

はじめに

Altera® RapidIO の相互運用性リファレンス・デザインは Arria II GX デバイス上でコンフィギュレーションされるアルテラの RapidIO MegaCore® ファンクションと Texas Instruments TMS320TCI6488 通信インフラストラクチャ・デジタル信号プロセッサ (TI 6488 DSP または TI 6488) の間のサンプル・インタフェースを提供します。アルテラは TI 6488 でアルテラの RapidIO MegaCore ファンクションのインストールと動作を示すために、このリファレンス・デザインを提供します。このデザインにより、アルテラ FPGA への RapidIO MegaCore ファンクションの統合を評価できます。

基本的な相互運用性を説明することに加えて、すべてのサポートされているパケット・サイズに対して、デザインにはすべてのデータ・レートで利用リンクを測定するためのサポートが含まれています。統計サポートは、転送のために Quartus II ソフトウェア v9.0 で利用可能な 3 つのポー・レートのための Serial RapidIO Arria II GX デバイスから TI 6488DSP へのリンクに最適なペイロード・サイズを決定することができます。TI 6488 は 1x リンク・モードのみをサポートするので、デザインは 1x モードのみをテストします。

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。

- TI 6488 をターゲットにする以下の RapidIO トランザクション・タイプを開始するよう RapidIO MegaCore ファンクションを指示。
 - NWRITE
 - NWRITE_R
 - SWRITE
 - NREAD
 - Maintenance Write
 - Maintenance Read
 - Doorbell メッセージ
- いずれかの 1x RapidIO MegaCore バリエーションで FPGA をコンフィギュレーション。包括的なデザイン・バリエーションのセットには、すべてのデータ・レート (1.250、2.500、と 3.125 gigabaud (GBaud)) およびサポートされたパケット・サイズでの 1x モードが含まれている。
- RapidIO MegaCore ファンクションと TI 6488 DSP の両方によって開始されたトランザクションを含む両方向に同時のトラフィックのテストをサポート。
- 収集スループット統計のサポートを含む。
- 自動データ整合性のチェックのサポートを含む。

このアプリケーション・ノートでは RapidIO 相互運用性リファレンス・デザインのインストールおよび実行方法を示します。それは、このデザインに使用されるシステムおよびハードウェアとソフトウェアの使用方法について説明し、そしてこのデザインで実行されたテストの性能を報告します。このアプリケーション・ノートには以下の項が含まれています。

- 「概要」
- 「リファレンス・デザインの概要」

- 3 ページの「機能の説明」
- 8 ページの「リファレンス・デザインの使用」
- 38 ページの「性能の概要」
- 40 ページの「デザインの制限」
- 40 ページの「参考資料」
- 41 ページの「改訂履歴」

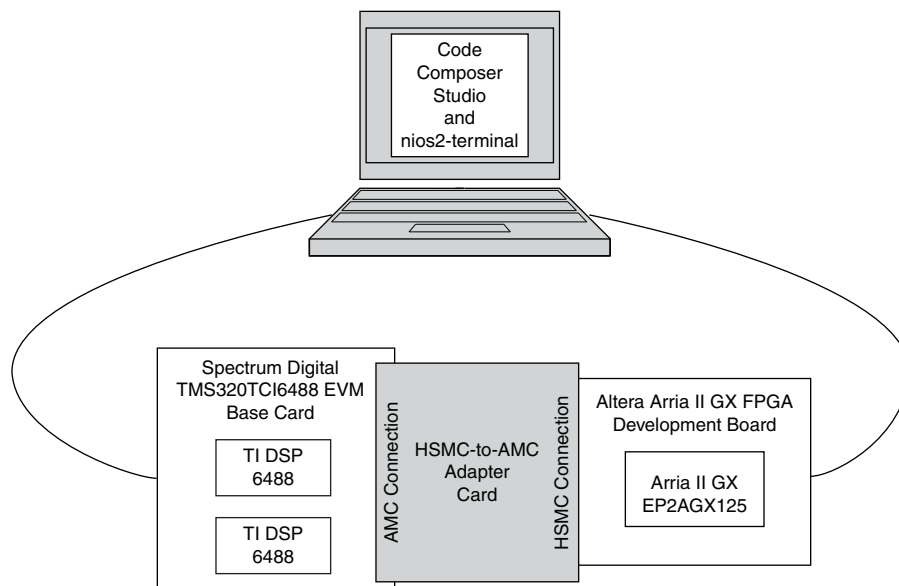
概要

リファレンス・デザインは、リファレンス・デザイン・インタフェース回路を通してアルテラの RapidIO MegaCore ファンクションを TI 6488 DSP に接続するサンプル・アプリケーションです。このデザインはアルテラの Arria II GX FPGA 開発ボードおよび TMS320TCI6488 Evaluation Module (EVM) を使用しています。Arria II GX FPGA 開発ボードには Arria II GX EP2AGX125 デバイスが含まれています。Spectrum Digital, Inc.(www.spectrumdigital.com) は TMS320TCI6488 EVM をプログラムおよびモニタするために、Texas Instruments CCS (Code Composer Studio) IDE (Integrated Development Environment) を含む TMS320TCI6488 EVM を提供します。アルテラは FPGA 上のリファレンス・デザインの各バリエーションを実装および Arria II GX FPGA 開発ボード上の EP2AGX125 デバイスをプログラムするソフトウェアを提供します。

リファレンス・デザインの概要

図 1 に、完全なシステムに関する図を示します。

図 1. RapidIO の相互運用性リファレンス・デザインの完全なシステム



TI 6488 DSP 上で動作する C コードは、以下のタスクを実行します。

- TI 6488 内部 PLL をコンフィギュレーション。
- TI 6488 SERDES ブロックをコンフィギュレーション。

- TI 6488 RapidIO コマンドおよびステータス・レジスタ (CSR) をプログラム。
- リンク・パートナ (アルテラ FPGA) により開始されるトランザクションに応答。
- リンク・パートナにライト・トランザクションを開始させ、性能に焦点を合わせる。
- アルテラ FPGA へのライトおよびリード動作を実行し、データ整合性をチェック。
- アルテラ FPGA への Doorbell メッセージを開始。
- DDR2 メモリ・スペースを表示。
- RapidIO リンクのステータスをチェック。

nios2 ターミナル・セッションでアルテラ FPGA 上で以下の等価なタスクを実行できます。

- RapidIO CSR をプログラム。
- リンク・パートナ (TI DSP カード) へのトランザクションを開始。
- リンク・パートにより開始されるトランザクションに応答。
- リンク上のデータ・スループットをモニタ。
- リンク・ステータスをモニタ。

機能の説明

RapidIO MegaCore ファンクション v9.0 の各データ・レートには 1 つのバージョンのデザインが使用できます。1x 1.250 Gbaud バリエーションのデザインはダイレクト・メモリ・アクセス (DMA) を使用して、ローカル・メモリ・ブロックから RapidIO MegaCore I/O ライト Avalon-MM スレーブ・ポートまでデータを転送します。他の 2 つの RapidIO MegaCore ファンクション・バリエーションのデザイン (2.500 および 3.125 Gbaud での 1x バリエーション) は Avalon® Memory-Mapped (Avalon-MM) マスター・ポートを持つパケット・ジェネレータを使用して、I/O バーストを生成し、それらを RapidIO MegaCore I/O ライト Avalon-MM スレーブ・ポートに転送します。

デザインは 1x 1.250 Gbaud デザイン・バリエーションにのみ DMA 手法を使用して、ライン・レート・トラフィックでリンクを飽和状態にすることができます。パケット・ジェネレータ手法で、デザインは RapidIO プロトコルの他のバリエーションのためのリンクを飽和状態にすることができます。

以下の項はすべてのデザイン・バリエーションの TI 6488 DSP ファンクションやクロックと同様に、デザインの DMA ベース・バージョンとデザインのパケット・ジェネレータ・ベース・バージョンについて説明します。

FPGA デバイスに実装される DMA ベース RapidIO デザイン

図 2 に、アルテラ FPGA に実装される DMA ベース RapidIO の相互運用性リファレンス・デザインのトップレベルのブロック図を示します。

図 2. DMA ベース・リファレンス・デザインのブロック図

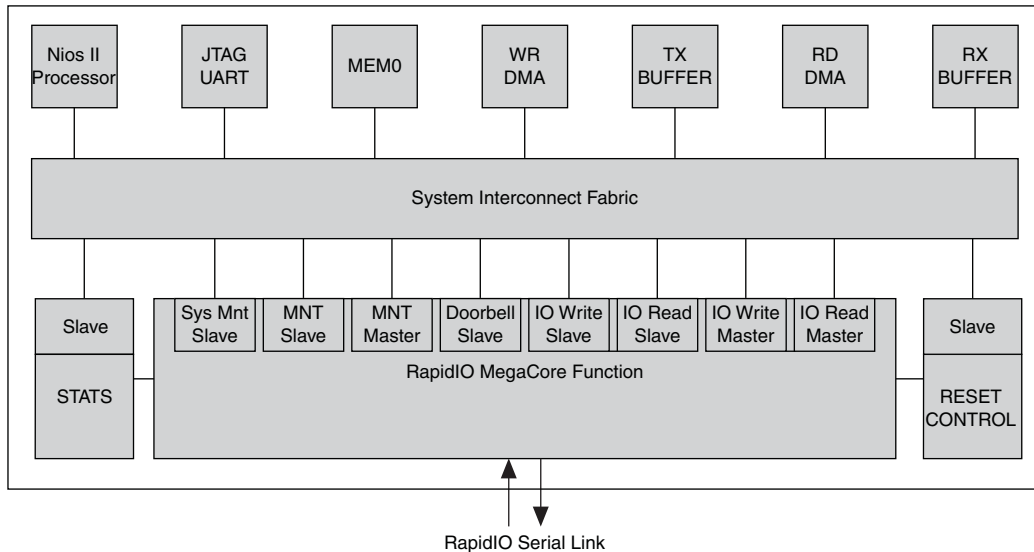


図 2 のシステムは、SOPC Builder を使用して設計されています。以下の項では、リファレンス・デザインにおけるメイン・システム・コンポーネントの役割について説明します。

Nios II エンベデッド・プロセッサ

FPGA デバイスで Nios® II エンベデッド・プロセッサを実装できます。プロセッサはこのリファレンス・デザインに含まれている RapidIO ドライバ・ソフトウェアを実行します。このドライバを使用して、RapidIO MegaCore ファンクションをプログラムして、それが実行するトランザクションを制御することができます。リモートのリンク・パートナーに対してトランザクションの送受信を行うように RapidIO MegaCore ファンクションをプログラムすることができます。

JTAG UART

このコンポーネントは、FPGA に nios2 ターミナルと通信するメカニズムを提供します。nios2 ターミナルは RapidIO ドライバへのユーザー・インタフェースです。

MEMO メモリ・セグメント

このメモリは実行可能なプログラムを格納します。このリファレンス・デザインのプログラム・コードは RapidIO ドライバです。プログラムがコンパイルされると、このメモリに格納されて、Nios II エンベデッド・プロセッサによって実行されます。

WR DMA

この DMA コンポーネントは TX BUFFER から RapidIO MegaCore ファンクション I/O ライト Avalon-MM スレーブ・ポートにデータを転送します。

TX BUFFER メモリ・セグメント

このメモリは、RapidIO MegaCore ファンクション I/O ライト Avalon-MM スレーブ・ポートに転送されるデータを保存します。WR DMA コンポーネントでは異なる RapidIO ライト・トランザクションにペイロードのためのデータが使用されます。

RD DMA

DMA コンポーネントは RapidIO MegaCore ファンクション I/O リード Avalon-MM スレーブ・ポートから RX BUFFER メモリまでデータを転送します。このメモリは RapidIO MegaCore ファンクションにより開始される NREAD トランザクションをサポートします。RD DMA コンポーネントはリモート処理エンドポイントまたはリンク・パートナーからデータ・リードを受け取って、RX BUFFER メモリに格納します。

RX BUFFER メモリ・セグメント

このメモリはリモート処理エンドポイントまたはリンク・パートナーからデータ・リードを格納します。RD DMA コンポーネントは、このメモリにデータを格納します。

このメモリはリモート処理エンドポイントによりアドレスを指定でき、このメモリ・セグメントから読み出し/書き込みが可能です。RX BUFFER メモリ・セグメントはリモート処理エンドポイントで開始される以下のトランザクションに修理を行って、FPGA をターゲットにします。

- NWRITE
- SWRITE
- NWRITE_R
- NREAD


バッファは指定可能なメモリの 64 キロバイトを持っています。

STATS

このカスタム SOPC Builder コンポーネントは以下の統計を維持します。

- TX スループット
- RX スループット
- 送受信パケット
- 送受信バイト
- 受信パケット
- 受信バイト
- 受け入れないパケット
- 取り消したパケット
- 再試行したパケット
- CRC エラー付きパケット
- トランスポート層により破棄されたパケット
- シンボル・エラー
- キャラクタ・エラー

サンプル・ウィンドウは期限が切れるたびに、割り込みをアサートするようにこのコンポーネントにプログラムできます。割り込みに対応し、ドライバは統計情報カウンタを読み込んで、それらを nios2 ターミナル・セッションに表示できます。

 1 × 1.250 Gbaud RapidIO MegaCore バリエーションは双方向のデータ整合性のテストを動作している間に、統計がダイナミックに表示されることはできません。この RapidIO バリエーションの双方向のデータ整合性のテスト・ランから統計を表示するために、テスト・ランが完了してから dit_stats を実行することが必要です。

