



この資料は英語版を翻訳したもので、内容に相違が生じる場合には原文を優先します。こちらの日本語版は参考用としてご利用ください。設計の際には、最新の英語版で内容をご確認ください。

はじめに

アルテラのトリプル・スピード・イーサネット (TSE) データ・パスのリファレンス・デザインは、アルテラの TSE MegaCore® ファンクションと 2 つのシリアル・トランシーバを使用するサンプル SOPC Builder システムを提供します。このリファレンス・デザインは、アルテラ TSE MegaCore ファンクションの動作を、ハードウェアでの最大ワイヤ・スピード性能まで実証します。このデザインにより、アルテラ FPGA デザインへの TSE MegaCore ファンクションの統合を評価できます。

このリファレンス・デザインは、以下の特長を備えています。

- Stratix® II GX PCI Express 開発キット、2 個の Small Form-factor Pluggable (SFP) モジュール、イーサネット・ケーブル・アセンブリ、PC、および USB-Blaster™ または ByteBlaster™ ケーブル・アセンブリのわずかなハードウェア機器を使用するだけで完全なテストを実施できる。
- アルテラ TSE MegaCore ファンクションの 2 つのインスタンスを実装し、SGMII モードでの 10/100/1000 Mbps イーサネット動作 (オート・ネゴシエーション付き) をサポートする。
- パケット数、パケット長、ペイロードのデータ型、送信元および送信先 MAC アドレスなど、プログラム可能な設定をサポートする。
- 最大理論データ・レートまでエラーなしで、イーサネット・パケットの送信および受信を実証する。
- CAT5 イーサネット・ケーブル・アセンブリを搭載した SFP モジュールとイーサネット・スイッチ (オプション) により外部ループバックをサポートする。
- デザインを制御し、TX および RX パケットの統計情報結果をモニタするコマンドライン・インタフェース (CLI) を提供する。
- 銅線用の SFP モジュール・イーサネット PHY をコンフィギュレーションするシリアル・インタフェースを提供する。

概要

このリファレンス・デザインは、イーサネット・アプリケーション向けの2つのアルテラ TSE MegaCore ファンクション (MAC + PCS + PMA) を統合する完全な実用サブシステムを実証します。このデザインでは、ハードウェア・プラットフォームとして、2つの SFP ケージを含む Stratix II GX PCI Express 開発キットを使用します。また、このリファレンス・デザインには、ドライバ、プログラミング情報、および CLI を含むソフトウェアも付属しており、デモやテストを実施できます。

このリファレンス・デザインは、SGMII モードを使用して 10、100、および 1000 Mbps イーサネット動作をすべて可能にする 1.25 Gbps シリアル・トランシーバを介して、TSE MegaCore ファンクションを銅線用の SFP モジュールにインタフェースします。このリファレンス・デザインは、イーサネット・パケットのストリームを TSE MegaCore ファンクションに送信します。TSE MegaCore ファンクションは次に、これらのパケットを SFP モジュールとケーブルに送ります。このケーブルでは、イーサネット・ケーブル・アセンブリを搭載した SFP モジュール、またはイーサネット・スイッチを通じてイーサネット・パケットが外部でループバックされます。このリファレンス・デザインは、さまざまなモードでの TSE MegaCore ファンクションの動作を、最大スループット・レートまでのライブ・トラフィックで実証し、レシーバのエラー率 (エラーがある場合) を示すことができます。

リファレンス・ デザインの 概要

このリファレンス・デザインは、さまざまな速度のイーサネット動作を制御、テスト、および監視できる汎用プラットフォームを提供します。このリファレンス・デザインは SOPC Builder を使用して開発されています。図 1 に、このリファレンス・デザインの上位レベルのブロック図を示します。

図 1. リファレンス・デザインのシステム・ブロック図

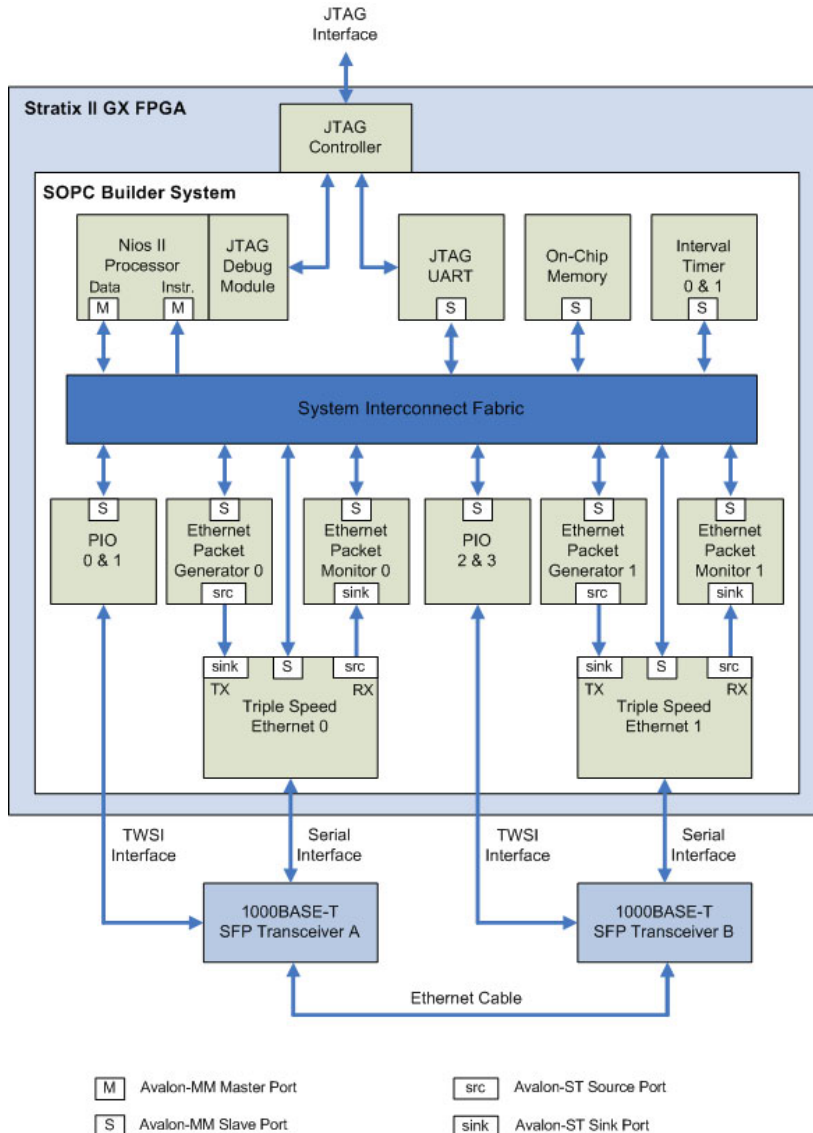


図 2 に、このシステムを生成するために使用する SOPC Builder のセットアップを示します。

図 2. リファレンス・デザインの SOPC Builder セットアップ

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

Use	Connections	Module Name	Description	Clock	Base	End	...
<input checked="" type="checkbox"/>	[Diagram showing connections to sys_clk]	cpu	Nios II Processor	sys_clk			
		instruction_master	Avalon Master				
		data_master	Avalon Master				
		jtag_debug_module	Avalon Slave				
<input checked="" type="checkbox"/>	[Diagram showing connections to sys_clk]	onchip_mem	On-Chip Memory (RAM or ROM)	sys_clk	0x00080800	0x00080fff	
		s1	Avalon Slave				
<input checked="" type="checkbox"/>	[Diagram showing connections to sys_clk]	jtag_uart	JTAG UART	sys_clk	0x00040000	0x0007ffff	
		avalon_jtag_slave	Avalon Slave				
<input checked="" type="checkbox"/>	[Diagram showing connections to sys_clk]	pll	PLL	clk	0x00081880	0x0008189f	
		s1	Avalon Slave				
<input checked="" type="checkbox"/>	[Diagram showing connections to sys_clk]	pio	PIO (Parallel IO)	sys_clk	0x00081920	0x0008192f	
		s1	Avalon Slave				
<input checked="" type="checkbox"/>	[Diagram showing connections to sys_clk]	pio_1	PIO (Parallel IO)	sys_clk	0x00081930	0x0008193f	
		s1	Avalon Slave				
<input checked="" type="checkbox"/>	[Diagram showing connections to sys_clk]	pio_2	PIO (Parallel IO)	sys_clk	0x00081940	0x0008194f	
		s1	Avalon Slave				
<input checked="" type="checkbox"/>	[Diagram showing connections to sys_clk]	pio_3	PIO (Parallel IO)	sys_clk	0x00081950	0x0008195f	
		s1	Avalon Slave				
<input checked="" type="checkbox"/>	[Diagram showing connections to sys_clk]	eth_gen_inst	Ethernet Generator	sys_clk	0x00081800	0x0008183f	
		avalon_slave_0	Avalon Slave				
		avalon_streaming_sou...	Avalon Streaming Source				
<input checked="" type="checkbox"/>	[Diagram showing connections to sys_clk]	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"/>	[Diagram showing connections to sys_clk]	eth_mon_inst	Ethernet Monitor	sys_clk			
		avalon_slave_0	Avalon Slave	sys_clk	0x000818a0	0x000818bf	
		avalon_streaming_sink	Avalon Streaming Sink				
<input checked="" type="checkbox"/>	[Diagram showing connections to sys_clk]	eth_gen_inst_1	Ethernet Generator	sys_clk			
		avalon_slave_0	Avalon Slave	sys_clk	0x00081840	0x0008187f	
		avalon_streaming_sou...	Avalon Streaming Source				
		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"/>	[Diagram showing connections to sys_clk]	eth_mon_inst_1	Ethernet Monitor	sys_clk			
		avalon_slave_0	Avalon Slave	sys_clk	0x000818c0	0x000818df	
		avalon_streaming_sink	Avalon Streaming Sink				
<input checked="" type="checkbox"/>	[Diagram showing connections to sys_clk]	timer	Interval Timer	sys_clk			
		s1	Avalon Slave	sys_clk	0x000818e0	0x000818ff	
<input checked="" type="checkbox"/>	[Diagram showing connections to sys_clk]	timer_1	Interval Timer	sys_clk			
		s1	Avalon Slave	sys_clk	0x00081900	0x0008191f	

