.h	Menuing library header
elem.c	Menuing library
bool_check.h	Defines TRUE and FALSE
Monitor / Generator / TSE MAC / PHY Peripherals	
hardware_def.h	Component information
alt_2_wire.h	TWSI library
alt_2_wire.c	TWSI library
tse_control.h	Hardware control header
tse_control.c	Hardware control routines

Testbench Simulation

You can simulate the reference design using the IP functional simulation model and testbench provided. The testbench files are provided in the **testbench** subdirectory of the project directory. You can use any supported simulator for this purpose. The testbench provides a script for the ModelSim simulator.

Simulating with the ModelSim Simulator

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

1. Start the ModelSim simulator.

2. Change the working directory to `<path>/tse_datapath_reference_design/tse_ref_design/testbench/`.
3. Enter the following command to set up the required libraries, compile the provided IP functional simulation model, and exercise the simulation model with the provided testbench:

```
> do run_tse_ref_design_tb.tcl ←
```

The ModelSim transcript pane (in the Main window) displays messages from the testbench reflecting the current task being performed.

Programming with POF

The reference design package includes a POF file in the `<path>/tse_datapath_reference_design/pof/` folder. However, if needed, a POF can be generated from an SOF to configure the flash on the development board.



Ensure that the SOF has been verified before converting it to a POF. This requires loading the SOF into the FPGA and running tests to confirm that the loaded configuration operates properly.

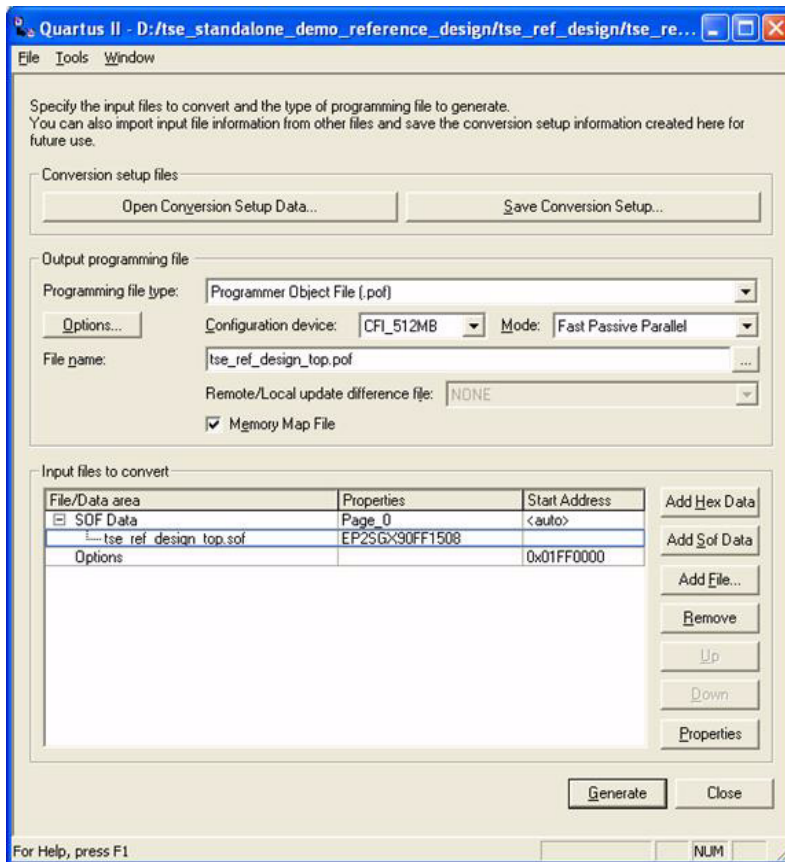
Programming the MAX[®] II flash memory with a POF involves the following activities:

- Generate a POF by converting an SOF
- Program the flash memory

Perform the following steps to generate a POF:

1. Start the Quartus II software.
2. On the File menu, click **Convert Programming Files**.
3. Enter the following parameters (Figure 17):
 - Programming file type: **Programmer Object File (.pof)**
 - Configuration device: **CFL_512MB**
 - Mode: **Fast Passive Parallel**
 - Options: set Configuration device JTAG user code to **0x1FF0000**
 - File name: `<output file name>.pof`
4. Click **Add File** and select SOF file to add.
5. Click **Generate**.