RESET CONTROL

このカスタム SOPC Builder コンポーネントにより、ユーザーは RapidIO MegaCore ファンクションへのソフト・リセットを実行できます。

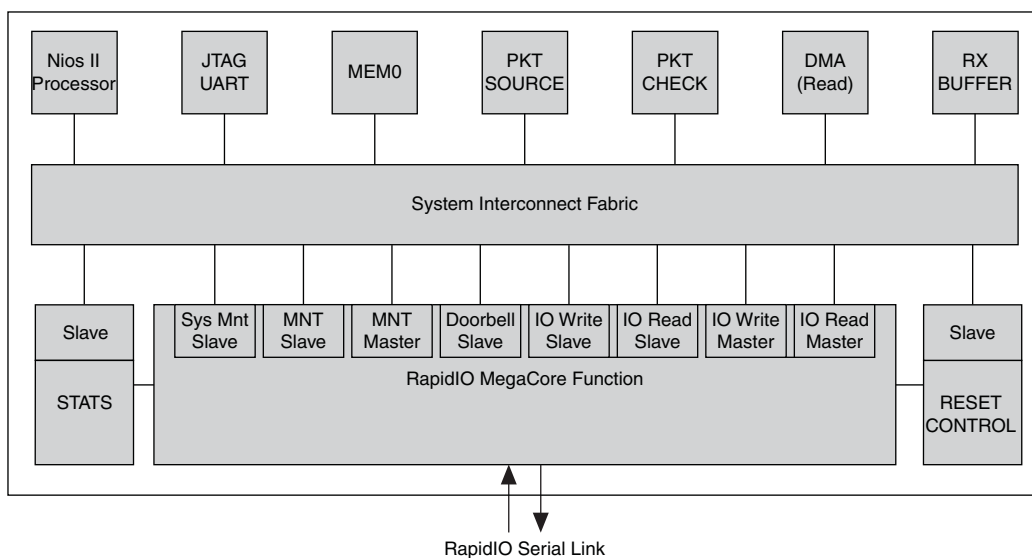
RapidIO MegaCore ファンクション

RapidIO MegaCore ファンクションはこのリファレンス・デザインで主なコンポーネントです。このコンポーネントはリンク・パートナ付き RapidIO リンクを確立しました。また、Avalon-MM インタフェースに送られたトランザクションに対応する RapidIO トランザクションに変換して、RapidIO シリアル・リンク上で送信します。それは RapidIO シリアル・リンクからの RapidIO トランザクションを I/O バースト転送に変換して、これらのバースト転送に対応する Avalon-MM スレーブ・ポートまたはマスター・ポートに提示します。

FPGA デバイスに実装されるパケット・ベース RapidIO デザイン

図 3 に、アルテラ FPGA に実装されるパケット・ジェネレータ・ベース RapidIO の相互運用性リファレンス・デザインのトップレベルのブロック図を示します。このデザインのバージョンはカスタム Avalon-MM トラフィック・ジェネレータに基づいています。

図 3. パケット・ジェネレータ・ベース・リファレンス・デザインのブロック図



pkt_source および pkt_check コンポーネントは DMA ベース・デザインではなく、パケット・ジェネレータ・ベース・デザインに表示されます。他のすべてのシステム・コンポーネントはデザインの DMA ベース・バージョンとパケット・ジェネレータ・ベース・バージョンで同様に機能します。

pkt_source コンポーネントは I/O バースト転送を発生させることができる Avalon-MM マスター・ポート付きカスタム SOPC Builder のコンポーネントです。このデザインでは転送が RapidIO MegaCore I/O ライト Avalon-MM スレーブ・ポートに送られます。

データ完全性のテストの間に、送信する各ライト・パケットに対して、pkt_source コンポーネントは pkt_check コンポーネントのパケット・ディスクリプタ FIFO にパケット・ディスクリプタを押します。pkt_check コンポーネントは送信したパケットのライト・アドレスに NREAD トランザクションを開始して、検索されるリード・データを期待されるデータと比較します。

pkt_check コンポーネントはバースト・リードを RapidIO MegaCore I/O リード Avalon-MM スレーブ・ポートに発生させることができる Avalon-MM マスター・ポート付きカスタム SOPC Builder のコンポーネントです。このコンポーネントにはパケット・ディスクリプタ FIFO が含まれます。pkt_check コンポーネントは FIFO が空でない場合と検出されるとき、パケット・ディスクリプタが FIFO からプルされて、RapidIO リード Avalon-MM スレーブ・ポートにリード・バーストを開始します。リード・バーストは、パケット・ディスクリプが指定するアドレスとデータ・レングスをターゲットとします。リード・データを受信すると、pkt_check コンポーネントは、パケット・ディスクリプタ・データと検索されたリード・データを比較することによって、データ完全性チェックを実行し、そして、FIFO から次のパケット・ディスクリプタをプルします。

TI 6488 DSP ファンクション

Altera RapidIO MegaCore によって開始された NWRITE、NWRITE_R、SWRITE、および NREAD トランザクションは TI DSP カードの上の TI6488DSP に接続された DDR2 SDRAM をターゲットとします。DDR2 SDRAM アドレス・スペースはアドレス 0x80000000 で起動します。データ整合性のテストがアドレス・スペースに書き込まれ、そしてシステムに対してデータ整合性をチェックするために、NREAD トランザクションを実行して、ライト・アドレスの内容を読み出します。

データが DDR2 SDRAM に書き込まれていることを確認するには、CCS IDE での Memory Editor を起動して、アドレス 0x80000000 と以上の内容が読み出されます。

クロック

表 1 に、Avalon システム・クロック・レート、データ・レート、モード、およびトランシーバ基準クロックの関係を示します。

表 1. RapidIO のクロック関係 – TI 6488 の相互運用性リファレンス・デザイン

データ・レート (Gbaud)	モード	ALTGX リファレンス・クロック・レート	Avalon システム・クロック
1.250	1x	125 MHz	40 MHz
2.500	1x	125 MHz	75 MHz
3.125	1x	125 MHz	84 MHz

基準クロックは Arria II GX FPGA 開発ボードで利用可能なクロックによって書き取られます。すべてのデザイン・バリエーションの基準クロックは 125 MHz クロックです。

リファレンス・デザインの使用

以下の項では、リファレンス・デザインの設定および使用方法について説明します。

- 「ハードウェア要件」
- 10 ページの「ソフトウェア要件」
- 10 ページの「リファレンス・デザインのインストール」
- 13 ページの「アプリケーションを実行するための準備」

ハードウェア要件

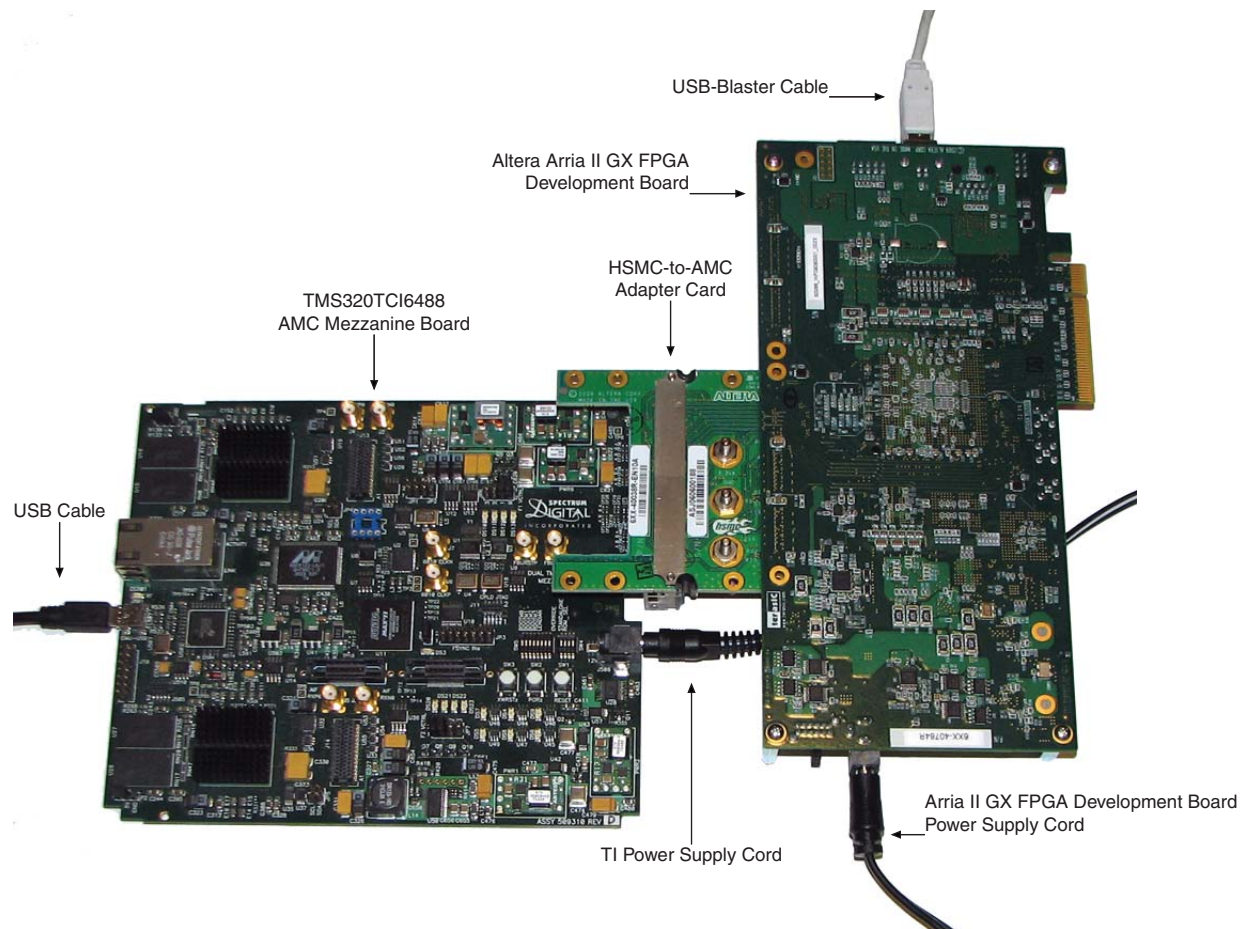
リファレンス・デザインには次のハードウェアが必要です。

- Windows XP オペレーティング・システムを動作しているコンピュータは、Code Composer Studio IDE v3.3 および nios2 ターミナル・セッションを同時に実行可能
- アルテラ Arria II GX FPGA 開発ボード（アルテラ EP2AGX125 デバイスを含む）
- スペクトル・デジタル・デュアル TMS320TCI6488 AMC メザニン・ボード
- アルテラ USB-Blaster™ ダウンロード・ケーブル
- HSMC-to-AMC アダプタ・カード

38 ページの「性能の概要」でレポートされた性能結果が 509310 Rev D Dual TMS320TCI6488 メザニン・ボードおよび Rev B Arria II GX FPGA 開発ボードを使用して得られたものです。

図 4 に、TI DSP に接続された Arria II GX FPGA 開発ボードを示します。

図 4. アルテラ Arria II GX FPGA 開発ボードおよびスペクトル・デジタル TMS320TCI6488 AMC メザニン・ボード



また、図 4 に、以下のコネクタも示します。

- アルテラ USB-Blaster ダウンロード・ケーブル
- USB ケーブル
- TI 12 V の電源コード
- アルテラ 16 V の電源コード

アルテラ USB-Blaster ダウンロード・ケーブルは Arria II GX FPGA 開発ボード上の FPGA をコンフィギュレーションし、また、nios2 ターミナル・セッションと FPGA の間で通信するのに使用されます。nios2 ターミナル・セッションは、このリファレンス・デザインで実行されたすべてのテストのための規格の入力、出力、およびエラー位置 (stdio) です。それは RapidIO ドライバへのユーザー・インターフェースです。

Code Composer Studio IDE は USB ケーブルを通して TI DSP カードと通信します。TI 12 V の電源コードおよび 16 V の電源コードはそれぞれ TI DSP カードおよび Arria II GX FPGA 開発ボードに電源を提供します。

ソフトウェア要件

リファレンス・デザインには次のソフトウェアが必要です。

- USB-Blaster ドライバ、SOPC Builder、および RapidIO MegaCore ファンクションを含むアルテラ Quartus® II v9.0 Service Pack 2 (SP2) ソフトウェア
- アルテラ Nios II エンベデッド・デザイン・スイート (EDS) v9.0 SP2
- TI Code Composer Studio IDE v3.3 (CCS)

リファレンス・デザインのインストール

この項では、リファレンス・デザインのインストール方法を説明します。

パッケージのダウンロード

リファレンス・デザインに必要なすべてのファイルが

altera_6488_srio_a2gx_ref_design.zip ファイルに含まれます。このファイルは「[Serial RapidIO To TI 6488 DSP Reference Design web page](#)」からダウンロードできます。

アルテラ FPGA パッケージ・ファイルの抽出

このプロジェクトのために指定した作業ディレクトリで

altera_6488_srio_a2gx_ref_design.zip ファイルを解凍します。ファイルを解凍した後、作業ディレクトリには **altera_srio_ref_design** および **altera_ti_srio_ref_design** という名前の 2 つのサブディレクトリが含まれています。**altera_srio_ref_design** ディレクトリにはアルテラの FPGA をプログラムするためのファイルが含まれています。図 5 に、**altera_srio_ref_design** のディレクトリ構造を示します。

図 5. altera_srio_ref_design ディレクトリ構造



パッケージ内容の説明

altera_srio_ref_design ディレクトリには、各 RapidIO×1 バリエーションに対して 1 行ずつ、3 つのサブディレクトリ、**srio_<rate>_x1** が存在します。

srio_1250_x1 ディレクトリには、1.250 Gbaud 1x RapidIO MegaCore ファンクションのための FPGA デザインが含まれます。このデザインは DMA を使用して、ローカル・メモリ・ブロックから RapidIO MegaCore I/O ライト Avalon-MM スレーブ・ポートにデータを転送します。これらのデータ転送は NWRITE、NWRITE_R、および SWRITE トランザクションの基底です。

srio_2500_x1 および **srio_3125_x1** ディレクトリは、残りの RapidIO MegaCore ファンクション・バリエーション (2.500 および 3.125 Gbaud で 1x バリエーション) に対して FPGA デザインを含みます。これらのデザインは I/O バースト転送を生成するために、Avalon-MM マスター・ポート付きパケット・ジェネレータを使用して、RapidIO MegaCore I/O ライト Avalon-MM スレーブ・ポートに送ります。これらのバーストは NWRITE、NWRITE_R、および SWRITE トランザクションの基底です。デザインは、DMA アプローチを使用することで、これらのデザイン・バリエーションにおけるライン・レート・トラフィックでリンクを飽和することができません。パケット・ジェネレータ・アプローチで、デザインは RapidIO プロトコルのモードとレートのためのリンクを飽和状態にすることができます。

各 **srio_<rate>_<mode>** ディレクトリには、関連デザインでアルテラ FPGA をプログラムするための SRAM Object File (**.sof**) が含まれます。さらに、各のディレクトリには、以下のファイルを含む完全なデザインを再生成するのに必要なすべてのファイルが含まれています。

- **srio_<rate>_<mode>.qpf**
- **srio_<rate>_<mode>.qsf**
- **srio_<rate>_<mode>_sys.ptf**
- **srio_<rate>_<mode>_sys.sopc**
- **srio_<rate>_<mode>_top.v**

srio_<rate>_<mode>_top.v ファイルは SOPC デザインのトップレベル・ラッパーです。それはクロッキング手法を実装します。クロッキング手法について詳しくは、7 ページの「クロック」を参照してください。