以下の項では、リファレンス・デザインで使用する各コンポーネントについて簡単に説明します。

Nios II プロセッサ

Nios II プロセッサは、リファレンス・デザインのシステム・コンポーネントを設定およびコンフィギュレーションするコントロール・プレーンとして使用します。また、イーサネット・パケットの生成を開始し、パケット受信のステータスを監視します。

オンチップ・メモリ

このリファレンス・デザインでは、256 K バイトのオンチップ・メモリ・ブロック・サイズを使用します。このメモリは、ソフトウェア・コードを保存するための SOPC システム RAM として使用されます。

JTAG UART

JTAG UART コアは、Nios II プロセッサと SOPC Builder システムの間で、シリアル・キャラクタ・ストリームを転送します。このコアは、JTAG インタフェースの複雑さを隠蔽する、シンプルなレジスタ・マップト Avalon® インタフェースを提供します。Nios II プロセッサは、コントロールおよびデータ・レジスタを読み書きしてコアと通信します。

PLL

PLL (Phase-Locked Loop) コアは、開発キットの 100 MHz クリスタルから入力クロックを取り込んで、SOPC Builder システム用のシステム全体のクロック・ソースとして 83.33 MHz PLL 出力クロックを生成します。

パラレル入力 / 出力

パラレル入力 / 出力 (PIO) コアは、1000BASE-T 銅線用の SFP モジュールの 2 線式シリアル・インタフェース (TWSI) に接続される、汎用 I/O ポートを提供します。PIO コアは、TWSI プロトコルに準拠した「ビット・バンギング」アプローチにより、1000BASE-T 銅線用の SFP モジュールの PHY レジスタへの簡単なアクセスを提供します。

トリプル・スピード・イーサネット MegaCore ファンクション

TSE MegaCore ファンクションは、イーサネット・アプリケーション向けの統合イーサネット MAC、PCS、および PMA ソリューションを提供します。このメガファンクションは、Avalon Streaming (Avalon-ST) インタフェースから Stratix II GX デバイス搭載の 1.25 Gbps シリアル・トランシーバ・インタフェースにイーサネット・パケットを送信し、逆方向からパケットを受信します。

イーサネット・パケット・ジェネレータ

イーサネット・パケット・ジェネレータ・ブロックは、Component Editor を使用して作成される SOPC カスタム・コンポーネントです。一方のサイドに制御用として Avalon Memory-Mapped (Avalon-MM) スレーブ・インタフェース、反対サイドにイーサネット・パケットを TSE MegaCore ファンクションに送信するための Avalon-ST ソース・インタフェースがあります。このブロックは、TSE MegaCore ファンクションの送信 FIFO インタフェースに送るイーサネット・パケットのストリームを生成して、このインタフェースをドライブします。

イーサネット・パケット・モニタ

イーサネット・パケット・モニタ・ブロックは、Component Editor を使用して作成される SOPC カスタム・コンポーネントです。一方のサイドには、制御用の Avalon-MM スレーブ・インタフェース、反対サイドにはデータ・パス用の Avalon-ST シンク・インタフェースがあります。このブロックには、TSE MegaCore ファンクションの受信 FIFO インタフェースによってイーサネット・パケットのストリームが供給されます。また、イーサネット・パケット・モニタは受信したペイロードの精度の検証も行います。

インターバル・タイマ

インターバル・タイマ・コアは、Nios II プロセッサ・システムがさまざまなイーサネット動作の性能とスループット・レートを計算するのに使用する 32 ビット・タイマです。

機能の説明

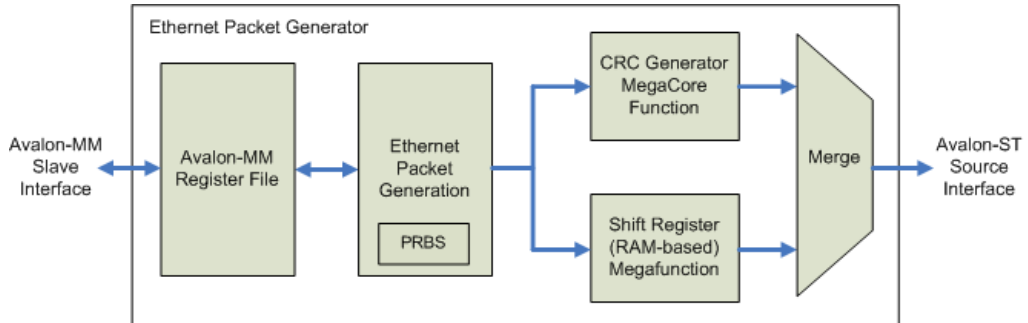
この項では、TSE リファレンス・デザインの以下のモジュールの動作について説明します。また、メモリ・マップ、レジスタ・マップおよび信号についても説明します。

- 「イーサネット・パケットの生成」
- 10 ページの「イーサネット・パケットの受信と検証」
- 11 ページの「フィジカル・メディアとのインタフェース」
- 11 ページの「イーサネット・パケットのループバック動作」
- 13 ページの「メモリ・マップ」
- 14 ページの「レジスタ・マップ」
- 19 ページの「インタフェース信号の説明」

イーサネット・パケットの生成

イーサネット・パケットは、リファレンス・デザインのイーサネット・パケット・ジェネレータ・ブロックによって生成されます。図 3 に、イーサネット・パケット・ジェネレータ・ブロックの上位レベルのブロック図を示します。

図 3. イーサネット・パケット・ジェネレータのブロック図



このモジュールは、以下のコンポーネントで構成されます。

- Avalon-MM レジスタ・ファイル
- イーサネット・パケット生成ブロック
- アルテラ CRC コンパイラ・ジェネレータ MegaCore ファンクション
- シフト・レジスタ (RAM ベース) メガファンクション

Avalon-MM レジスタ・ファイル・コンポーネントは、システムがイーサネット・パケットの生成を開始するために必要なすべてのパラメータおよび設定をコンフィギュレーションするためのレジスタ・インタフェースを提供します。システムはこのレジスタ・インタフェースを通じて、以下のようなプログラマブルな設定をコンフィギュレーションできます。

- 送信するパケットの総数
- インクリメンタルまたはランダム・データ型
- 固定またはランダム・パケット長
- 送信元および送信先 MAC アドレス
- PRBS ブロック用のランダム・シード

また、Avalon-MM レジスタ・ファイルは、送信動作のステータスを提供し、正常に送信されたパケット数をレポートします。イーサネット・パケット・ジェネレータのレジスタについては、[14 ページの表 2](#)を参照してください。

すべてのレジスタの設定のコンフィギュレーションが終了し、operation レジスタのスタート・ビットが1に設定されると、イーサネット・パケット・ジェネレータは、TSE MegaCore ファンクションへのイーサネット・パケットのストリームの生成を開始します。このスタート・ビットが設定されると、イーサネット・パケット・ヘッダとプログラムされているパケット数に対応するデータ・ペイロードを生成することにより、イーサネット・パケット生成ブロックのステート・マシンを起動します。このスタート・ビットは、イーサネット・パケット・ジェネレータ・ブロックのソフト・リセットとしても機能し、生成前にステート・マシンをすべてのジェネレータのステータス・レジスタと併せてリセットします。

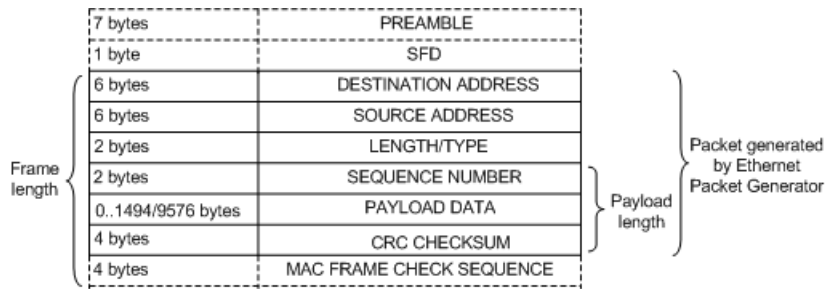
終了前に動作を停止する場合は、operation レジスタのストップ・ビットを設定する書き込みを発行して、現在実行中の動作を終了させることができます。ストップ・ビットがアサートされると、ステート・マシンは現在のパケットの生成を終了して動作を停止します。

次に、イーサネット・パケット生成ブロックで生成されるパケットはすべて、チェックサム計算のためにCRC ジェネレータ MegaCore ファンクションと、RAM ベースのシフト・レジスタ・メガファンクションに送信されます。CRC ジェネレータ MegaCore ファンクションによって計算済みチェックサム出力とパケットがマージされるのを待つ間、そのパケットを一時的に格納しておくために、RAM ベースのシフト・レジスタが使用されます。有効な CRC チェックサムとパケット・ストリームのマージが完了すると、完全なフレームが Avalon-ST インタフェースに送信されます。

イーサネット・パケットは欠落する可能性があるため、生成されたパケットにCRC チェックサムを埋め込んでおく受信インタフェースでのパケットの検証が容易です。パケットが欠落しても、システムは埋め込まれた CRC チェックサムに基づいて、中断することなく受信パケットの検証を継続できます。また、受信パケットの総数を提供することもできます。

イーサネット・パケット・ジェネレータで生成されるパケットは、7 バイトのプリアンブル、1 バイトのフレーム開始部 (SFD)、および4 バイトの MAC で計算されるフレーム・チェック・シーケンス (FCS) を除く標準イーサネット・パケットです。図 4 に、イーサネット・パケット・ジェネレータの生成フレーム・フォーマットを示します。

図 4. イーサネット・パケットの生成フレーム・フォーマット



送信先および送信元 MAC アドレスは、レジスタ・インタフェースを通じてプログラムできます。送信先 MAC アドレスは、ユニキャスト・アドレスまたはブロードキャスト・アドレスに設定できます。ユニキャスト・パケットが送信される場合、送信先 MAC アドレスはリファレンス・デザインを受信 TSE MAC アドレスと同じ値でなければなりません。送信元 MAC アドレスは、送信 TSE MAC アドレスと同じ値でプログラムする必要があります。これにより、リファレンス・デザインの TSE MegaCore ファンクションの受信エンドは、常に正しいパケットを受信して内容を検証できます。