Figure 17. Convert Programming Files Window



Perform the following steps to program the flash memory:

1. Ensure that the JTAG (the USB-Blaster cable) is connected to the development board and is enabled.
2. Identify the MAX II device in the programmer window.
3. Double-click on the MAX II device (EPM570GT100) and select the PFL-compliant MAX design (**altera_siigx_pcie_pfl.pof**), located in the `<path>/tse_datapath_reference_design/pof` directory (Figure 18).

4. Double-click on the Flash device (CFI_512MB) and select the supplied POF (**tse_ref_design_top.pof**) in the same directory or the newly generated flash image POF.
5. Turn on **Program/Configure** for the EPM570GT100 and CFI_512MB devices.

Figure 18. Programming Flash Window

Start	File	Device	Checksum	Usercode	Program/Configure	Verify
Stop	D:/tse_standalone_demo_reference_design/pol/altera_sigx_pcie_pfl.pof	EPM570GT100	002E5272	FFFFFFF	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Auto Detect	-CFM				<input checked="" type="checkbox"/>	<input type="checkbox"/>
Delete	-UFM				<input checked="" type="checkbox"/>	<input type="checkbox"/>
Add File...	D:/tse_standalone_demo_reference_design/pol/tse_ref_design_top.pof	CFI_512MB	CCA78073		<input checked="" type="checkbox"/>	<input type="checkbox"/>
	-Page_0				<input checked="" type="checkbox"/>	<input type="checkbox"/>
	-OPTION_BITS				<input checked="" type="checkbox"/>	<input type="checkbox"/>
	<none>	EP2SGX90	00000000	<none>	<input type="checkbox"/>	<input type="checkbox"/>

6. Click **Start** and wait for the flash to be programmed.



If the “Device 1 silicon ID is not ready” messages appear after “Performing verification of type Nominal on device(s)” (as shown in [Figure 19](#)), press the nCONFIG button (S1) on the development board to start the programming. This message appears with boards that have already had their flash programmed.

Figure 19. Device Not Ready Messages

Type	Message
i	Info: Started Programmer operation at Wed Nov 14 11:35:12 2007
i	Info: Device 1 contains JTAG ID code 0x020A20DD
i	Info: Device 1 silicon ID is ALTERA04-0
i	Info: Erasing MAXII configuration device(s)
i	Info: Programming device(s)
i	Info: Performing verification of type Nominal on device(s)
i	Info: Device 1 silicon ID is not ready - waiting for pfl_flash_access_granted to be asserted
i	Info: Device 1 silicon ID is not ready - waiting for pfl_flash_access_granted to be asserted
i	Info: Device 1 silicon ID is not ready - waiting for pfl_flash_access_granted to be asserted
i	Info: Device 1 silicon ID is not ready - waiting for pfl_flash_access_granted to be asserted
i	Info: Device 1 CFI Flash is Spansion S29GL512N or S29GL512P (16 bits data bus)
i	Info: Erasing CFI Flash configuration device(s)
i	Info: Programming status: erasing flash memory at byte address 0x00000000
i	Info: Programming status: erasing flash memory at byte address 0x00020000

Documents Referenced

This application note references the following documents:

- *CRC Compiler User Guide*
- Finisar FCMJ-8520/8521-3 1000BASE-T Copper SFP Transceiver Product Specification
- Marvell 88E1111 Datasheet – Integrated 10/100/1000 Ultra Gigabit Ethernet Transceiver
- *Shift Register (RAM-Based) User Guide*
- *Stratix II GX PCI Express Development Board Reference Manual*
- *Triple Speed Ethernet MegaCore Function User Guide*

Document Revision History

Table 13 shows the revision history for this application note.

Date and Document Version	Changes Made	Summary of Changes
June 2009 v1.1	Changed references from “version 7.2 or later” to “version 7.2”	—
January 2008 v1.0	Initial release.	—



101 Innovation Drive
San Jose, CA 95134
www.altera.com
Technical Support:
www.altera.com/support/

Copyright © 2009 Altera Corporation. All rights reserved. Altera, The Programmable Solutions Company, the stylized Altera logo, specific device designations, and all other words and logos that are identified as trademarks and/or service marks are, unless noted otherwise, the trademarks and service marks of Altera Corporation in the U.S. and other countries. All other product or service names are the property of their respective holders. Altera products are protected under numerous U.S. and foreign patents and pending applications, maskwork rights, and copyrights. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera Corporation. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