以下のセットから、各 **srio_<rate>_<mode>** ディレクトリには DMA ベースまたはパケット・ジェネレータ・ベース・デザインに対して、関連するサブディレクトリが含まれています。

表 2. **srio_<rate>_<mode>** ディレクトリに含まれたサブディレクトリ (その 1)

ディレクトリ名	DMA またはパケット・ベース	説明
.sopc_builder	両方	SOPC Builder により使用した Peripheral Template (.ptf) が含まれます。これらのファイルは SOPC Builder で使用可能なコンポーネントをリストします。
dma	両方	このリファレンス・デザインに使用される DMA コントローラのための Verilog HDL コードが含まれます。また、コンポーネントの機能とパラメータについて説明する SOPC Builder の TCL ファイルが含まれています。
rx_buffer	両方	Avalon-MM トラフィック・シンクの Verilog HDL コード、および SOPC Builder の TCL ファイルが含まれています。

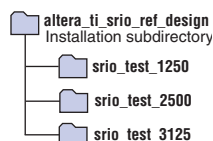
表 2. srio_<rate>_<mode> ディレクトリに含まれたサブディレクトリ (その 2)

ディレクトリ名	DMA またはパケット・ベース	説明
pkt_source	パケット・ベース	Avalon-MM トラフィック・ジェネレータの Verilog HDL コード、および SOPC Builder の TCL ファイルが含まれています。
pkt_checker	パケット・ベース	SOPC Builder の TCL ファイルと同様に、パケット・ディスクリプタ FIFO、NREAD トランザクション・ジェネレータ、およびデータ比較ブロックを含むデータ完全性のチェッカーの Verilog HDL が含まれています。
reset	両方	RESET CONTROL ブロックの Verilog HDL コード、かつ SOPC Builder の TCL ファイルが含まれています。
software_app	両方	以下の 3 つファイルが含まれています。 <ul style="list-style-type: none"> ■ create-this-app – ドライバをコンパイルするのに使用したスクリプト ■ srio_main_full.c – RapidIO ドライバを実装する C コード ■ srio_regs.h – RapidIO MegaCore 内部レジスタのニーモニック
software_bsp	両方	ファイル create-this-bsp が含まれています。このスクリプトはアプリケーションで使用される HAL ドライバを作成し、この場合は RapidIO ドライバです。
srio_stats	両方	STATS コンポーネントの Verilog HDL コード、かつ SOPC builder の TCL ファイルが含まれています。

TI 6488 DSP パッケージ・ファイルの抽出

10 ページの「パッケージのダウンロード」に記載されているとおり、リファレンス・デザイン zip ファイルをダウンロードした後、作業ディレクトリには **altera_ti_srio_ref_design** サブディレクトリが含まれています。このディレクトリには TI 6488 DSP のプログラミングのファイルが含まれています。図 6 に、**altera_ti_srio_ref_design** ディレクトリ構造を示します。

図 6. altera_ti_srio_ref_design ディレクトリ構造



パッケージ内容の説明

altera_ti_srio_ref_design ディレクトリは、Serial RapidIO データ・レートに対して 1 行ずつ、3 つのサブディレクトリが含まれます。

各ディレクトリには以下のファイルが含まれています。

- **lnk.cmd**
- **main_unidirectional.c**
- **main_bidirectional_perf.c**
- **main_bidirectional_dit.c**
- **main_tx_dbell.c**

■ srio_test_<rate>.pjt

表 3 では、これらのファイルについて説明します。

表 3. altera_ti_srio_ref_design サブディレクトリにおけるファイル

ファイル	説明
lnk.cmd	TI 6488 DSP のリンカ・コマンド・ファイルです。
main_unidirectional.c main_bidirectional_perf.c main_bidirectional_dit.c main_tx_dbell.c	TI 6488 DSP をプログラムするためのコードです。
srio_test_<rate>.pjt	Code Composer Studio プロジェクト・ファイルです。それは .c ファイルをコンパイルするためにすべての関連情報が含まれています。

srio_test_<rate>.pjt ファイルにはターゲットとしたライブラリへのパスが含まれています。ライブラリの位置を反映するために、例 1 に示すとおり、このファイルを編集して、パスを変更する必要があります。

例 1. srio_test_<rate>.pjt ファイルのパス

```
ProjectDir="C:\srio_ref_design\srio_a_6488_a2gx\srio_test_<rate>\\"
Source="..\..\..\CCStudio_v3.3\C6000\cgtools\lib\rts64plus.lib"
Source="C:\CCStudio_v3.3\C6000\csl_c6488\csl_c6488.lib"
Options=-g -pdv -fr"${Proj_dir}\Debug" \
-i"c:\srio_ref_design\PCTe_a_6488\ti6488_srio_source\default_package\csl_c6488\inc" \
-d"RATE_<rate>" -d"_DEBUG" -d"CHIP_64XX" -mv6400+
```

アプリケーションを実行するための準備

アプリケーションを実行するには、必要なソフトウェアをインストールして、ハードウェアに接続して、アルテラ FPGA をプログラムして、アプリケーションを選択して、TI 6488 DSP をプログラムする必要があります。そして、選択した性能、データ整合性、および Doorbell メッセージのテストを実行する必要もあります。以下の項では、ハードウェアの接続、アルテラ FPGA のプログラム、アプリケーションの選択、および TI 6488 DSP のプログラムについて説明します。

ハードウェアに接続

ハードウェアに接続する前に、10 ページの「ソフトウェア要件」でリストされる必要なソフトウェアをインストールする必要があります。

ハードウェアに接続するときは、次のステップを実行します。

1. CCS v3.3 をインストールします。
2. EVM で提供された *TMS320TCI6488 EVM Quick Start Installation Guide* の手順に従って、TI DSP カードをパワー・アップおよび初期化します。
3. TI DSP カードへの電源を取り外します。
4. TI DSP カードから TI USB ケーブルを取り除きます。
5. HSMC-to-AMC コネクタを使用して、Arria II GX FPGA 開発ボードを TI DSP カードに接続します。
6. TI USB ケーブルを TI DSP カードに再接続します。
7. USB-Blaster ケーブルを Arria II GX FPGA 開発ボードに接続します。
8. Arria II GX FPGA 開発ボードへの電源を供給します。

9. TI DSP カードへの電源を供給します。
10. Arria II GX FPGA 開発ボードをオンにします。

1× 1.250-Gbaud デザイン・バリエーションに対するアルテラ FPGA のプログラミング

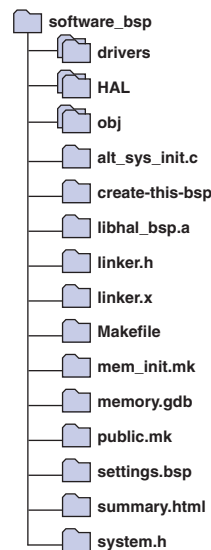
RapidIO の相互運用性リファレンス・デザインの 1× 1.250-Gbaud バリエーションに対するアルテラ FPGA のプログラムするには、以下のステップに従います。

1. Windows Start メニューで、**All Programs > Altera > Nios II EDS <version_number>** をポイントして、Nios II コマンド・シェルを起動するには、**Nios II <version_number> Command Shell** をクリックします。
2. ディレクトリ **altera_srio_ref_design\srio_1250_x1** に移動します。
3. ディレクトリ **software_bsp** に移動します。**create-this-bsp** スクリプト・ファイルはディレクトリで唯一のファイルであるべきです。
4. リファレンス・デザインに必要なすべての HAL ドライバを作成するスクリプトを実行するには、以下のコマンドを入力します。

```
./create-this-bsp ↵
```

図 7 に示すように、**software_bsp** ディレクトリにはサブディレクトリおよびファイルが含まれているはずですが、

図 7. create-this-bsp スクリプトの実行した後の software_bsp ディレクトリ構造



5. ディレクトリ **..\software_app** に移動します。
ディレクトリには次の 3 つファイルが含まれます。
 - **srio_regs.h**
 - **srio_main_full.c**
 - **create-this-app**

6. `srio_main_full.c` におけるドライバ・ソフトウェアをコンパイルするスクリプトを実行するには、以下のコマンドを入力します。

```
./create-this-app ←
```


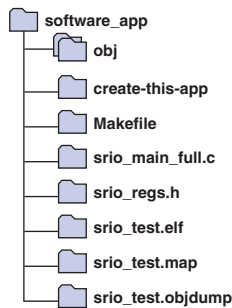
 **srio_regs.h** ファイルはアルテラの RapidIO MegaCore ファンクションにおける Serial RapidIO レジスタのニーモニックが含まれています。

図 8 に示すように、**software_app** ディレクトリにはサブディレクトリおよびファイルが含まれているはずですが。

図 8. create-this-app スクリプトの実行した後の software_app ディレクトリ構造



create-this-app スクリプトが実行される時、コンパイル・エラーがなかったら、FPGA をプログラムして、RapidIO MegaCore ドライバを実行することができます。

FPGA をプログラムするには、ソフトウェア・イメージをダウンロードし、そして、nios2 ターミナル・セッションを起動するには、以下のステップに従います。

1. FPGA をプログラムし、コマンド・シェルで以下のコマンドを入力します。

```
nios2-configure-sof -d 1 ../srio_1250_x1.sof ←
```
2. ソフトウェア・イメージをダウンロードするには、以下のコマンドを入力します。

```
nios2-download --device=1 -g srio_test.elf ←
```

アルテラ RapidIO MegaCore ファンクションが FPGA にダウンロードされます。

3. nios2 ターミナル・セッションを起動するには、以下のコマンドを入力します。

```
nios2-terminal --device=1 ←
```

nios2 ターミナル・セッションが起動され、FPGA 上の Nios II プロセッサと通信することが可能になります。

いずれかの RapidIO の相互運用性リファレンス・デザイン・テストを実行する前に、TI 6488 DSP を表示およびプログラムする必要があります。この手順については、以下の項で説明しています。

残りの RapidIO デザイン・バリエーションに対するアルテラ FPGA のプログラミング

他の RapidIO の相互運用性リファレンス・デザインのバリエーションに対して、アルテラ FPGA をプログラムするには、以下の例外を除き、14 ページの「1x1.250-Gbaud デザイン・バリエーションに対するアルテラ FPGA のプログラミング」の手順を実行します。

- パス `altera_srio_ref_design\srio_1250_x1` をパス `altera_srio_ref_design\srio_<rate>_x1` に置き換えます。
- `srio_1250_x1.sof` の `.sof` ファイル名を `srio_<rate>_x1.sof` に置き換えます。

TI 6488 DSP 上で実行するプログラムの選択

このリファレンス・デザインでは、TI 6488 DSP 上でロードできる 4 つのプログラムを提供します。それらは以下の 4 個のファイルで見つかります。

- `main_unidirectional.c`
- `main_bidirectional_perf.c`
- `main_bidirectional_dit.c`
- `main_tx_dbell.c`

`main_unidirectional.c` でのプログラムはスレーブのみとして TI 6488 DSP をコンフィギュレーションします。このプログラムは FPGA から TI 6488 DSP へのデータ流の性能を測定して、FPGA から TI 6488 DSP への受け渡した Doorbell メッセージをテストするのに使用されます。

`main_bidirectional_perf.c` でのプログラムはスレーブおよびマスターの両方になるように TI 6488 DSP をコンフィギュレーションします。このコンフィギュレーションでは、TI DSP は受信データおよびライト・トランザクションをプロセスして、FPGA へのライト・トランザクションを開始します。

`main_bidirectional_dit.c` でのプログラムはスレーブおよびマスターの両方になるように TI 6488 DSP をコンフィギュレーションします。このコンフィギュレーションでは、TI DSP は受信データおよびライト・トランザクションをプロセスします。また、FPGA へのライト・トランザクションを開始して、ライト位置からデータ整合性のチェックのためのデータを読み出します。



`main_bidirectional_dit.c` にあるプログラムはリンクを飽和にすることはできません。したがって、性能を測定するのに使用してはなりません。

`main_tx_dbell.c` にあるプログラムはアルテラ FPGA の上の RapidIO コアに Doorbell メッセージを送るために TI 6488 DSP をコンフィギュレーションします。

TI 6488 DSP ドライバをロードおよび実行する前に、`srio_test_<rate>.pjt` ファイルを編集して、選択するプログラムを指定できます。リファレンス・デザイン `.zip` ファイルに提供される `srio_test_<rate>.pjt` ファイルは、すべての 4 つのプログラムをソースする行が含まれています。テキスト・エディタでファイルを開いて、希望するプログラムに対応する行を非コメントします。他に対してコメントをします。

例 2 に、`main_unidirectional.c` ファイルにあるプログラムを指定する `srio_test_<rate>.pjt` ファイルの関連するソース行を示します。

例 2. `main_unidirectional.c` のコンパイルを指定する、`srio_test_<rate>.pjt` ファイルにあるセクション

```
Source="main_unidirectional.c"
#Source="main_bidirectional_dit.c"
#Source="main_bidirectional_perf.c"
#Source="main_tx_dbell.c"
```