生成される各パケットのパケット長は、プログラマブルな値に固定するか、またはランダムなパケット・サイズにすることができます。プログラマブルな固定パケット長の範囲は 24～9600 バイト、ランダムなパケット長の範囲は 24～1518 バイトです。フレーム長が 64 バイト未満の場合、TSE MegaCore ファンクションによってパディング・バイト (0x00) が自動的に付加されて 64 バイト以上になります。

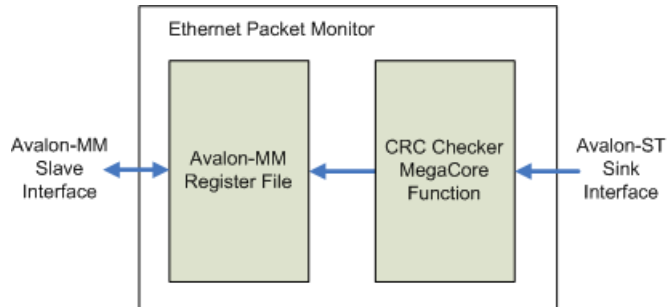
データ・ペイロードはインクリメンタルまたは擬似ランダム of のいずれかです。インクリメンタル・データ型が選択された場合、ペイロードのデータ値は 0x00 から始まり、0x01、0x02、というように続きます。ただし、擬似ランダム・データ型が選択された場合は、PRBS ブロックから生成されるランダム値がペイロードの内容として使用されます。PRBS ブロックで使用されるランダム・シードは、レジスタ・ファイルの rand_seed0 および rand_seed1 レジスタを介してプログラム可能です。使用される PRBS 多項式は PRBS23 ($2^{23}-1$) です。

2 バイトのシーケンス番号と 4 バイトの CRC チェックサムは、このリファレンス・デザインのパケット・ペイロードの一部であることに注意してください。シーケンス番号は、すべてのパケット・ペイロードの先頭の 2 バイトに格納され、デバッグ目的でパケットの受信順序を追跡するのに使用されます。CRC ジェネレータ MegaCore ファンクションによって計算される CRC チェックサムは、どのパケット・ペイロードでも最後の 4 バイトに格納されます。このチェックサムにより、イーサネット・パケット・ジェネレータから開始してイーサネット・パケット・モニタで終了するパケットのデータ・インテグリティが保証されます。したがって、このリファレンス・デザインの最小ペイロード長は 6 バイトです。

イーサネット・パケットの受信と検証

TSE MegaCore ファンクションの RX FIFO インタフェースから送信されるイーサネット・パケットは、リファレンス・デザイン内のイーサネット・パケット・モニタ・ブロックによって受信されます。次にパケットのペイロードが検証されます。図 5 に、イーサネット・パケット・モニタのブロック図を示します。

図 5. イーサネット・パケット・モニタのブロック図



このモジュールは、以下のコンポーネントで構成されます。

- Avalon-MM レジスタ・ファイル
- アルテラ CRC コンパイラ・チェッカ MegaCore ファンクション

Avalon-MM レジスタ・ファイル・コンポーネントは、システムがイーサネット・パケットの受信を開始して監視するために必要な設定をコンフィギュレーションするためのレジスタ・インタフェースを提供します。このレジスタ・インタフェースを介して、システムは受信するパケットの総数をコンフィギュレーションできます。また、受信動作のステータス、および受信した正常なパケット数、受信した不良パケット数、受信した性能に関するバイト数とサイクル数、およびスループット・レートに関する統計情報カウンタ・セットも提供されます。イーサネット・パケット・モニタのレジスタの各フィールドについては、17 ページの表 5 を参照してください。

このレジスタ・インタフェースがコンフィギュレーションされた後、イーサネット・パケット・モニタを有効にしてパケットを受信できるようにする必要があります。receive_ctrl_status レジスタのスタート・ビットを設定すると、パケット受信統計情報カウンタがリセットされ、モニタが受信パケットを受信できるようになります。

receive_ctrl_status レジスタのストップ・ビットがアサートされると、モニタはパケットの受信を停止し、統計情報カウンタの更新を停止します。

リファレンス・デザインで TSE MegaCore ファンクションの別のインスタンスにループバックされたイーサネット・パケットは、イーサネット・パケット・モニタ・モジュール内の CRC チェッカ MegaCore ファンクションに直接ストリームされます。このモジュールは、受信したパケットに基づいて CRC チェックサムを計算し、その値をパケット・ペイロードの最後の 4 バイトに埋め込まれたチェックサム値と照合して検証します。その後、受信したパケットが正常または不良パケットであることを示すステータスを出力し、それに応じて統計情報レジスタを更新します。

フィジカル・メディアとのインタフェース

このリファレンス・デザインでは、Finisar の 1000BASE-T 銅線用の SFP モジュールの 2 線式シリアル・インタフェース (TWSI) にインタフェースする 2 本の PIO ピンを使用して、内部 PHY デバイスのレジスタをコンフィギュレーションします。TWSI は、シリアル・データ・ライン (SDA) では双方向ラインとして、シリアル・クロック・ライン (SCL) では出力ラインとして動作します。このソフトウェアは、TWSI プロトコルに準拠するこれらの 2 本の PIO ピンをトグルして、SFP モジュールの PHY レジスタと通信します。Finisar の 1000BASE-T 銅線用の SFP モジュールの PHY チップには、スレーブ・アドレス 0xAC で TWSI を通してアクセスできます。



詳しくは、「Marvell PHY 88E1111 Datasheet」を参照してください。

SFP モジュールの PHY レジスタをコンフィギュレーションすると、インタフェース同士でオート・ネゴシエーションを実行して、イーサネット動作の共通モードを解決できます。CLI を通して、次のイーサネット動作モードがサポートされます。

- 10BASE-T 全二重
- 10BASE-T 半二重
- 100BASE-T 全二重
- 100BASE-T 半二重
- 1000BASE-T 全二重

イーサネット・パケットのループバック動作

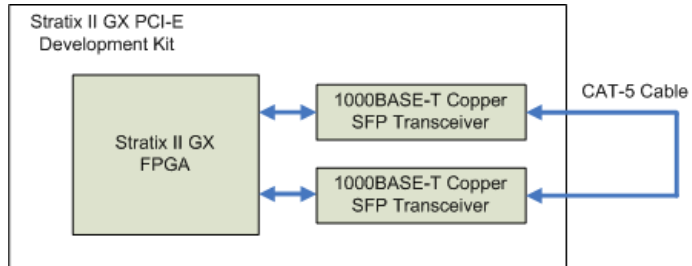
このリファレンス・デザインは、次のいずれかのオプションで動作できます。

- イーサネット・ケーブル・アセンブリによる 1000BASE-T 銅線用の SFP モジュールを介した外部ループバック
- ネットワーク・スイッチによる 1000BASE-T 銅線用の SFP モジュールを介した外部ループバック

イーサネット・ケーブル・アセンブリによる 1000BASE-T 銅線用の SFP モジュールを介した外部ループバック

図 6 に、イーサネット・ケーブル・アセンブリを使用した外部ループバック動作の上位レベルの図を示します。

図 6. イーサネット・ケーブル・アセンブリを介した外部ループバック動作



リファレンス・デザインが、イーサネット・ケーブル・アセンブリを使用した外部ループバック・モードで動作する場合は、1 本の CAT-5 ケーブルを両方の 1000BASE-T 銅線用の SFP トランシーバに接続する必要があります。このケーブルにより、SGMII モードの 2 つの TSE MegaCore ファンクション間にリンクが形成されます。

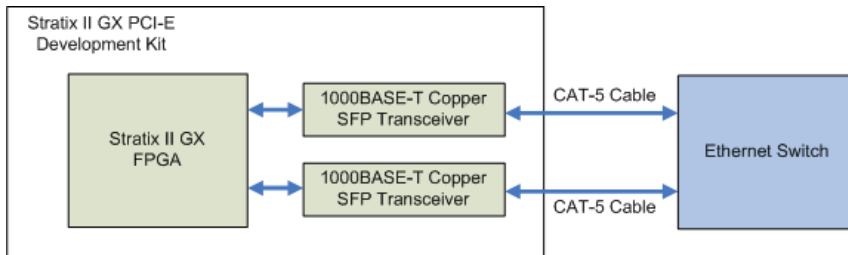
システムがコンフィギュレーションされると、イーサネット・パケット・ジェネレータが TX FIFO インタフェースを介して送信 TSE MegaCore ファンクションに送るパケットの生成を開始します。各パケットには、TSE MAC によって付加され、1000BASE-T 銅線用の SFP トランシーバに送出される 4 バイトのフレーム・チェック・シーケンスがあります。これらのパケットは、イーサネット・ケーブル・アセンブリを通じて別の 1000BASE-T 銅線用の SFP トランシーバに送信され、受信 TSE MegaCore ファンクションに到達します。パケットは次に FCS フィールドを使用して検証されます。その後、チェックサムが廃棄されます。残りのパケットは、RX FIFO インタフェースを介してイーサネット・パケット・モニタにストリーム出力されて、さらにパケット検証が行われます。

すべてのパケットの受信が終了すると、統計情報レジスタに基づいて、パケットの統計情報とスループット・レートがレポートされます。これは TSE MegaCore ファンクションによって達成できる実際のスループット・レートを示すものです。

ネットワーク・スイッチによる 1000BASE-T 銅線用の SFP モジュールを介した外部ループバック

図 7 に、イーサネット・スイッチとイーサネット・ケーブル・アセンブリを使用した外部ループバック動作の上位レベルの図を示します。

図 7. ネットワーク・スイッチによる外部ループバック動作



リファレンス・デザインが外部ループバック・モードで動作する場合は、2本のCAT-5ケーブルで1000BASE-T銅線用のSFPトランシーバをイーサネット・スイッチの2つのポートに接続する必要があります。このデザインでは、外部イーサネット・スイッチにより2つのTSE MegaCoreファンクションの間にリンクが形成されます。

この動作は、イーサネット・ケーブル・アセンブリのみを利用した外部ループバック・モードとまったく同じです。これらのパケットはイーサネット・スイッチに送信され、そのスイッチからスイッチに接続されている受信TSE MegaCoreファンクションの正しいポートに転送されます。このスイッチは、接続されている各ポートの送信元MACアドレスに基づいて転送先を自己学習し、パケットはその送信先MACアドレスに基づいて正しいポートに転送されます。また、イーサネット・スイッチは追加の管理用パケットも送信します。これらのパケットはTSEリファレンス・デザインで受信しますが、余分なパケットとみなされます。