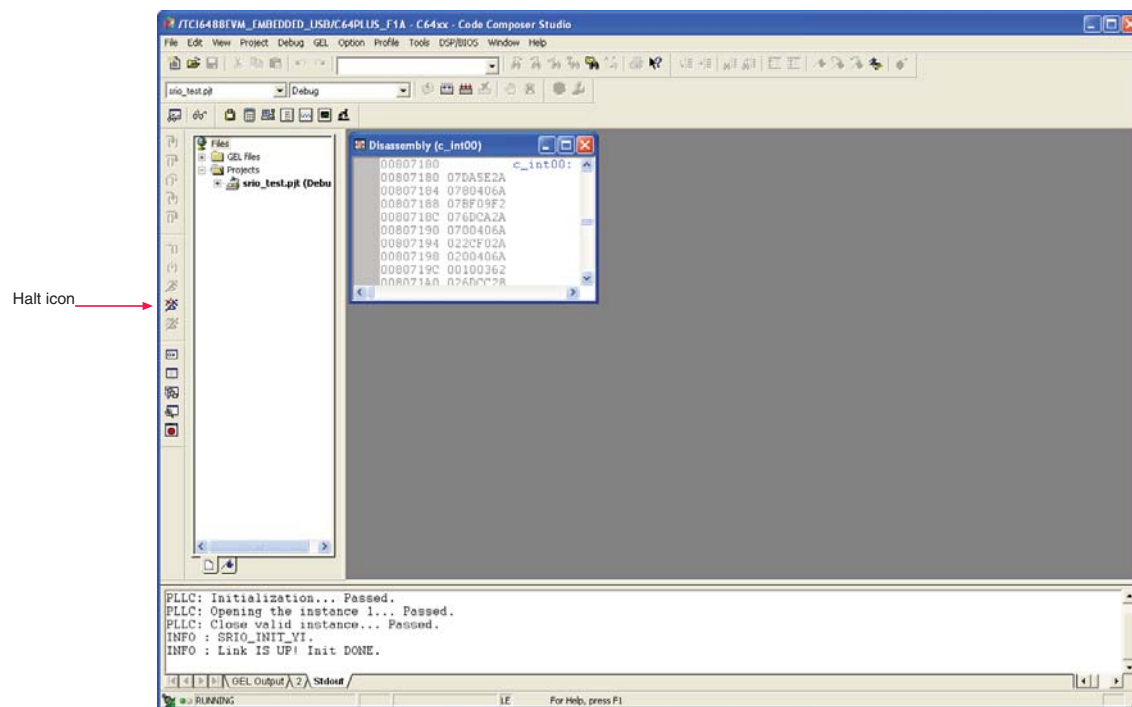
CCStudio IDE v3.3 の実行

.pjt ファイルを編集して、選択するプログラムを指定したら、TI 6488 DSP ドライバをロードおよび実行することができます。この項では、いずれかの RapidIO バリエーションに対して TI 6488 DSP ドライバをロードおよび実行するには、CCStudio v3.3 (CCS) セッションを開始およびリセットする方法を説明します。

新しいプロジェクトを実行するために CCS IDE セッションを準備するには、以下のステップに従います。

1. TI 6488 DSP プログラムは CCS セッションを実行している場合、以下のステップを実行して、プログラムを中止し、以前のセッションを完全に閉じます。
 - a. TMS320TCI6488 EVM の CCS ドライバ・ウィンドウで、左側の **Halt** アイコンをクリックして、プログラムを停止します。図 9 に、そのアイコンを示します。
 - b. Project メニューの **Close** をクリックします。
 - c. Debug メニューの **Disconnect** をクリックします。
 - d. File メニューの **Close Session** をクリックします。
 - e. Parallel Debug Manager の File メニューの **Exit** をクリックします。図 10 に Parallel Debug Manager を示します。

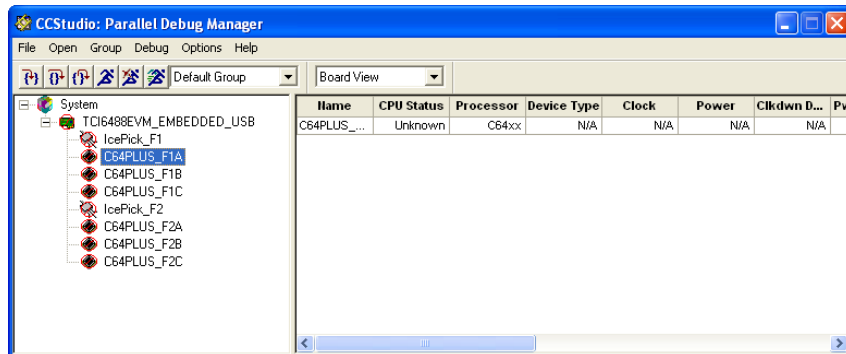
図 9. Halt アイコンがアクティブしている Code Composer Studio ウィンドウ



2. **TCI6488 EVM CCStudio v3.3** アイコンをダブル・クリックします。Parallel Debug Manager ウィンドウが表示されます。
3. **System** の下にある **TCI6488EVM_EMBEDDED_USB** で、**C64PLUS_F1A** をダブル・クリックして、EVM ボード上の F1 TI 6488 DSP のコア A の CCS セッションを起動します。図 10 に、CPU を選択した後の Parallel Debug Manager を示します。

4. Debug メニューの **Connect** をクリックします。ドライバと F1 TI 6488 DSP コア A 間の接続が確立されます。

図 10. CPU の選択後の Parallel Debug Manager



1× 1.250-Gbaud デザイン・バリエーションに対する TI 6488 DSP のプログラミング

.pjt ファイルを編集して、望ましいプログラムを指定し、CCS セッションをリセットしたら、TI 6488 DSP ドライバをロードおよび実行することができます。この項では、1× 1.250-Gbaud デザイン・バリエーションに対して TI 6488 DSP ドライバをロードおよび実行する方法を説明します。

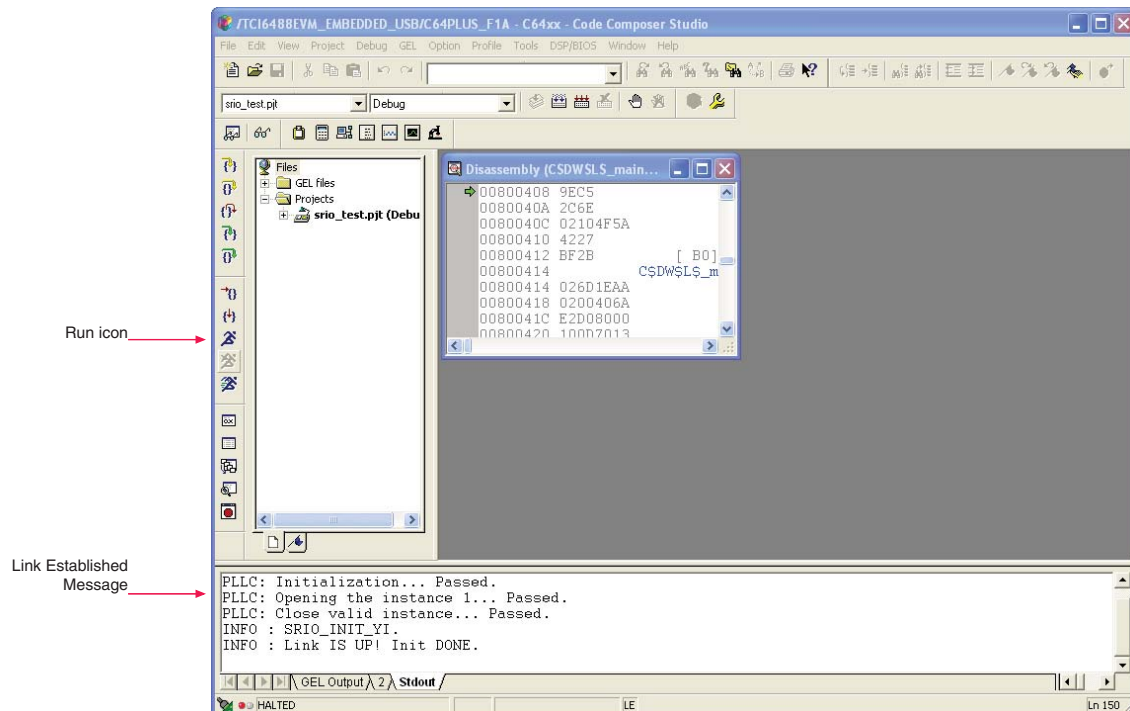
このデザインをロードして実行するには、以下のステップを実行します。

1. Project メニューの **Open** をクリックします。
2. **Project Open** ダイアログ・ボックスで、**srio_test_1250** ディレクトリに移動します。
3. ファイル **srio_test_1250.pjt** を選択します。
4. **Open** をクリックします。
5. File View ウィンドウの **Projects** ディレクトリで、**srio_test_1250.pjt** ファイルを選択します。
6. ファイル名を右クリックし、**Clean** をクリックします。
7. ファイル名がハイライトされたままで、**Build** をクリックします。TI 6488 DSP をコンフィギュレーションするコードがコンパイルされます。
8. File メニューの **Load Program** をクリックします。
9. Load Program ウィンドウで、**Debug** ディレクトリに移動します。
10. **srio_1250_test.out** を選択します。
11. **Open** をクリックします。
12. CCS ドライバ・ウィンドウで、左側の **Run** アイコンをクリックして、プログラムを実行します。図 11 に、そのアイコンを示します。

stdout ウィンドウ・パネルで、コードにエンベデッドされたメッセージが表示されます。表示された最終的なメッセージは、リンクが確立されていることを示します。

図 11 に、Run アイコンおよびリンクが確立されているという最終的なメッセージを示します。

図 11. Code Composer Studio ウィンドウ



残りのデザイン・バリエーションに対する TI 6488 DSP のプログラミング

その他の RapidIO の相互運用性リファレンス・デザインのバリエーションに対して、TI 6488 DSP をプログラムするには、以下の例外を除き、18 ページの「1×1.250-Gbaud デザイン・バリエーションに対する TI 6488 DSP のプログラミング」の手順を実行します。

- ディレクトリ **srio_test_1250** をディレクトリ **srio_test_<rate>** に置き換えます。
- **srio_test_1250.pjt** の .pjt ファイル名を **srio_test_<rate>.pjt** に置き換えます。
- **srio_test_1250.out** の .out ファイル名を **srio_test_<rate>.out** に置き換えます。

アプリケーションの実行

アルテラ FPGA をコンフィギュレーションして、TI 6488 DSP に選択したアプリケーションをロードすると、4つの種類から1つのテストであるアプリケーションを実行できます。

DMA ベースのシステムで実装された RapidIO MegaCore バリエーション (1×1.250-Gbaud バリエーション) とパケット・ジェネレータ・ベースのシステムで実装された残りのバリエーションの性能及びデータ・インテグリティ・テストの実行するステップは異なります。

以下の項では、4つのアプリケーション・プログラムの実行方法について説明しています。

単一方向の性能テストの実行

以下の項では、単一方向の性能テストの実行方法について説明しています。

1×1.250-Gbaud デザイン・バリエーションに対する単方向の性能テストの実行

1×1.250-Gbaud デザイン・バリエーションに対する単方向の性能テストを実行する前に、まだ行っていない場合は、アルテラ FPGA をコンフィギュレーションして、単方向の性能テストをロードする必要があります。1×1.250-Gbaud バリエーションの単方向の性能テスト付き TI 6488 DSP をロードし、そしてアルテラ FPGA をコンフィギュレーションするには、以下のステップに従います。

1. このデザイン・バリエーションのために FPGA を再プログラムした場合、**14 ページ**の「**1×1.250-Gbaud デザイン・バリエーションに対するアルテラ FPGA のプログラミング**」の説明に従います。



このデザイン・バリエーションのために FPGA を再プログラムしましたが、このバリエーションに対して `create-this-bsp` および `create-this-app` をすでに実行した場合、**15 ページ**のステップ**1**およびステップ**2**のみを実行する必要があります。

2. コンパイルされる `main_unidirectional.c` ファイルを指定するには、`srio_test_1250.pjt` ファイルを編集します。
3. **17 ページ**の「**CCStudio IDE v3.3 の実行**」の説明に従ってください。
4. **18 ページ**の「**1×1.250-Gbaud デザイン・バリエーションに対する TI 6488 DSP のプログラミング**」の説明に従ってください。

単一方向の性能テストを実行するには、ステップ**1**で開いた `nios2` ターミナル・セッションに戻ります。サポートされる RapidIO ドライバ・コマンドのリストについては、`h` を入力します。

表 4 に説明するとおり、次のコマンドのシーケンスは RapidIO MegaCore ファンクションをプログラムし、TI 6488 DSP にトラフィックを生成し、そして性能をモニタします。デフォルト `init` の設定は `NWRITE` トランザクションを生成します。

```
stop
r
rate 1250
clk 40
pload 256
init
start
stats
z
stop
```

表 4 に、シーケンスにおける個別のコマンドについて説明します。

表 4. 1×1.250-Gbaud RapidIO バリエーションで NWRITE の性能テストの nios2 ターミナル・コマンド

コマンド	説明
stop	WR DMA ブロックから RapidIO I/O ライト Avalon-MM スレーブ・ポートへの DMA 転送を停止します。以前のテスト・アクティビティが停止されることを確認するために、コマンド・シーケンスをこのコマンドで起動し、そして現在のテスト・アクティビティを停止するために、コマンド・シーケンスをこのコマンドで終了します。
r	RapidIO MegaCore ファンクションをリセットします。
rate 1250	レートを 1.250 Gbaud に設定します。
clk 40	Avalon-MM システム・クロックが 40 MHz に実行することをドライバに連絡ます。RapidIO MegaCore ファンクションのバリエーションとクロック・レートのリストおよび対応する Avalon-MM システム・クロック周波数について、7 ページの表 1 を参照してください。プログラムする clk 値はスループットを計算するのに使用されます。
pload 256	256 バイト・ペイロードをバーストするには、ライト DMA をプログラムします。許容値は最小値 8 と最大値 256 がある 8 バイトの倍数です。
init	RapidIO MegaCore ファンクションにある CSR のサブセットを初期化します。プログラムされた特定のコンフィギュレーション・レジスタについて、ファイル <code>srio_main_full.c</code> を参照してください。
start	WR DMA ブロックから RapidIO I/O ライト Avalon-MM スレーブ・ポートまで DMA 転送を起動します。
stats	統計収集の動作をイネーブルします。
z	統計収集の動作を停止します。

図 12 に、RapidIO バリエーションのための単一方向の統計収集サイクルの例を示します。

図 12. 1×1.250 Gbaud で単一方向の統計収集サイクルの例

```

Nios II ED5 9.0
Baud Rate : 1250 Mbaud
TX Throughput 0.912746 Gbps TX Efficiency 91.274590%
RX Throughput 0.000000 Gbps RX Efficiency 0.000000%

Sample Window ..... 134217727 cycles
RapidIO Payload ..... 256 bytes

TX Packets ..... 1495442
RX Packets ..... 0

IO S WR Bytes ..... 38283376
IO M RD Bytes ..... 0

IO S RD Bytes ..... 0
IO M WR Bytes ..... 0

Pkts Cancelled ..... 0
Pkts Not Accepted ..... 0
Pkt Retrys ..... 0
Pkt CRC Errors ..... 0
Symbol Errors ..... 0
Char Errors ..... 0
Pkts Dropped by Transport Layer ..... 0

Serial RapidIO>