メモリ・マップ

表1に、このリファレンス・デザインのSOPCシステムの完全なメモリ・マッピングを定義します。

表 1. TSE リファレンス・デザイン・メモリ・マップ (1 / 2)	
アドレス	説明
0x00040000 – 0x0007FFFF	オンチップ・メモリ - RAM (256 KB)
0x00080800 – 0x00080FFF	CPU JTAG デバッグ・モジュール
0x00081000 – 0x000813FF	トリプルスピード・イーサネット 0
0x00081400 – 0x000817FF	トリプルスピード・イーサネット 1
0x00081800 – 0x0008183F	イーサネット・パケット・ジェネレータ 0
0x00081840 – 0x0008187F	イーサネット・パケット・ジェネレータ 1

表 1. TSE リファレンス・デザイン・メモリ・マップ (2 / 2)

アドレス	説明
0x00081880 – 0x0008189F	PLL
0x000818A0 – 0x000818BF	イーサネット・パケット・モニタ 0
0x000818C0 – 0x000818DF	イーサネット・パケット・モニタ 1
0x000818E0 – 0x000818FF	インターバル・タイマ 0
0x00081900 – 0x0008191F	インターバル・タイマ 1
0x00081920 – 0x0008192F	PIO 0
0x00081930 – 0x0008193F	PIO 1
0x00081940 – 0x0008194F	PIO 2
0x00081950 – 0x0008195F	PIO 3
0x00081960 – 0x00081967	JTAG UART

レジスタ・マップ

表 2 に、イーサネット・パケット・ジェネレータのレジスタの各フィールドの機能を示します。

表 2. イーサネット・パケット・ジェネレータのレジスタ・マップ (1 / 2)

アドレス オフセット	名称	説明	アクセス	HW リセット
0x00	number_packet	32 ビット・パケット数レジスタ。 イーサネット・パケット・ジェネレータが TSE MegaCoreファンクションのために生成するパケットの総数。	RW	0x0
0x04	config_setting	16ビット・コンフィギュレーション設定レジスタ。 ビットの説明は、表 3 を参照してください。	RW	0x0
0x08	rand_seed0	32 ビット下位ランダム・シード・レジスタ。 config_setting[15]が1に設定されているときに、PRBS ジェネレータのビット 0 ~ 31 をプリロードするのに使用されます。	RW	0x0

表 2. イーサネット・パケット・ジェネレータのレジスタ・マップ (2 / 2)

アドレス オフセット	名称	説明	アクセス	HW リセット
0x0C	rand_seed1	16 ビット上位ランダム・シード・レジスタ。 config_setting[15] が 1 に設定されているときに、PRBS ジェネレータのビット 32 ~ 47 をプリロードするのに使用されます。ビット 16 ~ 31 は予約されています。	RW	0x0
0x10	source_addr0	32 ビット下位送信元 MAC アドレス・レジスタ。 イーサネット・パケットの送信元 MAC アドレス・フィールドのビット 0 ~ 31 を定義するのに使用されます。この送信元 MAC アドレスは、送信 TSE MegaCore ファンクションと同じ MAC アドレスを使用してプログラムする必要があります。	RW	0x0
0x14	source_addr1	16 ビット上位送信元 MAC アドレス・レジスタ。 イーサネット・パケットの送信元 MAC アドレス・フィールドのビット 32 ~ 47 を定義するのに使用されます。この送信元 MAC アドレスは、送信 TSE MegaCore ファンクションと同じ MAC アドレスを使用してプログラムする必要があります。ビット 16 ~ 31 は予約されています。	RW	0x0
0x18	destination_addr0	32 ビット下位送信先 MAC アドレス・レジスタ。 イーサネット・パケットの送信先 MAC アドレス・フィールドのビット 0 ~ 31 を定義するのに使用されます。この送信先 MAC アドレスは、ユニキャスト・パケット用の受信 TSE MegaCore ファンクションと同じ MAC アドレスでプログラムする必要があります。	RW	0x0
0x1C	destination_addr1	16 ビット上位送信先 MAC アドレス・レジスタ。 イーサネット・パケットの送信先 MAC アドレス・フィールドのビット 32 ~ 47 を定義するのに使用されます。この送信先 MAC アドレスは、ユニキャスト・パケット用の受信 TSE MegaCore ファンクションと同じ MAC アドレスでプログラムする必要があります。ビット 16 ~ 31 は予約されています。	RW	0x0
0x20	operation	3 ビット・オペレーション・レジスタ。 ビットの説明は、表 4 を参照してください。	RW/RO	0x0
0x24	packet_tx_count	32 ビット・パケット送信カウント・レジスタ。 イーサネット・パケット・ジェネレータによって正常に送信されたデータ・パケット数をカウントするのに使用されます。このレジスタは、operation[0] がアサートされると 0 にクリアされます。	RO	0x0

表 3 に、config_setting レジスタのビットの説明を示します。

ビット	名称	アクセス	説明
0	length_sel	RW	パケット長タイプ・セレクト 0: 固定パケット長 1: ランダム・パケット長
14:1	pkt_length	RW	固定パケット長。 パケット長を 24 ~ 9600 の範囲でプログラムできます。このフィールドは、パケット長タイプ・セレクト・ビットが 0 に設定されているときにのみ有効です。
15	pattern_sel	RW	データ・タイプ・セレクト 0: インクリメンタル・データ・パターン 1: ランダム・データ・パターン
31:16	reserved	—	予約ビット。読み出すと 0 を返します。

表 4 に、operation レジスタのビットの説明を示します。.

ビット	名称	アクセス	説明
0	start	RW	オペレーション開始。 1 を書き込むと、イーサネット・パケット・ジェネレータから TSE MegaCore ファンクションへのパケット生成がスタートします。このビットはパケット生成が始まると自動的にクリアされます。
1	stop	RW	オペレーション停止。 1 を書き込むと、イーサネット・パケット・ジェネレータは TSE MegaCore ファンクションへのパケット生成とパケット送信を停止します。このビットがアサートされると、現在のパケットの送信を終了して停止します。このビットは、オペレーション開始ビットが 1 に設定されるとクリアされます。
2	tx_done	RO	送信完了ステータス。 このビットは、イーサネット・パケット・ジェネレータが number_packet レジスタにプログラムされたすべてのパケットの送信を終了すると 1 に設定されます。このビットは、オペレーション開始ビットが 1 に設定されるとクリアされます。
31:3	reserved	—	予約ビット。読み出すと 0 を返します。

表 5 に、イーサネット・パケット・モニタのレジスタの各フィールドの機能を示します。

表 5. イーサネット・パケット・モニタのレジスタ・マップ (1/2)				
アドレス オフセット	名称	説明	アクセス	HW リセット
0x00	number_packet	32 ビット・パケット数レジスタ。 イーサネット・パケット・モニタが TSE MegaCore ファンクションから受信したパケッ トの総数。	RW	0x0
0x04	packet_rx_ok	32 ビット・パケット受信カウント・レジスタ。 エラーなしで受信したパケット数をカウントす るのに使用されます。	RO	0x0
0x08	packet_rx_error	32 ビット・パケット受信(エラーあり)レジスタ。 CRC エラーを伴って受信したパケット数をカ ウントするのに使用されます。	RO	0x0
0x0C	byte_rx_count0	32 ビット下位バイト受信カウント・レジスタ。 受信したデータ・バイト数をカウントするのに 使用されます。このレジスタは、64 ビット・バ イト・カウンタのビット 0 ~ 31 にマップしま す。最初に byte_rx_count0 レジスタを読 み出し、次のクロック・サイクルで、カウン タがインクリメントしている間に、byte_rx_ count1 レジスタを読み出します。この手順に より、パケット受信時に不正なカウントを読み 取らないようにします。	RO	0x0
0x10	byte_rx_count1	32 ビット上位バイト受信カウント・レジスタ。 受信したデータ・バイト数をカウントするのに 使用されます。このレジスタは 64 ビット・バ イト・カウンタのビット 32 ~ 63 にマップしま す。最初に byte_rx_count0 レジスタを読 み出し、次のクロック・サイクルで、カウン タがインクリメントしている間に、byte_rx_ count1 レジスタを読み出します。この手順に より、パケット受信で誤ったカウントを読み取 ることを防止します。	RO	0x0
0x14	cycle_rx_count0	32 ビット下位サイクル受信カウント・レジスタ。 最初に受信したパケット・バイトの開始から最 後に受信したパケット・バイトの終わりまでの サイクル数をカウントするのに使用されます。 このレジスタは 64 ビット・サイクル・カウ ンタのビット 0 ~ 31 にマップします。	RO	0x0

表 5. イーサネット・パケット・モニタのレジスタ・マップ (2 / 2)

アドレス オフセット	名称	説明	アクセス	HW リセット
0x18	cycle_rx_count1	32ビット上位サイクル受信カウント・レジスタ。 最初に受信したパケット・バイトの開始から最後に受信したパケット・バイトの終わりまでのサイクル数をカウントするのに使用されます。このレジスタは、64 ビット・サイクル・カウンタのビット 32 ~ 63 にマップします。	RO	0x0
0x1C	receive_ctrl_status	10 ビット受信コントロールおよびステータス・レジスタ。 ビットの説明は、表 6 を参照してください。	RW/RO	0x0

表 6 に、receive_ctrl_status レジスタのビットの説明を示します。

表 6. 受信コントロールおよびステータス・レジスタのビットの説明 (1 / 2)

ビット	名称	アクセス	説明
0	start	RW	オペレーション開始。 1 を書き込むと、TSE MegaCore ファンクションからイーサネット・パケット・モニタへのパケット受信を開始します。このビットはパケット受信が始まると自動的にクリアされます。
1	stop	RW	オペレーション停止。 1 を書き込むとイーサネット・パケット・モニタはパケット受信を停止します。このビットがアサートされると、イーサネット・パケット・モニタは残りのステータス・レジスタの更新をすぐに停止します。このビットは、オペレーション開始ビットが 1 に設定されるとクリアされます。
2	rx_done	RO	受信完了ステータス。 このビットは、イーサネット・パケット・モニタが number_packet レジスタにプログラムされたすべてのパケットの受信を終了すると 1 に設定されます。このビットは、オペレーションの開始ビットが 1 に設定されるとクリアされます。
3	crcbad	RO	CRC エラー・パケット。 このビットは、受信中の現在のパケットが CRC エラー・パケットとして検出されると 1 に設定されます。このステータス・ビットは、受信するすべてのパケットに対して継続的に更新されます。

表 6. 受信コントロールおよびステータス・レジスタのビットの説明 (2 / 2)

ビット	名称	アクセス	説明
9:4	rx_err	RO	受信エラー・ステータス。 TSE MegaCore ファンクションの受信 FIFO インタフェースからの rx_err[5:0] 信号は、最終フレーム・オクテットごとにフィールドにマップされます。すべての信号の定義については、 「Triple Speed Ethernet MegaCore Function User Guide」 を参照してください。
31:10	reserved	—	予約ビット。読み出すと 0 を返します。



TSE MegaCore と Marvell PHY 88E1111 のレジスタ・マップについて詳しくは、[「Triple Speed Ethernet MegaCore Function User Guide」](#) および [「Marvell PHY 88E1111 Datasheet」](#) を参照してください。

インタフェース信号の説明

この項では、リファレンス・デザインのアプリケーション・ロジックのインタフェース信号について説明します。

表 7 に、クロックおよびリセット信号を示します。

表 7. クロックおよびリセット信号		
信号名	入力/出力	説明
clk	入力	リファレンス・デザイン・クロック。クロックのソースは 100 MHz オシレータ (X1) です。
ref_clk_to_the_altera_ethernet	入力	TSE トランシーバ基準クロック。クロックのソースは 156.25 MHz オシレータ (X3) です。
reset_n	入力	リファレンス・デザインのすべてのロジックに対する単一リセット。これは RESET (S4) プッシュ・ボタンに接続されます。

表 8 に、トリプルスピード・イーサネット 0 信号を示します。

信号名	入力 / 出力	説明
rxp_to_the_altera_ethernet	入力	シリアル差動受信インタフェース。これは SFP A (J6) の sfpa_rx_p0 信号に接続されます。
txp_from_the_altera_ethernet	出力	シリアル差動送信インタフェース。これは SFP A (J6) の sfpa_tx_p0 信号に接続されます。
led_an_from_the_altera_ethernet	出力	オート・ネゴシエーション・ステータス。これは USER_LED0 (D16) に接続されます。
led_link_from_the_altera_ethernet	出力	リンク同期ステータス。これは USER_LED2 (D14) に接続されます。
led_col_from_the_altera_ethernet	出力	フレーム送信の衝突検出。これは USER_LED4 (D12) に接続されます。
led_crs_from_the_altera_ethernet	出力	送信および受信パスのキャリア検知。これは USER_LED6 (D10) に接続されます。

表 9 に、トリプルスピード・イーサネット 1 信号を示します。

信号名	入力 / 出力	説明
rxp_to_the_altera_ethernet_1	入力	シリアル差動受信インタフェース。これは SFP B (J7) の sfpb_rx_p0 信号に接続されます。
txp_from_the_altera_ethernet_1	出力	シリアル差動送信インタフェース。これは SFP B (J7) の sfpb_tx_p0 信号に接続されます。
led_an_from_the_altera_ethernet_1	出力	オート・ネゴシエーション・ステータス。これは USER_LED1 (D15) に接続されます。
led_link_from_the_altera_ethernet_1	出力	リンク同期ステータス。これは USER_LED3 (D13) に接続されます。
led_col_from_the_altera_ethernet_1	出力	フレーム送信の衝突検出。これは USER_LED5 (D11) に接続されます。
led_crs_from_the_altera_ethernet_1	出力	送信および受信パスのキャリア検知。これは USER_LED7 (D9) に接続されます。

表 10 に、SFP インタフェース信号を示します。

表 10. SFP インタフェース信号		
信号名	入力 / 出力	説明
out_port_from_the_pio	出力	SFP A シリアル・クロック・ライン。これは SFP A (J6) の sfpa_mod1_scl 信号に接続されます。
bidir_port_to_and_from_the_pio_1	入力 / 出力	SFP A シリアル・データ・ライン。これは SFP A (J6) の sfpa_mod2_sda 信号に接続されます。
out_port_from_the_pio_2	出力	SFP B シリアル・クロック・ライン。これは SFP B (J7) の sfpb_mod1_scl 信号に接続されます。
bidir_port_to_and_from_the_pio_3	入力 / 出力	SFP B シリアル・データ・ライン。これは SFP B (J7) の sfpb_mod2_sda 信号に接続されます。
sfpa_txdisable	出力	SFP A トランスミッタ・ディセーブル。これは SFP A (J6) の sfpa_txdisable 信号に接続されます。
sfpb_txdisable	出力	SFP B トランスミッタ・ディセーブル。これは SFP B (J7) の sfpb_txdisable 信号に接続されます。

TSE リファレンス・ デザインの使用

TSE リファレンス・デザインには、ハードウェア・コンポーネントとソフトウェア・コンポーネントの両方が含まれています。

前提条件

このデザインを動作させるには、以下のハードウェアが必要です。

- Stratix II GX PCI Express 開発ボード
- アルテラ・プログラミング・ケーブル (ByteBlaster または USB-Blaster)
- Finisar アクティブ銅線用の SFP (FCMJ-8521-3) モジュール x2
- ストレート接続の CAT5 イーサネット・ケーブル・アセンブリ

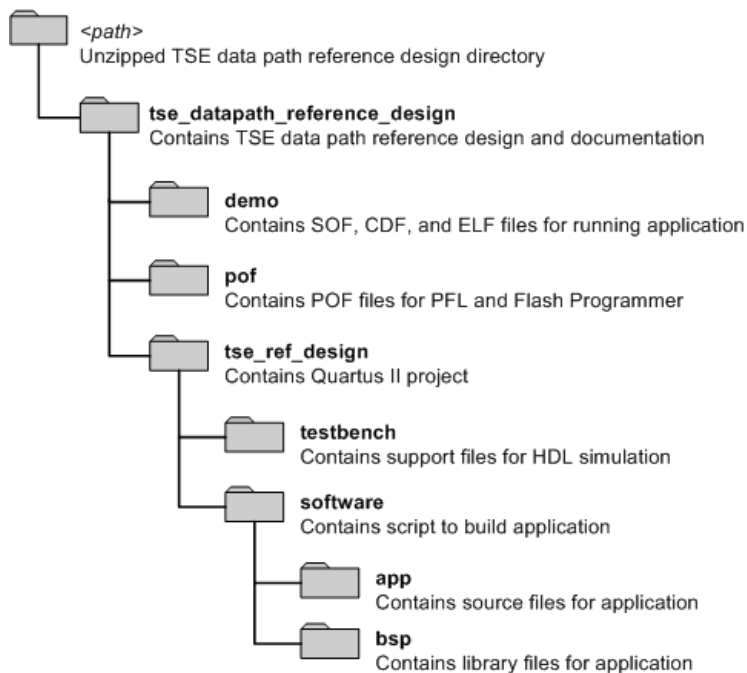
また、ワークステーションに以下のソフトウェアがインストールされている必要があります。

- Quartus® II ソフトウェア v7.2 以降
- Nios II EDS v7.2 以降

ディレクトリ構造

図 8 に、`<path>` ディレクトリにリファレンス・デザイン (`tse_datapath_reference_design.zip`) を解凍した後にできるディレクトリ構造を示します。

図 8. ディレクトリ構造



ハードウェア設定

この項では、開発ボードやイーサネットのコンフィギュレーションなど、必要なハードウェアのセットアップ方法について説明します。

ボードのセットアップ

開発ボードをセットアップするには、以下のステップを実行します。

1. Finisar 1000BASE-T 銅線用の SFP モジュールを、開発ボードの SFP_A (J6) スロットと SFP_B (J7) スロットに差し込みます。モジュールからカチッと音がするのを確認します。

2. “ポイント・ツー・ポイント” 接続を使用するか、ハブまたはスイッチを使用して、Finisar PHY モジュールの両方のポートを接続します。どの接続方式を使用するかは、「[イーサネットのコンフィギュレーション](#)」を参照してください。
3. アルテラ・プログラミング・ケーブルを JTAG 接続ポート (J5) に接続します。
4. Stratix II GX 開発ボードの電源アダプタをボード (J3) に接続します。
5. 開発ボードの電源スイッチ (SW1) をオンにして電源を入れます。

イーサネットのコンフィギュレーション

テスト・セットアップに使用するネットワーク・コンフィギュレーションとして、“ポイント・ツー・ポイント” か、スイッチまたはハブのいずれを使用するかを選択する必要があります。“ポイント・ツー・ポイント” コンフィギュレーションでは、イーサネット・ケーブル・アセンブリを利用して、一方のイーサネット・ポートを他方のイーサネット・ポートに直接接続します。この接続構成はシンプルで、信頼性が高く、エラーが発生する可能性が低いいため、初めて使用する場合に推奨されます。

ただし、スイッチまたはハブを利用して両方のポートを接続する場合は、以下のパラメータを設定してテストが正常に行われるようにします。

- **Ethernet Speed**— テストを実施するネットワーク速度が、スイッチまたはハブで確実にサポートされるようにします。
- **Network Topology**— アクティブ・ネットワーク (LAN) に接続する場合、TSE MAC に割り当てる MAC アドレスが、ネットワーク上の他の MAC アドレスと競合しないようにします (31 ページの「[TSE MAC Menu](#)」を参照)。
- **Network Traffic**— アクティブ・ネットワーク (LAN) に接続する場合は、このリファレンス・デザインによってネットワークに大量のトラフィックがあふれて、通常のネットワーク機能が中断される可能性があります。

どのネットワーク接続方法を使用するかを決定し、Finisar PHY モジュールの両方のイーサネット・ポートが適切に接続されるようにする必要があります。

ハードウェア・デザインによるボードのコンフィギュレーション

次のステップでは、デザイン用のハードウェア・イメージを使用して Stratix II GX FPGA をプログラムします。以下のステップを実行します。

1. Nios II Command Shell インスタンスを開きます。Windows ワークステーションの [スタート] メニューで [プログラム]、Altera、Nios II EDS <version> を順にポイントし、Nios II <version> Command Shell をクリックします。
2. Nios II Command Shell で、tse_ref_design_top.cdf および tse_ref_design_top.sof ファイルが存在する <path>/tse_datapath_reference_design/demo/ フォルダにディレクトリを変更します。