```

コマンド `start` の代わりにコマンド・シーケンスでコマンド `dit` を入力する場合、データ整合性のチェックを可能にします。このテストで、FPGA は通常単一方向の性能テストのようにライト・コマンドを開始します。しかしながら、`dit` テストで、FPGA はまた、ライト動作の成功を確認するのにリード・コマンドが開始されます。

dit テスト中、nios2 ターミナルは送信およびチェックされるパケットの実行するカウントを表示します。1x 1.25-Gbaud RapidIO MegaCore バリエーションはダイナミック統計収集をサポートしません。パケットが送信およびチェックされたら、Serial RapidIO> プロンプトが再び利用可能です。このテスト中のパケット・アクティビティの概要を表示するには、Serial RapidIO> プロンプトで dit_stats を入力します。

表 5 に、これらのデータ整合性のチェック・コマンドに関する情報を提供します。

表 5. データ整合性のテストに対する nios2 ターミナル・コマンド

コマンド	説明
dit N	データ整合性のテストを実行します。dit コマンドは FPGA が TI DSP に書き込むパケットの数を指定する 10 進数引数をかかります。データ整合性のテストでは、FPGA は各パケットを書き込み、そしてデータが正しく書き込まれるのを確認するために、そのデータを読み出します。
dit_stats	実行が完了した時に、統計収集の概要を表示します。データ整合性のテストを実行している間、RapidIO MegaCore バリエーションが統計をダイナミックに表示できないので、データ整合性のテストは 1x 1.250 Gbaud RapidIO バリエーションで実行した後、統計を表示するのに、このコマンドを使用する必要があります。

図 13 に、コマンド dit 100 がコマンド start に代わってコマンド・シーケンスに挿入された後に、この RapidIO バリエーションのための単一方向の統計収集サイクルの例を示しています。dit_stats ディスプレイには、データ整合性のチェックのために、RapidIO MegaCore ファンクションで受信されるリード・データを示します。

図 13. 1x 1.250 Gbaud でデータ整合性のテスト付き単一方向の統計の例

```

Nios II EDS 9.0
tested packet 61
tested packet 62
tested packet 63
PASS: Data Integrity Test
Serial RapidIO> dit_stats
Command Read: dit_stats
TX Packets ..... 200
RX Packets ..... 100

  IO S WR Bytes ..... 25600
  IO M RD Bytes ..... 0

  IO S RD Bytes ..... 25600
  IO M WR Bytes ..... 0

Pkts Cancelled ..... 0
Pkts Not Accepted ..... 0
Pkt Retrys ..... 0
Pkt CRC Errors ..... 0
Symbol Errors ..... 0
Char Errors ..... 0
Pkts Dropped by Transport Layer ..... 0


Serial RapidIO>

```

1x 2.500-Gbaud デザイン・バリエーションに対する単一方向性能テストの実行

1x 2.500-Gbaud デザイン・バリエーションに対する単一方向の性能テストを実行する前に、まだ行っていない場合は、アルテラ FPGA をコンフィギュレーションして、単一方向の性能テストをロードする必要があります。1x 2.500-Gbaud バリエーションの単一方向の性能テスト付き TI 6488 DSP をロードし、そしてアルテラ FPGA をコンフィギュレーションするには、以下のステップに従います。

1. このデザイン・バリエーションのために FPGA を再プログラムした場合、15 ページの「残りの RapidIO デザイン・バリエーションに対するアルテラ FPGA のプログラミング」の説明に従います。

 このデザイン・バリエーションのために FPGA を再プログラムしましたが、このバリエーションに対して create-this-bsp および create-this-app をすでに実行した場合、15 ページの「残りの RapidIO デザイン・バリエーションに対するアルテラ FPGA のプログラミング」の .sof ファイル名の置換が示されている 15 ページのステップ 1 とステップ 2 のみを実行する必要があります。

2. **main_unidirectional.c** ファイルをコンパイルされるように、**srio_test_2500.pjt** ファイルを編集して、それを指定します。
3. 17 ページの「CCStudio IDE v3.3 の実行」の説明に従ってください。
4. 19 ページの「残りのデザイン・バリエーションに対する TI 6488 DSP のプログラミング」の説明に従ってください。

単一方向の性能テストを実行するには、ステップ 1 で開いた nios2 ターミナル・セッションに戻ります。サポートされる RapidIO ドライバ・コマンドのリストについては、h を入力します。

表 6 に説明するとおり、次のコマンドのシーケンスは RapidIO MegaCore ファンクションをプログラムし、TI 6488 DSP にトラフィックを生成し、そして性能をモニタします。デフォルト init の設定は NWRITE トランザクションを生成します。

```
stop
r
rate 2500
mode 1
gap 4
clk 75
pload 256
init
start
stats
z
stop
```

表 6 に、シーケンスにおける個別のコマンドについて説明します。

表 6. 1×2.500-Gbaud RapidIO バリエーションで NWRITE の性能テストの nios2 ターミナル・コマンド (その 1)

コマンド	説明
stop	パケット・ジェネレータを停止します。以前のテスト・アクティビティが停止されることを確認するために、コマンド・シーケンスをこのコマンドで起動し、そして現在のテスト・アクティビティを停止するために、コマンド・シーケンスをこのコマンドで終了します。
r	RapidIO MegaCore ファンクションをリセットします。
rate 1250	レートを 2.500 Gbaud に設定します。
mode 1	モードを設定します。TI 6488 DSP は x1 RapidIO バリエーションのみをサポートするので、相互運用性リファレンス・デザインはモード 1 のみをサポートします。
gap 4	Avalon-MM システム・クロック・サイクル数を指定するには、Avalon-MM ライト・バーストから RapidIO MegaCore I/O ライト Avalon-MM スレーブ・ポートへのギャップを設定します。より小さいギャップの値で、パケット・ジェネレータはバースト間の短い遅延のある RapidIO MegaCore I/O ライト Avalon-MM スレーブ・ポートにトラフィックを押し、スループットを増加させます。

表 6. 1× 2.500-Gbaud RapidIO バリエーションで NWRITE の性能テストの nios2 ターミナル・コマンド (その 2)

コマンド	説明
clk 75	Avalon-MM システム・クロックが 75 MHz に実行することをドライバに連絡ます。RapidIO MegaCore ファンクションのバリエーションとクロック・レートのリストおよび対応する Avalon-MM システム・クロック周波数について、7 ページの表 1 を参照してください。
pload 256	256 バイト・ペイロードをバーストするには、パケット・ジェネレータをプログラムします。許容値は最小値 8 と最大値 256 がある 8 バイトの倍数です。
init	RapidIO MegaCore ファンクションにある CSR のサブセットを初期化します。プログラムされた特定のコンフィギュレーション・レジスタについて、ファイル <code>srio_main_full.c</code> を参照してください。
start	パケット・ジェネレータが RapidIO I/O ライト Avalon-MM スレーブ・ポートへのトラフィックの転送を開始することができます。
stats	統計収集の動作をイネーブルします。
z	統計収集の動作を停止します。

図 14 に、RapidIO バリエーションのための単一方向の統計収集サイクルの例を示します。

図 14. 1× 2.500 Gbaud での単一方向の統計収集サイクルの例

```

Nios II EDS 9.0
Baud Rate : 2500 Mbaud
TX Throughput 1.825492 Gbps TX Efficiency 91.274590%
RX Throughput 0.000000 Gbps RX Efficiency 0.000000%

Sample Window ..... 13421727 cycles
RapidIO Payload ..... 256 bytes

TX Packets ..... 1595139
RX Packets ..... 0

IO S WR Bytes ..... 408355596
IO M RD Bytes ..... 0

IO S RD Bytes ..... 0
IO M WR Bytes ..... 0

Pkts Cancelled ..... 0
Pkts Not Accepted ..... 0
Pkt Retrans ..... 0
Pkt CRC Errors ..... 0
Symbol Errors ..... 0
Char Errors ..... 0
Pkts Dropped by Transport Layer ..... 0

Serial RapidIO> _

```

コマンド `start` の代わりにコマンド・シーケンスでコマンド `dit` を入力する場合、`pkt_check` コンポーネントによりデータ整合性のチェックを可能にします。このテストで、FPGA は通常単一方向の性能テストのようにライト・コマンドを開始します。しかしながら、`dit` テストで、FPGA はまた、ライト動作の成功を確認するのにリード・コマンドが開始されます。22 ページの表 5 に、これらのデータ整合性のチェック・コマンドに関する情報を提供します。

図 15 に、コマンド `stats` の前にコマンド `dit 80000000` がコマンド・シーケンスに入力された後、この RapidIO バリエーションのための単一方向の統計収集サイクルの例を示しています。ディスプレイにはデータ整合性のチェックの用に、RapidIO MegaCore ファンクションで受信されるリード・データを示します。

図 15. 1× 2.500 Gbaud でデータ整合性のテスト付き単一方向の統計の例

```

Nios II EDS 9.0
Baud Rate : 2500 Mbaud
TX Throughput 1.703810 Gbps TX Efficiency 85.190506%
RX Throughput 1.703810 Gbps RX Efficiency 85.190506%
Sample Window ..... 134217227 cycles
RapidIO Payload ..... 256 bytes
TX Packets ..... 2977624
RX Packets ..... 1488812
IO S WR Bytes ..... 381135872
IO M RD Bytes ..... 0
IO S RD Bytes ..... 381135872
IO M WR Bytes ..... 0
Pkts Cancelled ..... 0
Pkts Not Accepted ..... 0
Pkt Retrys ..... 0
Pkt CRC Errors ..... 0
Symbol Errors ..... 0
Char Errors ..... 0
Pkts Dropped by Transport Layer ... 0
Serial RapidIO>

```

1× 3.125-Gbaud デザイン・バリエーションに対する単一方向性能テストの実行

1× 3.125-Gbaud デザイン・バリエーションに対する単一方向の性能テストを実行する前に、正しいバリエーションのための FPGA をプログラムし、そして単一方向の性能テストおよび正しいレート設定に対する TI 6488 DSP をプログラムする必要があります。

適切な変更で「1× 2.500-Gbaud デザイン・バリエーションに対する単一方向性能テストの実行」の手順に従って、1× 3.125-Gbaud バリエーションをプログラムして実行します。プログラムを実行する時、正しい rate (3125)、mode (1)、および clk (84) を指定し、また、gap を指定します。この rate の正しい clk 設定に対して、7 ページのを参照してください。gap 設定はクロック・サイクルで測定されて、すべてのパケット・ジェネレータ・ベース・バリエーションに対して同じままであることができます。


双方向性能テストの実行

以下の項では、双方向性能テストの実行方法について説明しています。

1× 1.250-Gbaud デザイン・バリエーションに対する双方向性能テストの実行

アプリケーションを実行する前に、アルテラ FPGA をコンフィギュレーションして、双方向性能テストをロードする必要があります。1× 1.250-Gbaud バリエーションの双方向の性能テスト付き TI 6488 DSP をロードし、そしてアルテラ FPGA をコンフィギュレーションするには、以下のステップに従います。

1. このデザイン・バリエーションのために FPGA を再プログラムした場合、14 ページの「1× 1.250-Gbaud デザイン・バリエーションに対するアルテラ FPGA のプログラミング」の説明に従います。

 このデザイン・バリエーションのために FPGA を再プログラムしましたが、このバリエーションに対して create-this-bsp および create-this-app をすでに実行した場合、その時 15 ページのステップ 1 およびステップ 2 のみを実行する必要があります。

2. コンパイルされる main_bidirectional_perf.c ファイルを指定するには、srio_test_1250.pjt ファイルを編集します。

3. 17 ページの「CCStudio IDE v3.3 の実行」の説明に従ってください。
4. 18 ページの「1×1.250-Gbaud デザイン・バリエーションに対する TI 6488 DSP のプログラミング」のステップ 1～11 を実行します。

FPGA および TI 6488 DSP 上で双方向性能のテストを実行するには、以下のステップに従います。

1. ステップ 1 で開いた nios2 ターミナル・セッションに戻ります。
2. Serial RapidIO> プロンプトで、次のコマンドのシーケンスを入力します。


```
stop
r
rate 1250
clk 40
pload 256
init
start
stats
```
3. 18 ページの「1×1.250-Gbaud デザイン・バリエーションに対する TI 6488 DSP のプログラミング」のステップ 12 を実行します。(CCS ドライバ・ウインドウで、左側の Run アイコンをクリックして、プログラムを実行します)。
4. nios2 ターミナルで z を入力して、統計収集を停止します。
5. stats を入力して、統計収集を再起動し、表示されます。
6. TI DSP からの要求に反応する FPGA ではなく、FPGA から開始されるトラフィックを停止するには、stop を入力します。

最初に、nios2 ターミナルが FPGA から TI 6488 DSP へのトラフィックのみが表示されます。ステップ 3 を実行した後に、トラフィックが両方向に流れ、そして統計は FPGA および TI 6488 DSP から開始されるトランザクションのゼロ以外の数を示します。

図 16 に、RapidIO バリエーションの双方向の性能統計収集サイクルの例を示します。

図 16. 1×1.250 Gbaud で双方向の性能統計収集サイクルの例

```

Nios II EDS 9.0
Baud Rate : 1250 Mbaud
TX Throughput 0.903022 Gbps TX Efficiency 90.302238%
RX Throughput 0.911937 Gbps RX Efficiency 91.193687%

Sample Window ..... 134217727 cycles
RapidIO Payload ..... 256 bytes

TX Packets ..... 1479512
RX Packets ..... 1494117

IO S WR Bytes ..... 378755060
IO M RD Bytes ..... 0

IO S RD Bytes ..... 0
IO M WR Bytes ..... 382494064

Pkts Cancelled ..... 0
Pkts Not Accepted ..... 0
Pkt Retrys ..... 0
Pkt CRC Errors ..... 0
Symbol Errors ..... 0
Char Errors ..... 0
Pkts Dropped by Transport Layer ... 0

Serial RapidIO>

```

stop を入力すると、次のコマンド・シーケンスを入力することによって、異なるバースト・サイズ、<payload value>を試みることができます。

```
pload <payload value>
start
```

<payload value> の許容値は 21 ページの表 4 に記載されています。

1× 2.500-Gbaud デザイン・バリエーションに対する双方向性能テストの実行

アプリケーションを実行する前に、アルテラ FPGA をコンフィギュレーションして、双方向性能テストをロードする必要があります。1× 1.250-Gbaud バリエーションの双方向の性能テスト付き TI 6488 DSP をロードし、そしてアルテラ FPGA をコンフィギュレーションするには、以下のステップに従います。

1. このデザイン・バリエーションのために FPGA を再プログラムした場合、15 ページの「残りの RapidIO デザイン・バリエーションに対するアルテラ FPGA のプログラミング」の説明に従います。