3. コマンドラインで、以下のように入力します。

```
> quartus_pgm tse_ref_design_top.cdf ↓
```



PC に複数のプログラミング・ケーブルがインストールされている場合は、`--list` オプションを使用して、使用可能なハードウェアのリストを表示します。`--cable=<cable number>` オプションを追加して、正しいプログラミング・ケーブルを指定します。例:

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

`-h` オプションを指定すると、Quartus II Programmer に関する情報が表示されます。例: `quartus_pgm -h`

4. イメージが正常にプログラムされると、[図 9](#) に示す情報が Command Shell に表示されます。

図 9. Nios II Command Shell

```

C:\> Sopc Builder 7.2
/cygdrive/d
[SOPC Builder]$ cd tse_standalone_demo_reference_design/
/cygdrive/d/tse_standalone_demo_reference_design
[SOPC Builder]$ cd demo/
/cygdrive/d/tse_standalone_demo_reference_design/demo
[SOPC 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
[SOPC Builder]$

```

- 開発ボードの SFP ケージに、すでに両方の Finisar 1000BASE-T 銅線用の SFP モジュールが接続されている場合は、USER_LED D13 と USER_LED D14 が点灯し、リンクが同期していることを示します。
- オート・ネゴシエーションが完了してリンクが確立されると、USER_LED D15 と USER_LED D16 が少し遅れて点灯します。

オプションで、POF イメージをフラッシュ・メモリにプログラムすることができます。ボードに電源が投入されたときに、プログラムされた内容が FPGA にロードされます。詳細は、39 ページの「POFによるプログラミング」を参照してください。

これで、ボードはハードウェア・イメージを使用して正常にコンフィギュレーションされました。

ソフトウェア設定

ハードウェアのセットアップが完了したら、以下の項に示すとおり、ソフトウェアを設定できます。

ベンチマーク・アプリケーションのダウンロードと実行

次のステップでは、事前にコンパイルされたベンチマーク・アプリケーションをシステム・メモリにダウンロードして、Nios II プロセッサの動作を開始することです。以下のステップを実行します。

- 引き続き Nios II Command Shell を使用します。main.elf ファイルは、同じ <path>/tse_datapath_reference_design/demo/ ディレクトリにあります。
- コマンドラインで、以下のように入力します。

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

このコマンドは、まず Nios II プロセッサのイメージ・ファイルをシステム・メモリにダウンロードし、ついでプロセッサを再起動してコードの実行を開始します。また、ターミナル・アプリケーションも実行を開始するため、ソフトウェア・アプリケーションと通信できるようになります。



PC に複数のプログラミング・ケーブルがインストールされている場合は、--list オプションを使用して、使用可能なハードウェアのリストを表示します。--cable=<cable number> オプションをコマンドに追加して、正しいプログラミング・ケーブルを指定します。例：

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

-h オプションを指定すると、Nios II ダウンロードおよびターミナルに関する情報が表示されます。例：

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

図 10 に、Command Shell に表示される情報を示します。

図 10. ベンチマーク・アプリケーション

```

SOPC Builder 7.2
/cydrive/d/tsc_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>
    
```

図 10 に示す情報が表示されたら、ベンチマーク・アプリケーションは正常に起動しています。

ソフトウェア のクイック・ チュートリアル —システムとの 対話

ベンチマーク・アプリケーションでは、メニュー方式のインターフェースを使用してテストをコンフィギュレーション、実行、および解析します。このメニューを図 11 に示します。

図 11. アプリケーション・テスト・メニュー

```

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) 1000000000 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_

```

この項では、シンプルなテストを実行してメニュー・システムの使用方法に関する簡単なチュートリアルを説明します。

このチュートリアルでは、100 Mbit リンク（半二重）を使用するようにシステムをコンフィギュレーションします。このテストでは、それぞれ長さが1000 バイトのパケットを100,000 個送信するようにコンフィギュレーションされています。このテストをコンフィギュレーションして実行するには、以下のステップを実行します。

1. リンク速度が 100 Mbit/sec になるようにコンフィギュレーションします。コマンドラインで、以下のように入力します。

```
> 1b ↵
```

コマンドが処理され、**Link Speed Value** フィールドが 100 に設定されます。後続のすべてのコマンドで、報告される Value フィールドがこの値に応じて変化することを確認してください。

2. リンクの通信方式を半二重にコンフィギュレーションします。コマンドラインで、以下のように入力します。

```
> 2a ↓
```

3. パケット・サイズが 1000 バイトになるようコンフィギュレーションします。コマンドラインで、以下のように入力します。

```
> 3e 1000 ↓
```

4. これでシステムがコンフィギュレーションされます。コマンド・プロンプトで以下のように入力して、テストを実行します。

```
> 6a ↓
```

テストを実行すると、コンソール・ウィンドウに一連のステータス・メッセージが表示されます (図 12)。

図 12. テスト・ステータス・メッセージ

```

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|

```

これらのメッセージはテストの進捗状況を示しています。各行の上の矢印は送信されたパケット数を表し、右の余白はテストの完了を示します。スループット・レート of 計算結果も出力され、1 秒間あたりのパケット数の概算値が示されます。テスト中は、リンクの稼働状況を示す USER_LED D9 と USER_LED D10 の両方が点灯している必要があります。テストが完了すると、コマンド・シェルに Report Menu が表示されます (図 13)。このテストが完了した後、新しいテストを実行する場合は、コマンド・プロンプトで 2a と入力して Test Menu に変更します。Test Menu から、コマンド・プロンプトで 6a と入力して同じテスト・コンフィギュレーションを再実行するか、新しいテストを開始する前に新しい設定をリコンフィギュレーションすることができます。

図 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: 0xee1122334450
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 ソフトウェア の説明

リファレンス・デザインでは、Nios II プロセッサを使用して、システムのベンチマーク・テストをコンフィギュレーションおよび実行するソフトウェア・アプリケーションを実行します。この項では、このアプリケーションの機能について説明します。

メニュー方式のインタフェース

このアプリケーションは、メニュー方式のユーザー・インタフェースを備えており、これを使用してテストをパラメータ化して、ハードウェア上で実行できます。メニュー・インタフェースでオプションを選択するときは、メニュー項目を選択し、次にオプション（ある場合）を選択して、**Enter** を押します。メニュー・オプションには、選択したオプションに値を指定できるものもあります。数値を必要とするユーザー指定オプションの場合、数値は 16 進値（例：0x1000）または 10 進値（例：4000）として指定できます。

すべてのテスト・パラメータにデフォルト値があります。一般に、デフォルトのテストを開始するには、コマンド・プロンプトでどの設定も変更しないで 6a と入力します。

以下の項では、使用可能なメニューについて説明します。

Test Menu

Test Menu は、システムでベンチマーク・テストをコンフィギュレーションおよび実行するのに使用します（[図 11](#) を参照）。以下のテスト・パラメータを制御できます。

- **Link Speed**— イーサネット・リンクの速度を指定します。10、100、または 1000 Mb/s/sec を設定できます。
- **Link Configuration**— リンクの通信モードを指定します。モードは全二重または半二重に設定できます。（ただし、1000 Mb/s/sec リンクの場合は全二重しかサポートされていません。）
- **Packet Length**— リンク上で送信するパケットの長さを指定します。固定サイズまたはランダム・サイズを設定できます。固定パケット長を指定する場合、許容範囲は 24 ~ 9600 バイトです。ランダム・オプションを選択した場合、生成されるパケット長の範囲は 24 ~ 1518 バイトです。
- **Number of Packets**— テストのために送信するパケット数を指定します。数値は 32 ビットの符号なし整数です。有効範囲は 1 ~ 2^{32} パケットです。
- **Packet Data**— パケットのペイロード部分で送信するデータ型を指定します。データ型にはインクリメントまたはランダムのいずれかを使用できます。インクリメント型を指定した場合、データ・ペイロードはパケットごとに増分される 32 ビット符号なし整数で構成されます。この整数は 2^{32} に達すると 0 にリセットされます。ランダム・データ型を選択した場合は、データ・ペイロードで使用中の擬似ランダム 32 ビット符号なし整数 ($2^{23}-1$) になります。擬似ランダム整数は、**Seed value** メニュー・オプションを使用して生成されます。

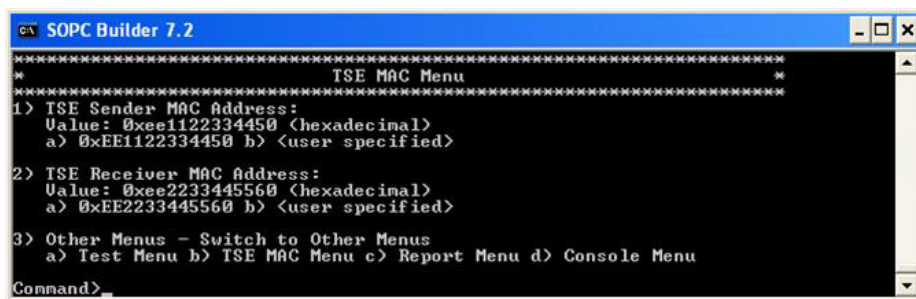
- **Seed Value—Seed value** メニュー・オプションは、**Packet Data** メニュー・オプションがランダムに設定されている場合のみ表示されます。これはランダム整数の生成を指定します。シード値は 46 ビットの 16 進数です（この長さを超える値は切り捨てられる）。
- **Test Control**—テストの開始を制御します。テストは開始すると、送信パケット数が**Number of Packets**パラメータに指定された値と等しくなるまで実行されます。あるいは、**Enter** を押して手動でテストを中断します。テストが停止すると、ビューは **Report Menu** に変わり、テスト結果を表示できます。テストの実行中は、スループットの概算値が表示されます（パケット数 / 秒で出力）。