このデザイン・バリエーションのために FPGA を再プログラムしましたが、このバリエーションに対して create-this-bsp および create-this-app をすでに実行した場合、その時 15 ページのステップ 1 およびステップ 2 のみを実行する必要があります。

2. コンパイルされる main_bidirectional_perf.c ファイルを指定するには、srio_test_2500.pjt ファイルを編集します。
3. 17 ページの「CCStudio IDE v3.3 の実行」の説明に従ってください。
4. 19 ページの「残りのデザイン・バリエーションに対する TI 6488 DSP のプログラミング」のステップ 1～11 を実行します。

FPGA および TI 6488 DSP 上で双方向性能のテストを実行するには、以下のステップに従います。

1. ステップ 1 で開いた nios2 ターミナル・セッションに戻ります。
2. Serial RapidIO> プロンプトで、次のコマンドのシーケンスを入力します。

```
stop
r
rate 2500
mode 1
gap 4
clk 75
pload 256
init
start
stats
```

3. 18 ページの「1× 1.250-Gbaud デザイン・バリエーションに対する TI 6488 DSP のプログラミング」のステップ 12 を実行します。(CCS ドライバ・ウインドウで、左側の Run アイコンをクリックして、プログラムを実行します)。
4. nios2 ターミナルで z を入力して、統計収集を停止します。
5. stats を入力して、統計収集を再起動し、表示されます。
6. TI DSP からの要求に反応する FPGA ではなく、FPGA から開始されるトラフィックを停止するには、stop を入力します。

最初に、nios2 ターミナルが FPGA から TI 6488 DSP へのトラフィックのみが表示されます。ステップ 3 を実行した後に、トラフィックが両方向に流れ、そして統計は FPGA および TI 6488 DSP から開始されるトランザクションのゼロ以外の数を示します。

図 17 に、RapidIO バリエーションの双方向の性能統計収集サイクルの例を示します。

図 17. 1× 2.500 Gbaud で双方向の性能統計収集サイクルの例

```

Nios II EDS 9.0
Baud Rate : 2500 Mbaud
TX Throughput 1.805784 Gbps TX Efficiency 90.289200%
RX Throughput 1.823876 Gbps RX Efficiency 91.193810%

Sample Window ..... 134217227 cycles
RapidIO Payload ..... 256 bytes

TX Packets ..... 1577918
RX Packets ..... 1593728

IO S WR Bytes ..... 403947040
IO M RD Bytes ..... 0

IO S RD Bytes ..... 0
IO M WR bytes ..... 407994212

Pkts Cancelled ..... 0
Pkts Not Accepted ..... 0
Pkt Retxys ..... 0
Pkt CRC Errors ..... 0
Symbol Errors ..... 0
Char Errors ..... 0
Pkts Dropped by Transport Layer .... 0

Serial RapidIO>

```

1× 3.125-Gbaud デザイン・バリエーションに対する双方向性能テストの実行

1× 3.125-Gbaud デザイン・バリエーションに対する双方向の性能テストを実行する前に、正しいバリエーションのための FPGA をプログラムし、そして双方向の性能テストおよび正しいレート設定に対する TI 6488 DSP をプログラムする必要があります。

適切な変更で、27 ページの「1× 2.500-Gbaud デザイン・バリエーションに対する双方向性能テストの実行」の手順に従って、1× 3.125-Gbaud バリエーションをプログラムして実行します。プログラムを実行する時、正しい rate (3125)、mode (1)、および clk (84) を指定し、また、gap を指定します。このレートの正しい clk 設定に対して、7 ページのを参照してください。gap 設定はクロック・サイクルにおける測定されて、パケット・ジェネレータ・ベース・バリエーションのすべてに同じままであることができます。


双方向データの整合性テストの実行

以下の項では、双方向データの整合性テストの実行方法について説明しています。

1× 1.250-Gbaud デザイン・バリエーションに対する双方向データの整合性テストの実行

アプリケーションを実行する前に、アルテラ FPGA をコンフィギュレーションして、双方向データの整合性テストをロードする必要があります。1× 1.250-Gbaud バリエーションの双方向データの整合性テスト付き TI 6488 DSP をロードし、そしてアルテラ FPGA をコンフィギュレーションするには、以下のステップに従います。

1. このデザイン・バリエーションのために FPGA を再プログラムした場合、14 ページの「1× 1.250-Gbaud デザイン・バリエーションに対するアルテラ FPGA のプログラミング」の説明に従います。

 このデザイン・バリエーションのために FPGA を再プログラムしましたが、このバリエーションに対して create-this-bsp および create-this-app をすでに実行した場合、15 ページのステップ 1 および ステップ 2 のみを実行する必要があります。

2. main_bidirectional_dit.c ファイルがコンパイルされることを指定するには、srio_test_1250.pjt ファイルを編集します。

3. 17 ページの「CCStudio IDE v3.3 の実行」の説明に従ってください。
4. 18 ページの「1×1.250-Gbaud デザイン・バリエーションに対する TI 6488 DSP のプログラミング」のステップ 1～11 を実行します。

FPGA および TI 6488 DSP 上で双方向データの整合性テストを実行するには、以下のステップに従います。

1. ステップ 1 で開いた nios2 ターミナル・セッションに戻ります。
2. Serial RapidIO> プロンプトで、次のコマンドのシーケンスを入力します。

```
stop
r
rate 1250
clk 40
pload 256
init
```

3. 18 ページの「1×1.250-Gbaud デザイン・バリエーションに対する TI 6488 DSP のプログラミング」のステップ 12 を実行します。(CCS ドライバ・ウインドウで、左側の **Run** アイコンをクリックして、プログラムを実行します)。
4. Serial RapidIO> プロンプトで、次のコマンドを入力します。

```
dit 10000000
```

nios2 ターミナルは送信およびチェックされるパケットの実行するカウントを表示します。1×1.25-Gbaud RapidIO MegaCore バリエーションはダイナミック統計収集をサポートしません。10000000 つのパケットが送信およびチェックされると、Serial RapidIO> プロンプトが再び利用可能です。このテスト中のパケット・アクティビティの概要を表示するには、Serial RapidIO> プロンプトで dit_stats を入力します。

dit コマンドの値としてパケットの小数を使用することができます。10000000 の値で SignalTap® II エンベデッド・ロジック・アレイ・アナライザを使用してパケット・アクティビティを表示することができます。詳細は、「SignalTap II エンベデッド・ロジック・アナライザを使用した RapidIO-Avalon インタフェースの表示」を参照してください。

図 18 に、以下の手順を実行した後で nios2 ターミナル・ウインドウの例を示します。

1. nios2 ターミナルで、以下のコマンド・シーケンスを実行します。

```
stop
r
rate 1250
clk 40
pload 256
init
```

2. CCS ドライバ・ウインドウで、左側の **Run** アイコンをクリックして、プログラムを実行します。
3. nios2 ターミナルで、以下のコマンド・シーケンスを実行します。

```
dit 20000
dit_stats
```

図 18. 1× 1.250 Gbaud で表示される双方向データの整合性テストの例および dit_stats

```

Nios II EDS 9.0
tested packet 4e1b
tested packet 4e1c
tested packet 4e1d
tested packet 4e1e
tested packet 4e1f
PASS: Data Integrity Test
Serial RapidIO> dit_stats
Command Read: dit_stats
TX Packets ..... 41843
RX Packets ..... 23686

IO S WR Bytes ..... 5120000
IO M RD Bytes ..... 471808


IO S RD Bytes ..... 5120000
IO M WR Bytes ..... 471808

Pkts Cancelled ..... 0
Pkts Not Accepted ..... 0
Pkt Retrgs ..... 0
Pkt CRC Errors ..... 0
Symbol Errors ..... 0
Char Errors ..... 0
Pkts Dropped by Transport Layer ... 0

Serial RapidIO>

```

図 18 で、dit コマンドを実装するには、bidirectional_dit テストはデータの整合性のための 0x4E20 (10 進数で 20000) パケットをテストします。パケット付番は 0x0 から 0x4E1F までです。dit_stats コマンドは実行中に収集された統計を出力します。FPGA は 5120000 バイトのデータを書き込んで、そして読み出します。それは、トータル・ペイロードに 20000 トランザクションであり、各転送は 256 バイトであります。これらのデータ転送を実装するには、FPGA が 20000 ライト・パケットおよび 20000 リード要求パケットを送信し、かつ 20000 リード応答パケットを受け取ります。また、TI 6488 DSP はデータの整合性チェックを実行するの時、データを書き込んで、読み出します。統計レポート時に、TI DSP は 471808 バイトの合計ペイロードである 1843 ライト・パケットおよび 1843 リード応答パケットを送信して、書き込んだデータの整合性をチェックします。それに対応して、FPGA は 1843 リード応答パケットにあるデータの 471808 バイトを送信します。

 他の 2 つの RapidIO MegaCore バリエーションとは対照的に、1× 1.250 Gbaud RapidIO MegaCore バリエーションはデータの整合性テストを実行している間に統計をダイナミックに表示されません。この RapidIO バリエーションの双方向のデータ完全性テスト・ランから統計を表示するために、テスト・ランが完了してから dit_stats を実行することが必要です。

1× 2.500-Gbaud デザイン・バリエーションに対する双方向データの整合性テストの実行

1× 2.500-Gbaud デザイン・バリエーションに対する双方向の性能テストを実行する前に、正しいバリエーションのための FPGA をプログラムし、そして双方向の性能テストおよび正しいレート設定に対する TI 6488 DSP をプログラムする必要があります。

Serial RapidIO> プロンプトでプログラムを実行する時、適切な変更で 28 ページの「1× 1.250-Gbaud デザイン・バリエーションに対する双方向データの整合性テストの実行」の手順に従って、1× 2.500-Gbaud バリエーションをプログラムして実行します。

1. mode 1 コマンドの次に rate コマンドを追加します。
このテストが性能を測定しないので、gap 設定は重要ではありません。
2. プログラムを実行する時、正しい rate (2500)、mode (1)、および clk (75) の値を指定します。
3. dit コマンドの次に、dit_stats コマンドを stats コマンドに置き換えます。

図 19 に、1× 2.500-Gbaud デザイン・バリエーション上で FPGA および TI 6488 DSP が双方向データの整合性テストのためにプログラムされる nios2 ターミナル・ウインドウの例を示して、以下のステップが実行されます。

1. nios2 ターミナルの Serial RapidIO> プロンプトで、以下のコマンド・シーケンスを入力します。

```
stop
r
rate 2500
mode 1
clk 75
pload 256
init
```

2. CCS ドライバ・ウインドウで、左側の **Run** アイコンをクリックして、プログラムを実行します。

3. Serial RapidIO> プロンプトで、次のコマンドを入力します。

```
dit 80000000 ←
```

4. ダイナミック統計収集を表示するために、Serial RapidIO> プロンプトで、以下のコマンドを入力します。

```
stats ←
```

5. ダイナミック統計ディスプレイを停止するために、Serial RapidIO> プロンプトで、以下のコマンドを入力します。

```
z ←
```

図 19 に、stats コマンドから出力の例を示します。FPGA および TI DSP の両方から開始されるトラフィックを示します。

図 19. 1× 2.500 Gbaud で表示される双方向データの整合性テスト stats

```

Nios II EDS 9.0
Baud Rate : 2500 Mbaud
TX Throughput 1.703819 Gbps TX Efficiency 85.190948%
RX Throughput 1.703819 Gbps RX Efficiency 85.190948%

Sample Window ..... 134217227 cycles
RapidIO Payload ..... 256 bytes

TX Packets ..... 2977591
RX Packets ..... 1488869

IO S WR Bytes ..... 381125308
IO M RD Bytes ..... 12544

IO S RD Bytes ..... 381125308
IO M WR Bytes ..... 12544

Pkts Cancelled ..... 0
Pkts Not Accepted ..... 0
Pkt Retrys ..... 0
Pkt CRC Errors ..... 0
Symbol Errors ..... 0
Char Errors ..... 0
Pkts Dropped by Transport Layer ..... 0

Serial RapidIO>

```

1× 3.125-Gbaud デザイン・バリエーションに対する双方向データの整合性テストの実行

1× 3.125-Gbaud デザイン・バリエーションに対する双方向の性能テストを実行する前に、正しいバリエーションのための FPGA をプログラムし、そして双方向の性能テストおよび正しいレート設定に対する TI 6488 DSP をプログラムする必要があります。

適切な変更で 30 ページの「1x 2.500-Gbaud デザイン・バリエーションに対する双方向データの整合性テストの実行」の手順に従って、1x 3.125-Gbaud バリエーションをプログラムして実行します。プログラムを実行する時、正しい rate (3125)、mode (1)、および clk (84) の値を指定します。このレートの正しい clk 設定に対して、7 ページのを参照してください。このテストが性能を測定しないので、gap 設定は重要ではありません。

Doorbell メッセージ・テストの実行

Doorbell メッセージ・テストは以下の 2 つの部分で構成されています。

- **FPGA-to-DSP:** FPGA 上の RapidIO MegaCore ファンクションは Doorbell メッセージを TI DSP に送信し、そしてテストは Doorbell Sent ステータスをレポートします。
- **DSP-to-FPGA:** TI DSP は FPGA 上の Doorbell メッセージを RapidIO MegaCore ファンクションに送信し、そしてテストは RapidIO MegaCore ファンクションが受け取る Doorbell メッセージをレポートします。

テストの 2 つの部分の用に TI DSP を異なる方法でプログラムする必要があります。

アプリケーションを実行する前に、実行するテストの部分に対して、アルテラ FPGA をコンフィギュレーションして、正しいプログラムで TI 6488 DSP をロードする必要があります。

アルテラ FPGA をコンフィギュレーションするには、RapidIO デザイン・バリエーションに応じて、14 ページの「1x 1.250-Gbaud デザイン・バリエーションに対するアルテラ FPGA のプログラミング」または 15 ページの「残りの RapidIO デザイン・バリエーションに対するアルテラ FPGA のプログラミング」の説明に従います。



このデザイン・バリエーションのために FPGA を再プログラムしましたが、このバリエーションに対して create-this-bsp および create-this-app をすでに実行した場合、15 ページのステップ 1 および ステップ 2 のみを実行する必要があります。

FPGA-to-DSP Doorbell メッセージ・テストのロードおよび実行

RapidIO デザイン・バリエーションに対する FPGA-to-DSP Doorbell メッセージ・テスト付き TI 6488 DSP をロードするには、以下のステップに従います。