TSE MAC Menu

TSE MAC Menu は、送信側および受信側 TSE MAC の MAC アドレスをコンフィギュレーションするのに使用されます（図 14 を参照）。

- **TSE Sender MAC Address**—送信側の MAC アドレスを指定します。MAC アドレスは 12 桁の 16 進数です。リファレンス・デザインで使用するデフォルトの TSE Sender MAC アドレスは、0xEE1122334450 です。この MAC アドレスは、ネットワーク・スイッチを介して接続されているネットワーク上の他のアドレスと競合しない限り、テストの目的でユニキャストまたはマルチキャスト・アドレスにコンフィギュレーションすることができます。
- **TSE Receiver MAC Address**—受信側の MAC アドレスを指定します。MAC アドレスは 12 桁の 16 進数です。リファレンス・デザインで使用するデフォルトの TSE Receiver MAC アドレスは、0xEE2233445560 です。この MAC アドレスは、ネットワーク・スイッチを介して接続されているネットワーク上の他のアドレスと競合しない限り、テストの目的でユニキャストまたはマルチキャスト・アドレスにコンフィギュレーションすることができます。

図 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

Report Menu には、システムで実行したテストのステータスが表示されます。概念上、Report Menu システムは、レポートを保管する 10 個のスロットを提供します。レポートは FIFO (First-In First-Out) 形式で保管されます。10 番目のレポート・スロットが使用されると、最も古いレポート・データが廃棄されて新しいレポート用の場所が確保されます。

図 15 に、Report Menu を示します。

図 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: 0xee11223344550
TSE Receiver MAC: 0xee22334455660
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 hits/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 Menu - Switch to Other Menus
   a) Test Menu b) TSE MAC Menu c) Report Menu d) Console Menu

Command>

```

Report Menu には以下の情報が表示されます。

- **Number**—完了したテストに対応する整数。最初のテストは 1 で、以後、新しいテストが開始されるたびにこの値が増分されます。
- **Status**—テストのステータスを示します。**Completed**、**Interrupted**、または **Error** のいずれかです（場合によって、追加ステータス情報も出力される）。
- **TSE Sender MAC**—TSE MAC 送信側の MAC アドレス。
- **TSE Receiver MAC**—TSE MAC 受信側の MAC アドレス。
- **Link Speed**—イーサネット・リンク速度（10、100、または 1000 Mbits/sec）と通信モード（half または full）が表示されます。
- **Packets to Send**—テストで送信するようにコンフィギュレーションされているパケット数を表示します。パケット長とデータ型も表示されます。データ型が **random** の場合は、シード値が表示されます。

- **Packets Sent**— テスト中にイーサネット・パケット・ジェネレータが実際に送信したパケットの総数を表示します。システムでエラーが発生した場合、実際の値は要求された値よりも少ない可能性があります。また、送信されたパケットのスループット測定値が、パケット数 / 秒の形式で表示されます。
- **Packets Received**— テスト中にイーサネット・パケット・モニタが受信したパケットの総数を表示します。有効パケット数とエラー・パケット数も表示されます（パケット総数は、受信した有効パケット数とエラー・パケット数を合計した値でなければなりません）。
- **Bytes Received**— イーサネット・パケット・モニタが受信した総バイト数を表示します。このバイト数は、パケット全体の長さ（ヘッダとデータを加えた長さ）で計算されています。また、バイト・スループット・レートがビット数 / 秒で表示されます。



テストが完了するまでに要する合計時間は、イーサネット・パケット・モニタにあるサイクル・カウント・レジスタに基づきます。このサイクル・カウント・レジスタは、テストの開始時に 0 に設定され、最初のパケットの受信後にシステム・クロック周波数レートでインクリメントし始めます。サイクル・カウント・レジスタは、モニタがすべてのパケットを受信すると自動的にカウントを停止し、またテストが手動で中断された場合も停止します。時間はサイクル・カウント・レジスタ値をシステム・クロック周波数で除算して計算されます（計算には、浮動小数点演算を使用）。

この時間計算方法は非常に正確ですが、サイクル・カウント・レジスタの開始と停止を行うために、テスト中に最初のパケットと最後のパケットをモニタが受信する必要があります。モニタがこれらのパケットを受信しなかったり、イーサネット・パケット・モニタで大きなエラーが発生した場合、レポートされる時間は不正確になります。

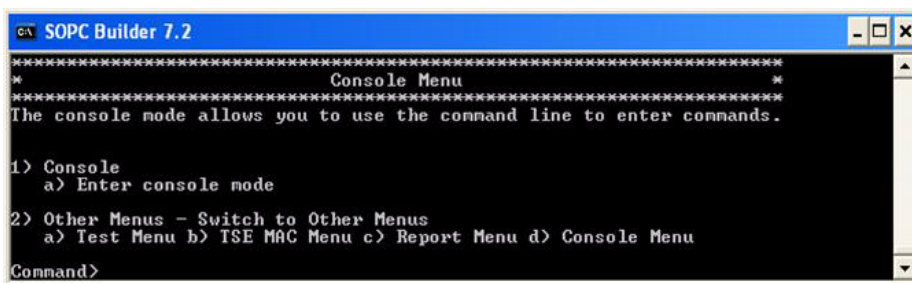
- **Report Control**— レポートの選択を制御します。現在閲覧中のテスト番号とアクセス可能な最初および最後のレポート番号が表示されます。**next** および **previous** コントロールを使用するか、レポート番号を入力して、レポートを閲覧できます。

Console Menu

Console Menu には、コンソール・ベースのシステムに対するインタフェースがあります。Test Menu、TSE MAC Menu、および Report Menu のすべてのコマンドと機能をこのコンソール・インタフェースから使用できます。また、このコンソールから任意のメモリ位置を読み出したり、そこに書き込むこともできます。

図 16 に Console Menu を示します。

図 16. Console Menu



Console Menu には以下のメニュー項目があります。

- **Console**— コンソール・アプリケーションの起動を制御します。コンソールでの操作を終了し、メニュー・システムに戻るには、**x** キーを押してから、**Enter** を押します。

コンソールでの操作

このコンソールはコマンドライン入力モードで動作し、**Enter** を押すとコマンドが終了します。ほとんどのコマンドは追加引数も受け入れ、引数はスペースで区切られます。

このコンソールは、表 11 にリストするコマンドを認識します。ほとんどのコマンドは、メニュー・システムの機能と同じ機能を実行し、スループット・テストを実行するための代替インタフェースとなります。特定のコマンドの詳細情報を参照するには、コマンド名を入力して、その後に「?」キーを入力します（コマンド名との間にスペースは不要）。

コマンド	用途	同じ機能を持つメニュー	
		メニュー	場所
speed	リンク速度	Test	Link Speed
duplex	リンクの通信	Test	Link Configuration
plen	パケット長	Test	Packet Length
pnum	パケット数	Test	Number of Packets
pdata	パケット・データ	Test	Packet Data
seed	シード値	Test	Seed Value

表 11. コンソール・コマンド (2 / 2)			
コマンド	用途	同じ機能を持つメニュー	
		メニュー	場所
start	テストの開始	Test	Start Test
macaddr_send	送信側 MAC アドレス	TSE MAC	TSE Sender MAC Address
macaddr_recv	受信側 MAC アドレス	TSE MAC	TSE Receiver MAC Address
report	レポートの表示	Report	—
reportn	レポートをキューに投入	Report	Report Control
oreg	メモリ位置の読み出し / 書き込み (オフセット付き)	—	—
reg	メモリ位置の読み出し / 書き込み	—	—
test	テスト・パラメータを一覧表示	Report	—
map	メモリ・マップを表示	—	—
?	コマンド情報	—	—
x	コンソールを終了	—	—

付録: ソフトウェア・ アプリケーションの実行

ソフトウェア・アプリケーションをコンパイルするには、Nios II プロセッサ・エンベデッド開発スイート v7.2 以降をインストールする必要があります。

このソフトウェア・アプリケーションは C プログラミング言語で記述されており、標準 Makefile スクリプトを使用して簡単に作成できます。Nios II ソフトウェア開発環境には、ソフトウェア・アプリケーションを素早く簡単に作成できるユーティリティが揃っています。

このアプリケーションには、Nios II ソフトウェア開発ユーティリティを使用する bash シェル・スクリプトが含まれています。このスクリプトを実行すると、アプリケーションを生成するための Makefile セットが作成されます。



Nios II CLI またはソフトウェアのビルド・フローについて詳しくは、「Nios II ソフトウェア開発ハンドブック」の「アルテラの開発ツール」の章または「Nios II IDE ヘルプ・システム」のソフトウェア・チュートリアルを参照してください。

ソフトウェア・アプリケーションの作成

この項では、ソフトウェア・アプリケーション用の標準 **Makefile** として、1つはシステム・ライブラリ用もう1つはメイン・アプリケーション用の **Makefile** の作成方法を説明します。これらの **Makefile** が作成されたら、ソフトウェア・アプリケーションをコンパイルしてターゲット・ハードウェア上で実行できます(22ページの「ハードウェア設定」を参照)。

Makefile を作成するには、以下のステップを実行します。

1. **Nios II SDK Command Shell** を開きます。Windows ワークステーションの [スタート] メニューで [プログラム]、**Altera**、**Nios II EDS <version>** を順にポイントし、**Nios II <version> Command Shell** をクリックします。

2. ディレクトリを **tse_datapath_reference_design.zip** プロジェクト・ファイルを解凍したフォルダに変更し、**software** ディレクトリを開きます。

3. **software** ディレクトリで、以下のコマンドを入力します。

```
> sh create-software ↓
```

このスクリプトでシステム・ライブラリとアプリケーションが作成されると、一連のメッセージがコンソールに送信されます。

4. プロジェクト用のシステム・ライブラリを作成するには、まずディレクトリを **bsp** ディレクトリに変更します。コマンドラインで、以下のように入力します。

```
> make ↓
```

このコマンドは、システム・ライブラリの **Makefile** を実行し、それによってコンパイル済みのシステム・ライブラリが作成されます。

5. アプリケーションを作成するには、ディレクトリを **app** ディレクトリに変更します。コマンドラインで、以下のように入力します。

```
> make ↓
```

このコマンドは、アプリケーションを実行する実行可能な **ELF** ファイルを作成します。

これでアプリケーションのコンパイルが完了しました。これでターゲット・ハードウェア上でアプリケーションを実行できます。

アプリケーション構造

このアプリケーションは、主要機能部分ごとに区分された半階層構造で構成されています。

メニュー / コンソール

メニューまたはコンソール・インタフェースを通じてアプリケーションと対話します。アプリケーションの実行を開始すると、**main** エントリ・ポイントにユーザーが操作できる以下の4つのメニューが作成されます。

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

これらの各メニューは、**elem.c** で提供される簡単なメニュー・ライブラリを使用します。これらのメニューを使用すると、直観的な方法でテスト・コントロールを操作できます。

テスト・コントロール

テストのコンフィギュレーション、開始、および停止は、テスト・コントロール API (**test_control.h** に宣言されている) により、**tseStartTest**、**tsePollTest**、および **tseStopTest** 関数を呼び出すことによって行われます。テストを開始する前に、テスト用のすべてのパラメータが **struct gTestStruct** 構造体を通じて指定されます。この構造体に値が供給されると、この構造体は **tseStartTest** 関数を通じてテストを開始するのに使用できます。テストのステータスは **tsePollTest** でポーリングされ、ステータス情報は **struct gTestStruct** 構造体を介して渡されます。テストを停止するには、**tseStopTest** 関数を呼び出します。

テスト・コントロールは次に、ペリフェラル・コントロール機能を直接呼び出します。以下の項では、この呼び出しについて説明します。

モニタ / ジェネレータ / TSE MAC / PHY ペリフェラル

このシステムの主要なハードウェア・コンポーネントは、モニタ、ジェネレータ、TSE MAC、および PHY コンポーネントです。これらのペリフェラルのコントロールは、主に **tse_control.c** で発生します。レジスタ・マクロとハードウェア情報の一部は **hardware_def.h** に定義されています。

また、PHY コンポーネントのソフトウェア・コントロールは、**alt_2_wire.c** にあるシンプルな TWSI API を通じて行われます。この API は、PIO ペリフェラルを介して、TWSI マスタ・ペリフェラルの“ビット・バン”実装を提供します。

ソース・ファイルのレイアウト

表 12 に、アプリケーションで使用するソース・ファイルと、アプリケーションの機能を示します。

表 12. テスト・コントロール	
Source File	用途
test_control.h	テスト・コントロール・ルーチン・ヘッダ
test_control.c	テスト・コントロール・ルーチン
helpers.h	変換ライブラリ・ヘッダ
helpers.c	変換ライブラリ・ルーチン
メニュー / コンソール	
main.c	ソフトウェアへのエントリ・ポイント
test_menu.c	Test Menu の要素
console_menu.c	Console Menu の要素
report_menu.c	Report Menu の要素
tsemac_menu.c	TSE MAC Menu の要素
elem.h	メニュー・ライブラリ・ヘッダ
elem.c	メニュー・ライブラリ
bool_check.h	TRUE と FALSE を定義
モニタ / ジェネレータ / TSE MAC / PHY パリフェラル	
hardware_def.h	コンポーネント情報
alt_2_wire.h	TWSI ライブラリ
alt_2_wire.c	TWSI ライブラリ
tse_control.h	ハードウェア・コントロール・ヘッダ
tse_control.c	ハードウェア・コントロール・ルーチン

テストベンチ・シミュレーション

付属の IP 機能シミュレーション・モデルとテストベンチを使用して、リファレンス・デザインをシミュレーションできます。テストベンチ・ファイルは、プロジェクト・ディレクトリの **testbench** サブディレクトリにあります。サポートされているどのシミュレータでもリファレンス・デザインのシミュレーションに使用できます。テストベンチには、ModelSim シミュレータ用のスクリプトが用意されています。

ModelSim シミュレータによるシミュレーション

ModelSim シミュレータを使用してシミュレーションを実行するには、以下のステップを実行します。

1. ModelSim シミュレータを起動します。
2. 作業ディレクトリを `<path>/tse_datapath_reference_design/tse_ref_design/testbench/` に変更します。
3. 以下のコマンドを入力して、必要なライブラリを設定し、付属の IP 機能シミュレーション・モデルをコンパイルし、付属のテストベンチでシミュレーション・モデルを実行します。

```
>do run_tse_ref_design_tb.tcl ↓
```

ModelSim Transcript ペイン (Main ウィンドウ内) に、実行中の現在のタスクを反映するテストベンチからのメッセージが表示されます。

POF によるプログラミング

リファレンス・デザイン・パッケージには、`<path>/tse_datapath_reference_design/pof/` フォルダに POF ファイルが含まれています。ただし、必要に応じて、SOF から POF を生成して開発ボード上のフラッシュをコンフィギュレーションすることができます。



必ず SOF を検証してから POF に変換してください。これには、SOF を FPGA にロードし、テストを実行してロードしたコンフィギュレーションが正しく動作することを確認します。

MAX® II フラッシュ・メモリを POF でプログラムするには、以下の作業を行います。

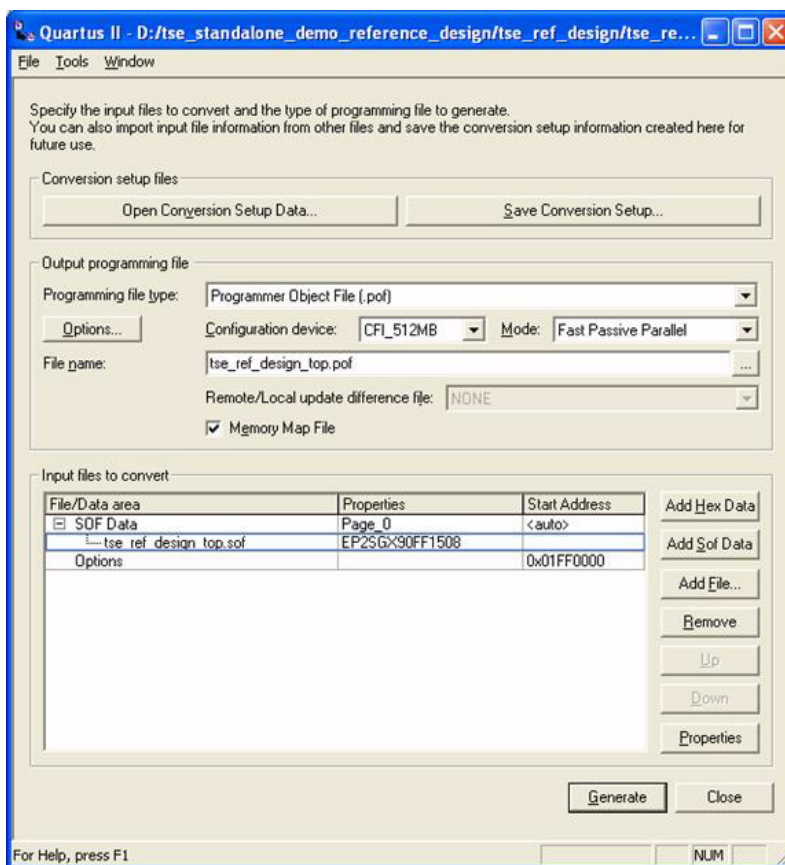
- SOF を変換して POF を生成する。
- フラッシュ・メモリをプログラムする。

以下のステップを実行して、POF を生成します。

1. Quartus II ソフトウェアを起動します。
2. File メニューの **Convert Programming Files** をクリックします。

3. 以下のパラメータを入力します (図 17)。
 - プログラミング・ファイル・タイプ: **Programmer Object File (.pof)**
 - コンフィギュレーション・デバイス: **CFI_512MB**
 - モード: **高速パッシブ・パラレル**
 - オプション: コンフィギュレーション・デバイス JTAG ユーザー・コードを **0x1FF0000** に設定
 - ファイル名: **<output file name>.pof**
4. **Add File** をクリックし、追加する SOF ファイルを選択します。
5. **Generate** をクリックします。

図 17. Convert Programming Files ウィンドウ



以下のステップを実行して、フラッシュ・メモリをプログラムします。

1. JTAG (USB-Blaster ケーブル) が開発ボードに接続され、イネーブルされていることを確認します。
2. Programmer ウィンドウで、MAX II デバイスを識別します。
3. MAX II デバイス (EPM570GT100) をダブル・クリックし、<path>/tse_datapath_reference_design/pof ディレクトリにある PFL 準拠の MAX デザイン (altera_siigx_pcie_pfl.pof) を選択します (図 18)。
4. フラッシュ・デバイス (CFI_512MB) をダブル・クリックし、同じディレクトリ内にある付属の POF (tse_ref_design_top.pof) または新たに生成されたフラッシュ・イメージ POF を選択します。
5. EPM570GT100 および CFI_512MB デバイスの **Program/Configure** をオンにします。

図 18. Programming Flash ウィンドウ

	File	Device	Checksum	Usercode	Program/ Configure	Verify
Start	D:/tse_standalone_demo_reference_design/pof/altera_siigx_pcie_pfl.pof	EPM570GT100	002E5272	FFFFFFF	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Stop	-CFM				<input checked="" type="checkbox"/>	<input type="checkbox"/>
Auto Detect	-UFM				<input checked="" type="checkbox"/>	<input type="checkbox"/>
Delete	D:/tse_standalone_demo_reference_design/pof/tse_ref_design_top.pof	CFI_512MB	CCA78073		<input checked="" type="checkbox"/>	<input type="checkbox"/>
Add File...	-Page_0				<input checked="" type="checkbox"/>	<input type="checkbox"/>
	-OPTION_BITS				<input checked="" type="checkbox"/>	<input type="checkbox"/>
	<none>	EP25Gx90	00000000	<none>	<input type="checkbox"/>	<input type="checkbox"/>

6. **Start** をクリックし、フラッシュがプログラムされるまで待ちます。



「Performing verification of type Nominal on device(s)」の後に「Device 1 silicon ID is not ready」が表示された場合 (図 19 に示す) は、開発ボードの nCONFIG ボタン (S1) を押して、プログラミングを開始します。このメッセージは、ボード上のフラッシュがすでにプログラムされている場合に表示されます。

図 19. Device Not Ready メッセージ

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

参考資料

このアプリケーション・ノートでは、以下のドキュメントを参照しています。

- [「CRC Compiler User Guide」](#)
- [「Finisar FCMJ-8520/8521-3 100BASE-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」](#)

改訂履歴

表 13 に、このアプリケーション・ノートの改訂履歴を示します。

日付およびドキュメント・バージョン	変更内容	概要
2008 年 1 月 v1.0	初版	—



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

Copyright © 2007 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.