1. **main_unidirectional.c** ファイルがコンパイルされることを指定するには、**srio_test_<rate>.pjt** ファイルを編集します。
2. 17 ページの「CCStudio IDE v3.3 の実行」の説明に従ってください。
3. RapidIO デザイン・バリエーションに応じて、18 ページの「1x 1.250-Gbaud デザイン・バリエーションに対する TI 6488 DSP のプログラミング」または 19 ページの「残りのデザイン・バリエーションに対する TI 6488 DSP のプログラミング」の説明に従います。

FPGA および TI 6488 DSP 上で FPGA-to-DSP Doorbell メッセージ・テストを実行するには、以下のステップに従います。

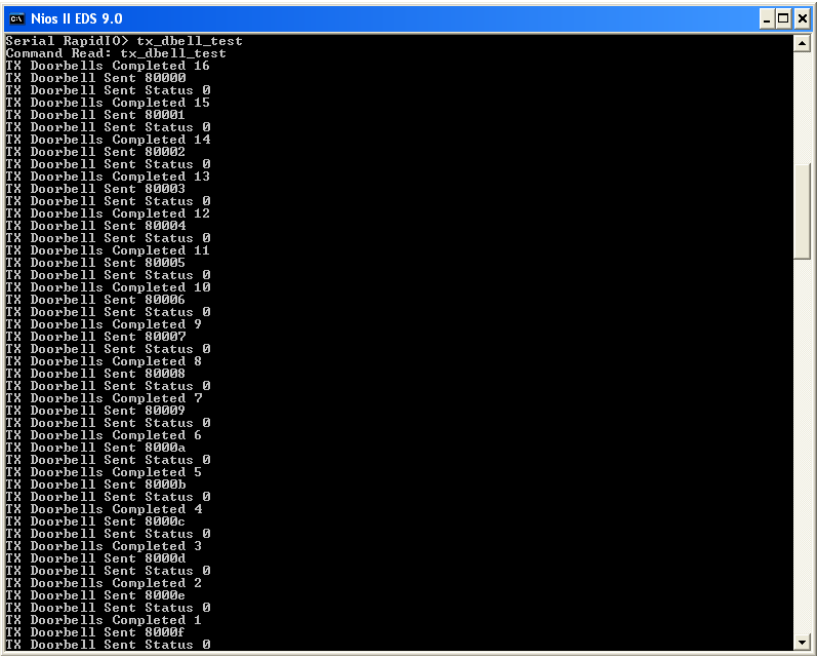
1. FPGA をコンフィギュレーションおよびプログラムした時、開いた nios2 ターミナル・セッションに戻ります。

2. Serial RapidIO> プロンプトで、次のコマンドのシーケンスを入力します。

```
link
init
tx_dbell_test
```

tx_dbell_test コマンドは、16 の Doorbell メッセージの 5 つのグループを TI DSP に送るよう FPGA での RapidIO MegaCore ファンクションに指示します。テストが Doorbell メッセージのグループを送信するよう RapidIO MegaCore ファンクションに指示すると、テストはグループにおける各 Doorbell メッセージのステータスおよび内容を検証します。図 20 に、FPGA での RapidIO MegaCore ファンクションから TI DSP まで発信される Doorbell メッセージの最初のグループのために nios2 ターミナルへのテスト出力を示します。

図 20. FPGA から TI DSP への成功した最初のグループの Doorbell メッセージ・テスト出力



```
Serial RapidIO> tx_dbell_test
Command Read: tx_dbell_test
TX Doorbells Completed 16
TX Doorbell Sent Status 0
TX Doorbells Completed 15
TX Doorbell Sent Status 0
TX Doorbells Completed 14
TX Doorbell Sent Status 0
TX Doorbells Completed 13
TX Doorbell Sent Status 0
TX Doorbells Completed 12
TX Doorbell Sent Status 0
TX Doorbells Completed 11
TX Doorbell Sent Status 0
TX Doorbells Completed 10
TX Doorbell Sent Status 0
TX Doorbells Completed 9
TX Doorbell Sent Status 0
TX Doorbells Completed 8
TX Doorbell Sent Status 0
TX Doorbells Completed 7
TX Doorbell Sent Status 0
TX Doorbells Completed 6
TX Doorbell Sent Status 0
TX Doorbells Completed 5
TX Doorbell Sent Status 0
TX Doorbells Completed 4
TX Doorbell Sent Status 0
TX Doorbells Completed 3
TX Doorbell Sent Status 0
TX Doorbells Completed 2
TX Doorbell Sent Status 0
TX Doorbells Completed 1
TX Doorbell Sent Status 0
```

TI DSP は 64 つの未処理のメッセージのみを処理するようにプログラムされます。したがって、Doorbell メッセージの 5 つのグループにはタイム・アウトします。グループにおける各 Doorbell メッセージの Doorbell Sent Status はタイム・アウトを示します。図 21 に、Doorbell Sent Status において FPGA での RapidIO MegaCore ファンクションから送信される Doorbell メッセージの 5 つのグループにある各メッセージのタイム・アウトを示しているテスト出力です。

図 21. FPGA から TI DSP への 5 番目のグループのタイム・アウトを示した Doorbell メッセージ・テスト出力

```

Nios II EDS 9.0
TX Doorbells Completed 16
TX Doorbell Sent 80000
TX Doorbell Sent Status 2
TX Doorbells Completed 15
TX Doorbell Sent 80001
TX Doorbell Sent Status 2
TX Doorbells Completed 14
TX Doorbell Sent 80002
TX Doorbell Sent Status 2
TX Doorbells Completed 13
TX Doorbell Sent 80003
TX Doorbell Sent Status 2
TX Doorbells Completed 12
TX Doorbell Sent 80004
TX Doorbell Sent Status 2
TX Doorbells Completed 11
TX Doorbell Sent 80005
TX Doorbell Sent Status 2
TX Doorbells Completed 10
TX Doorbell Sent 80006
TX Doorbell Sent Status 2
TX Doorbells Completed 9
TX Doorbell Sent 80007
TX Doorbell Sent Status 2
TX Doorbells Completed 8
TX Doorbell Sent 80008
TX Doorbell Sent Status 2
TX Doorbells Completed 7
TX Doorbell Sent 80009
TX Doorbell Sent Status 2
TX Doorbells Completed 6
TX Doorbell Sent 8000a
TX Doorbell Sent Status 2
TX Doorbells Completed 5
TX Doorbell Sent 8000b
TX Doorbell Sent Status 2
TX Doorbells Completed 4
TX Doorbell Sent 8000c
TX Doorbell Sent Status 2
TX Doorbells Completed 3
TX Doorbell Sent 8000d
TX Doorbell Sent Status 2
TX Doorbells Completed 2
TX Doorbell Sent 8000e
TX Doorbell Sent Status 2
TX Doorbells Completed 1
TX Doorbell Sent 8000f
TX Doorbell Sent Status 2
Serial RapidIO>

```

DSP-to-FPGA Doorbell メッセージ・テストのロードおよび実行

DSP-to-FPGA Doorbell メッセージ・テスト付き TI 6488 DSP をロードして、FPGA および TI 6488 DSP 上でこのテストを実行するには、以下のステップに従います。

1. FPGA をコンフィギュレーションおよびプログラムした時に開いた nios2 ターミナル・セッションに戻ります。
2. Serial RapidIO> プロンプトで、次のコマンドのシーケンスを入力します。
init ←
3. **main_tx_dbell.c** ファイルがコンパイルされることを指定するには、**srio_test_<rate>.pjt** ファイルを編集します。
4. 17 ページの「CCStudio IDE v3.3 の実行」の説明に従ってください。
5. RapidIO デザイン・バリエーションに応じて、18 ページの「1×1.250-Gbaud デザイン・バリエーションに対する TI 6488 DSP のプログラミング」または 19 ページの「残りのデザイン・バリエーションに対する TI 6488 DSP のプログラミング」の説明に従います。

TI 6488 DSP は 16 Doorbell メッセージを生成して、FPGA 上の RapidIO MegaCore ファンクションに送信します。図 22 に、テストが完了した後の CCS ウィンドウを示します。

6. Serial RapidIO> プロンプトで、次のコマンドのシーケンスを入力します。
get_dbells ←

図 23 に示すとおり、nios2 ターミナルは FPGA 上の RapidIO MegaCore ファンクションによって受信される Doorbell メッセージに関する情報を表示します。

図 22. TI DSP から FPGA に送信される Doorbell メッセージを表示する Code Composer Studio

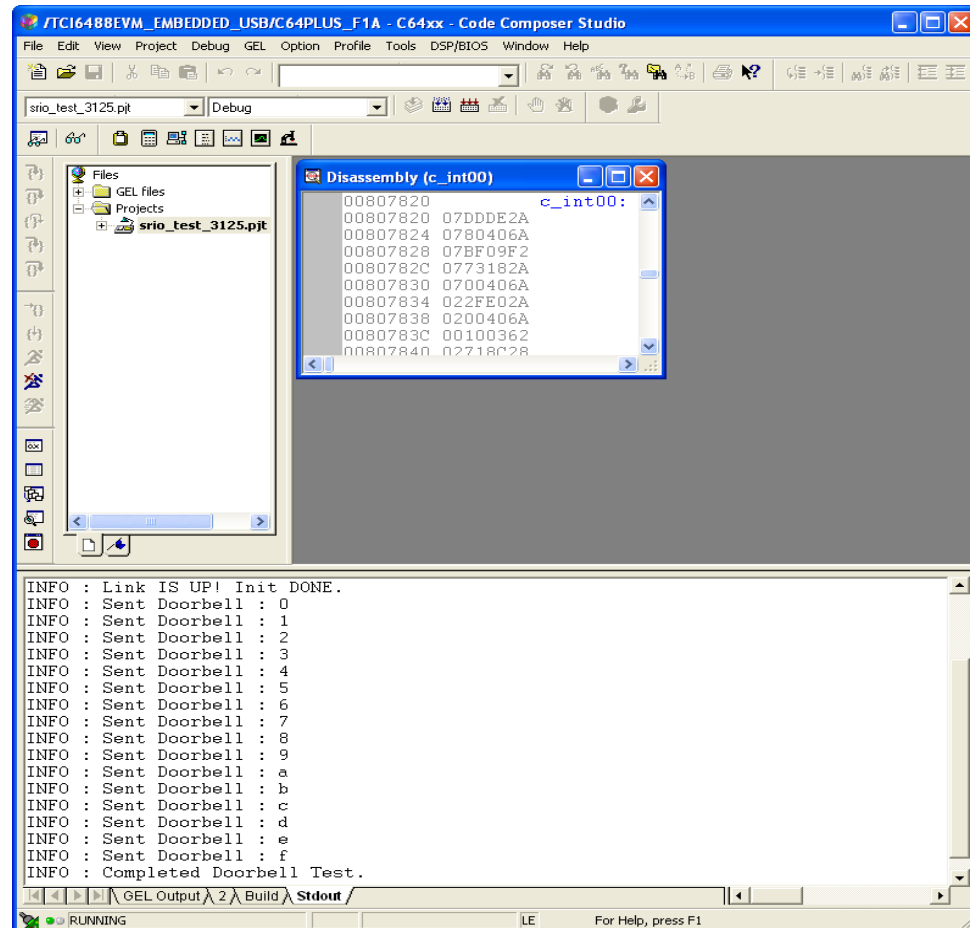
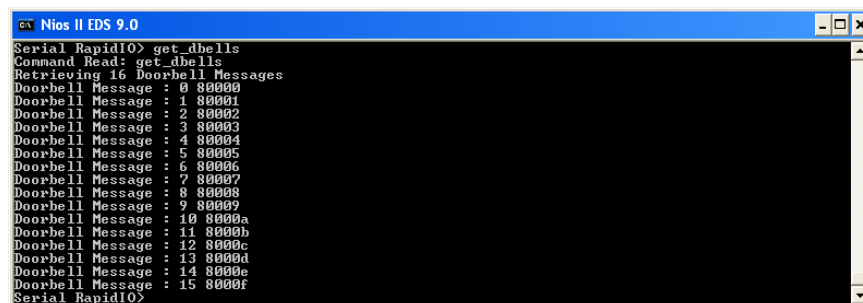


図 23. FPGA で受信される Doorbell メッセージ



SWRITE または NWRITE_R トランザクションの生成

単一方向の性能、双方向の性能、および双方向のデータの整合性のために説明されるコマンド・シーケンスで、RapidIO MegaCore ファンクションは NWRITE にトランザクションだけを生成します。FPGA から SWRITE または NWRITE_R トランザクションをテストするために、コマンド・シーケンスにおける以下の追加のコマンドを挿入して、init コマンドに続いて、start コマンドに先行します。

- SWRITE トランザクションを生成するには、以下のコマンドを入力します。:
load 0x1040c 0x0008ff02 ←
- NWRITE_R トランザクションを生成するには、以下のコマンドを入力します。:
load 0x1040c 0x0008ff01 ←

 レジスタ 0x1040c、入力 / 出力スレーブ・マッピング・ウインドウ 0 コントロール・レジスタについて詳しくは、[「RapidIO MegaCore Function User Guide」](#) を参照してください。

メンテナンス・ライト・トランザクションの生成

メンテナンス・ライト・トランザクションを生成するには、Serial RapidIO> プロンプトで、以下のコマンドを入力します。

```
rmw <addr> <value>
```

ここで、<addr> はリモート処理エンドポイントで書き込むコンフィギュレーション・レジスタのアドレス・オフセットです。例えば、以下のコマンドを入力することにより、

```
rmw 0x60 0x00AAFFFF
```

TI 6488 DSP RapidIO ペリフェラルのベース・デバイス ID を 0xAA にプログラムします。

メンテナンス・リード・トランザクションの生成

メンテナンス・リード・トランザクションを生成するには、Serial RapidIO> プロンプトで、以下のコマンドを入力します。

```
rnr <addr>
```

ここで、<addr> はリモート処理エンドポイントで読み出すコンフィギュレーション・レジスタのアドレス・オフセットです。例えば、前項で rmw コマンドを実装した後に、次のコマンドを入力すると、

```
rnr 0x60
```


0x00AAFFFF の値を返るべきです。

SignalTap II エンベデッド・ロジック・アナライザを使用した RapidIO-Avalon インタフェースの表示

リファレンス・デザインには、定義済みの SignalTap II エンベデッド・ロジック・アナライザ・ファイル、**stp1.stp** が含まれます。このファイルはシステム・インタコネクタ・ファブリックおよび RapidIO MegaCore ファンクションの間で通信する信号のサブセットを定義します。また、SignalTap II エンベデッド・ロジック・アナライザはそれらをキャプチャするべきであると指定します。**.stp** ファイルに信号を追加することができます。この項では、SignalTap II エンベデッド・ロジック・アナライザを起動する方法および信号を表示する方法について説明します。

信号を表示するのに SignalTap II エンベデッド・ロジック・アナライザを使用するには、以下のステップに従います。

1. nios2 ターミナル・セッションを出入します。
2. 現在のデザイン・バリエーションに対して、メインのデザイン・ディレクトリ、**altera_srio_ref_design/srio_<rate>_<mode>**に移動します。
3. Quartus II プロジェクト・ファイル、**srio_<rate>_<mode>.qpf**を開きます。
4. Tools メニューの **SignalTap II Logic Analyzer** をクリックします。
5. SignalTap II ウィンドウで、**Setup** をクリックします。**Hardware Setup** ダイアログ・ボックスが表示されます。
6. **Hardware Setup** ダイアログ・ボックスで、**Currently selected hardware** に **USB-Blaster** を選択します。
7. **Close** をクリックします。
8. Nios II コマンド・シェルに戻れます。
9. プログラムを再ロードするには、デザイン・バリエーションに応じて、14 ページの「**1×1.250-Gbaud デザイン・バリエーションに対するアルテラ FPGA のプログラミング**」または 15 ページの「**残りの RapidIO デザイン・バリエーションに対するアルテラ FPGA のプログラミング**」の説明に従います。
10. 以下のコマンドを入力して、nios2 ターミナル・セッションを起動します。


```
nios2-download --device=1 -g srio_test.elf ←
nios2-terminal --device=1 ←
```
11. 20 ページの「**1×1.250-Gbaud デザイン・バリエーションに対する単方向の性能テストの実行**」の説明に従って、nios2 ターミナルで、start コマンドを含む最初のコマンドをテスト・コマンド・シーケンスに入力します。
12. SignalTap II ウィンドウに戻って、**Autorun Analysis** アイコンをクリックします。 24 にアイコンを示します。

24. SignalTap II Embedded Logic Analyzer Autorun Analysis アイコン

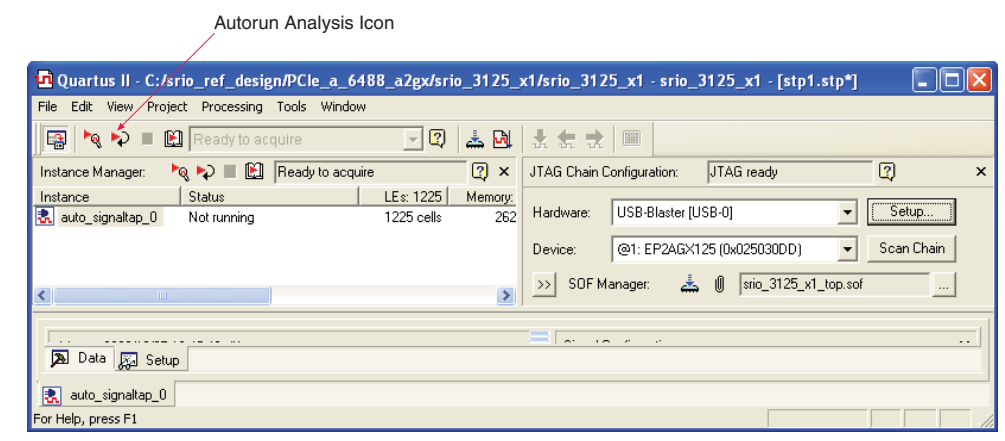
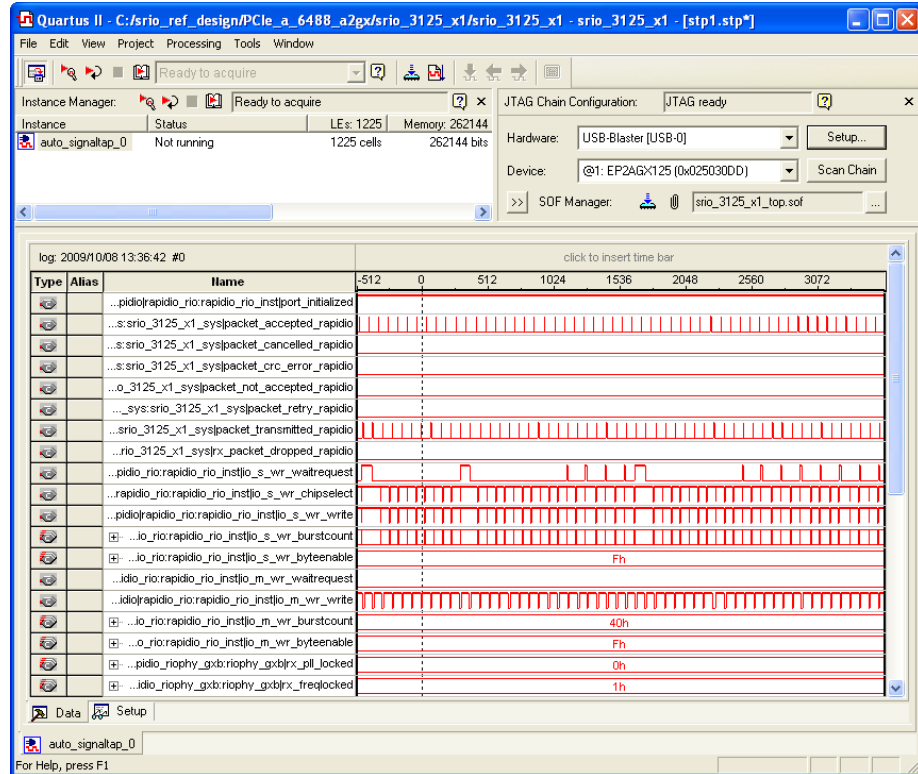


図 25 に、SignalTap II エンベデッド・ロジック・アナライザを示します。ウィンドウには、RapidIO MegaCore ファンクションの I/O スレーブ・ライトおよび I/O マスター・ライトにおける動作が表示されます。

図 25. SignalTap II エンベデッド・ロジック・アナライザ・セッション



性能の概要

このセクションには 3 つのデザイン・バリエーションの性能観測をそれぞれ示す 3 つのグラフが含まれます。

Quartus II ソフトウェア v9.0 と CCS IDE v3.3、アセンブリ・バージョン 509310 Rev D TI DSP カード、および Arria II GX FPGA 開発ボード Rev B を使用することで、この項でレポートされる性能結果を得ました。

図 26 ～ 図 28 に、このデザインを使用して、測定された性能値を示します。各グラフでは、3 種類の RapidIO WRITE トランザクションの性能を示します。

図 26. 1.250 Gbaud で ×1 RapidIO MegaCore ファンクションにおける増加するペイロードのあるスループット

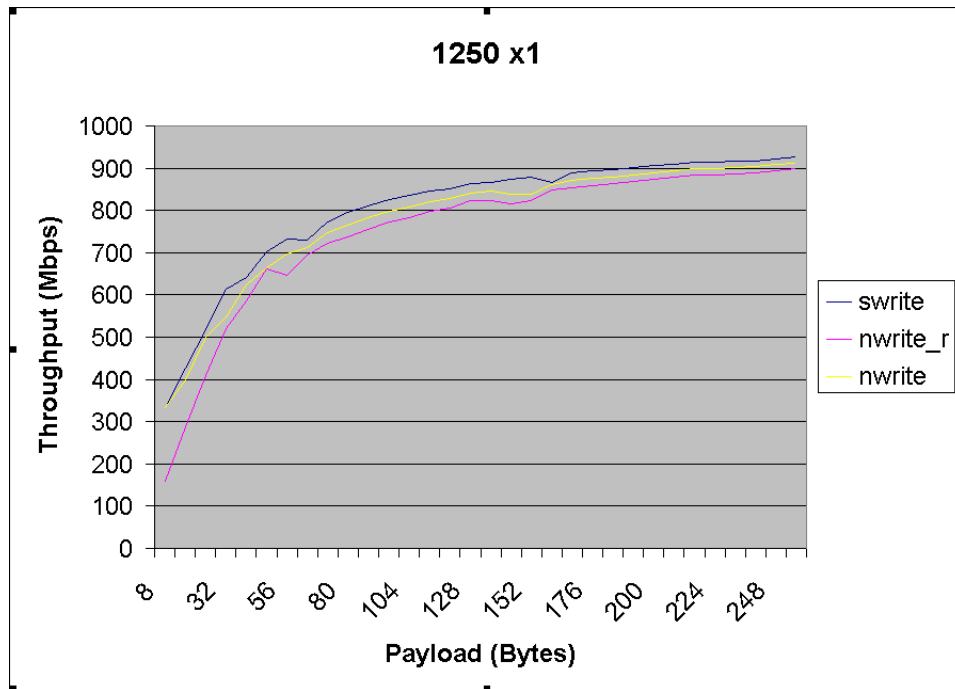


図 27. 2.500 Gbaud で ×1 RapidIO MegaCore ファンクションにおける増加するペイロードのあるスループット

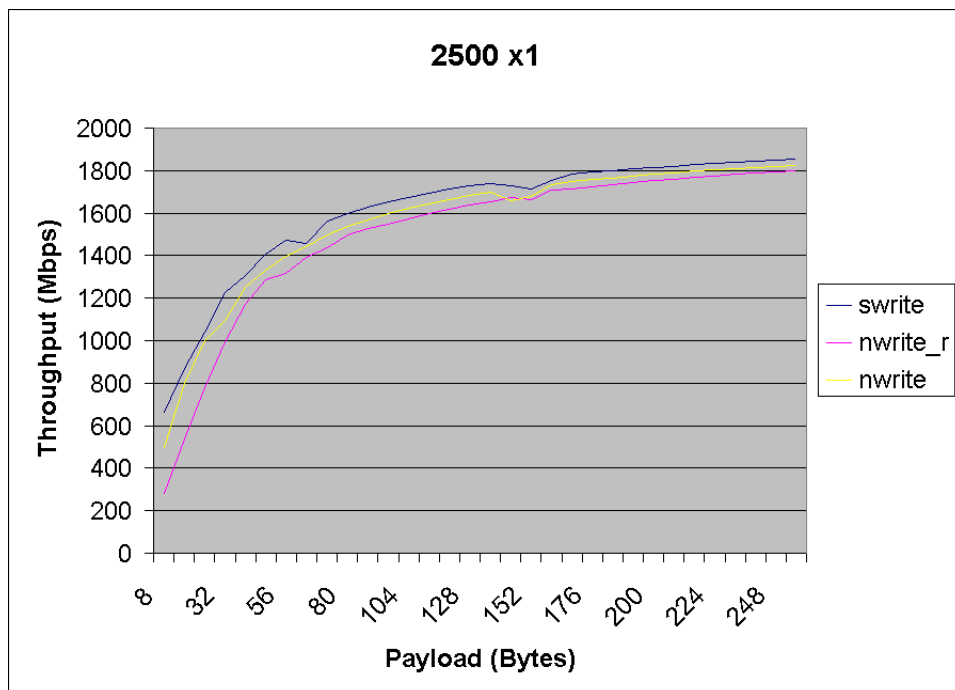
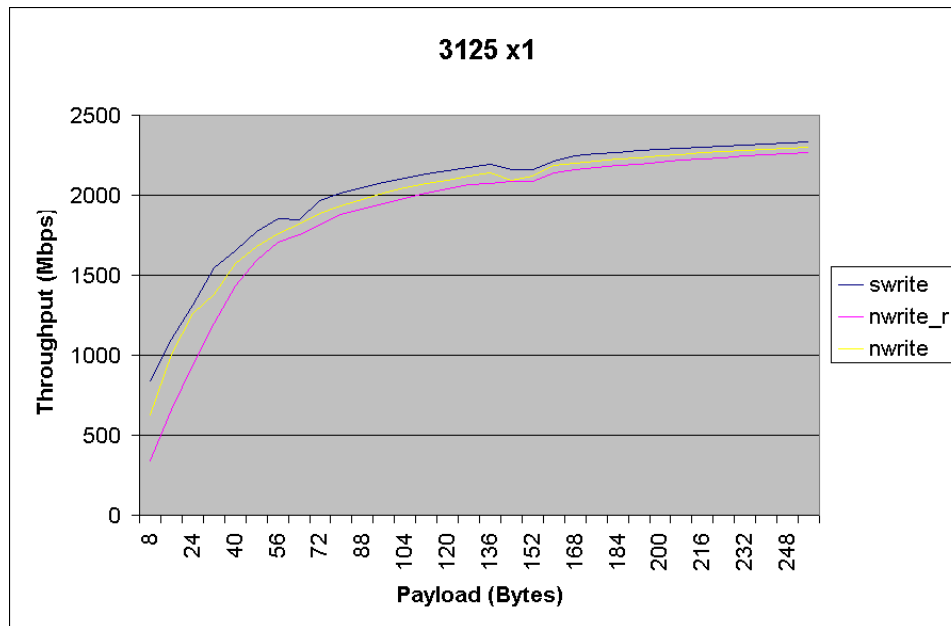


図 28. 3.125 Gbaud で ×1 RapidIO MegaCore ファンクションにおける増加するペイロードのあるスループット



デザインの制限

この項では、このリファレンス・デザインの最新バージョンの既知の制限を記載します。

ハードウェアおよび RapidIO MegaCore ファンクション・ドライバの両方は次のトランザクションをサポートします。しかしながら、このアプリケーションの最新バージョンにはこれらのトランザクションの結果が含まれません。

- メンテナンス・ポート・ライト
- メンテナンス・ポート・リード

デザインは 1× 1.250 Gbaud デザイン・バリエーションにのみ DMA 手法を使用して、ライン・レート・トラフィックでリンクを飽和状態にすることができます。パケット・ジェネレータ手法で、デザインは RapidIO プロトコルの他のバリエーションのためのリンクを飽和状態にすることができます。

現在のデザインはトランザクション・ミックスを生成しません。入力/出力スレーブ・マッピング・ウインドウ 0 コントロール・レジスタを設定することにより、手動でトランザクション・タイプを切り換える必要があります。

参考資料

このアプリケーション・ノートは、以下の資料に関連する情報を記載または参照しています。

- 「[RapidIO MegaCore Function User Guide](#)」
- EVM に添付されている *TMS320TCI6488 EVM Quick Start Installation Guide*

改訂履歴

表 7 に、このアプリケーション・ノートの改訂履歴を示します。

表 7. 改訂履歴

日付およびドキュメント・バージョン	変更内容	概要
2009 年 12 月 v1.0	初版。	—



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.



I.S. EN ISO 9001